Gator Protocol: A decentralized on-chain marketplace for inference from AI models

By Timmy0x

The current market for AI inference is dominated by centralized infrastructure, mostly controlled by companies such as Google, Microsoft, and OpenAI. This centralization forces high trust assumptions about the integrity of some companies and limits the free market's ability to decide prices.

In this paper, I present the Gator Protocol as a solution: a decentralized inference marketplace for AI models, where computations are done off-chain and settled on-chain. For a node to participate in the Gator Protocol, they must stake a certain amount of $GAT tokens. These tokens can be slashed if nodes act maliciously. The software for completing requests and submitting them to the protocol will be fully open-sourced and user-friendly so people can rent out their idle hardware to the Gator Protocol and get paid. When someone creates an on-chain request to a certain AI model through the Gator Protocol, entropy for that request is made public, allowing deterministic results so multiple nodes can verify that the computation was valid. If multiple nodes return different results, some of their $GAT will be slashed depending on the magnitude of the difference in responses. If the user selects only one confirmation, a kind of node called a verifier will use the entropy provided in a request to check if the computation is valid. If it is not, that node will submit the proof to the DAO and a vote will be held to slash the node's tokens and reward the verifier.

To fairly allocate compute subsidies in the form of $GAT, we introduce the Gator Protocol DAO which can vote on giving out $GAT. 30% of $GAT tokens will be provided to this DAO.

Introduction

Artificial intelligence (AI) refers to the intelligence of machines or software. Its uses are far-reaching and incredibly disruptive. AIs often appear in the form of LLMs optimized for conversations, autonomous agents, and algorithms. These solutions are mostly hosted on centralized infrastructure, controlled by tech companies such as Google, Microsoft, and OpenAI. The current market for AI inference is dominated by centralized infrastructure, used by individuals, businesses, or decentralized protocols. This centralization forces high trust assumptions about the integrity and reliability of those companies, which have track records of unethically using data, manipulating customers, censoring materials, or being influenced by nation-state actors. Developers looking to build applications integrated with AI are forced to choose between a few companies that can close their APIs at any time and are making increased efforts to lock in their developers and users. The alternative option for developers is to self-host models. Although it will become easier over time to self-host, it does not productively scale and will continue to be less efficient than a competitive marketplace.

Creating a competitive, permissionless marketplace requires a kind of machine that operates virtually across a network. Blockchains are distributed computers that settle on state and keep records. Blockchains are the ideal decentralized computing platform.

We propose a solution to the trust problems of AI and the scalability problems of self-hosting in the form of the Gator Protocol, a marketplace for inferring AI models. The Gator Protocol functions similar to an order book of requests for AI model inference. Bitcoin-style tokenomics rewards those who successfully deliver inference results from AI models. The DAO controls a sizable share of the token supply to vote on funding for those who provide value to the Gator Protocol ecosystem.

The Gator Protocol requires blockchain technology, which can settle the state of computer programs and complete records through a distributed network of computers. Given advancements in technology surrounding sovereign chains, which inherits the security of Ethereum while also having fast transaction finality and low gas fees. The Gator Protocol smart contracts will be hosted on a custom Ethereum Layer 3. Although it is important to note that the Gator Protocol will be mostly chain-agnostic, so as to be able to provide responses through smart contracts on various EVM chains.

Please note the definitions of some terminology which will be used throughout this whitepaper and to describe various parts of the Gator Protocol:

- **Node:** a computer which runs software (the “client”) that connects to the Gator Protocol to deliver results from AI models.
- **Verifier/verifier node:** a computer which runs software to verify nodes’ computations and create on-chain votes to slash malicious nodes tokens.

- **Client:** any kind of software which helps a person or group of persons operate a node.
- **Verifier client:** any kind of software which helps a person or group of persons operate a verifier node.
- **User:** any person who directly or indirectly requests results from AI models using the Gator Protocol.
- **Interface:** any kind of software which delivers the results of inference from the Gator Protocol to a user.

This whitepaper is divided into the following sections:

1. Architectural overview. We outline the four main parts of the Gator Protocol.
    a. On-chain smart contract
    b. Node client
    c. $GAT
    d. DAO
2. GatorChain
3. Security
4. Performance
5. Tokenomics
6. Cost effectiveness
7. Value of our solution
8. Use cases
9. Governance
10. Appendix
    a. Ensuring deterministic results from models.

1. Architectural overview

The core functional objective of the Gator Protocol is to connect people who need to infer from an AI model to those who have compute, while evading censorship. Nodes provide access to inference from AI models, earning tokens for completing requests and submitting them to the smart contract. Nodes stake $GAT, an ERC20 token, through the main smart contract to settle disputes over different responses to model requests and to prevent Sybil attacks. Users pay for model responses with $GAT tokens. $GAT is used to vote on DAO proposals.

Users select an odd-numbered amount of nodes to return responses. We call this number of responses the number of confirmations. We foresee multiple responses from nodes as being more of a power use case: mostly for on-chain access to algorithms which need extremely high security.

The Gator Protocol has five parts:

1. On-chain smart contract
2. Mempool
3. Node client
4. $GAT

5. DAO

1.A Smart contract

The smart contract is the central part of the Gator Protocol. The smart contract has the following functions:

1. **Registering models.** Models registered through the Gator Protocol can optionally provide on-chain relevant information to people interested in running the model and completing requests on-chain.
2. **Creating requests.** Anyone who wants to permissionlessly create a request to a certain model must complete an on-chain transaction that provides the parameters so the request can be completed. These parameters include the amount of nodes they want to process the request, an odd number.
3. **Delivering results.** Nodes listen in on the smart contract and once they pick up on a request, they rush to complete it and provide the result quickly back to the smart contract.

Entropy is a parameter for the user. This entropy is used by nodes as a seed for inferring from the AI models, so multiple queries from the same models can give the same outputs on different computers.

If only one confirmation was selected, the node's response is assumed to be true. A verifier node checks to see if the computation is valid and if it is not, it submits a proof to the DAO who can slash the node’s tokens and reward the verifier node.

If multiple confirmations is selected: once all the responses are submitted, the smart contract compares the similarity of the responses, and if it is below a certain threshold specified during model creation, the node that returned the response deviated from the response that the majority of nodes returned slashed. In an edge case where each node returns different responses, none of their tokens will be slashed.

$$Rv = Majority(\{R_1, R_2 \ldots R_n\})$$

For speed purposes, the smart contract will be deployed on an Ethereum layer 3, which provides secure settlement, fast finality, and low costs.

1.B Mempool

Speed is extremely important, especially for consumer applications. Even with fast blockchains, blockchain settlement will be a disadvantage unless the inference can be done parallel to the settlement. A private mempool–where peers connect and interfaces–is the best option for this because it is off-chain, upgradeable, and flexible.

Interfaces can run the mempool software, connect with peers (other interfaces), and broadcast requests to the other peers. Nodes can listen in on the mempool, and do the inference before the transaction is confirmed. This means as long as the settlement time does not exceed the time it takes the node to perform the inference, the Gator Protocol will not have a speed disadvantage.

1.C Node client

One of the core principles of the Gator Protocol is security through inclusiveness. The more nodes that complete and submit requests, the more secure the network is because it reduces the probability of nodes submitting incorrect responses for malicious reasons.

For this reason, we aim to make the original Gator Protocol client as easy to use as possible such that anybody can install a desktop app, stake some $GAT, select an LLM, download that LLM, and participate in the Gator Protocol to earn $GAT.

The client will frequently check the smart contract for new requests, and rush to complete and submit them. Because the smart contract is on a sovereign chain, the cost of submitting a response will be trivial.

It is important to note that Gator Protocol does not have a fixed fee per length of a response. When a request is created, the fee for miners is decided by the requester. The speed of fulfillment will be determined by the fee for nodes, similar to an asset order book where it takes more time for lower bids to be fulfilled. For more information about how nodes are compensated, read 1.E.

$$RID = Hash(EncodePack(sender,\ modelID,\ prompt,\ bid,\ entropy))$$

1.D DAO

The DAO has two main purposes: incentives, and slashing.

Incentives

To incentivize builders to create node clients, interfaces, apps, new use cases, and upgrades for the Gator Protocol, it is necessary to have an organization in charge of allocating capital and resources to projects using the Gator Protocol. The DAO will be in charge of growing the ecosystem. 10% of all $GAT tokens will be transferred to the DAO.

The DAO has the option to sell these tokens to raise capital or send $GAT tokens to subsidize the compute costs of builders.

For a proposal to pass, a 70% supermajority must vote for it.

$$Proposal\ Passes\ If\ \frac{V_{favor}}{V_{total}} > 0.70$$

Slashing

As mentioned in 1.A, if only one confirmation is requested by the user, verifiers will rush to check if the node's inference is valid by doing the same inference with the same entropy. If it is not, the verifier can submit a request to the DAO for that node's token stake to be slashed. The amount that nodes tokens are slashed is decided when the model is registered. If it passes, the verifier will receive a reward in the form of $GAT. This reward is decided by the DAO.

1.E $GAT

See the Tokenomics section for more information about token distribution.

Each time a node completes a request, some $GAT tokens are issued to that node. Every 1 year, the request subsidy is decreased by 20%. This request subsidy is issued until $GAT reaches its maximum supply of 42 million.

Using R_t to describe the Rate of rewards over time, we can express the equation as:

$$R_{t+1} = R_t \times (1 - 20)$$

To prevent malicious actors from registering fake models and spamming the smart contract with millions of fake requests to game the token supply system, only certain models will receive token subsidies. These models will be decided by the DAO.

We believe a secure ecosystem cannot function without a secure asset. This is a core value of the Gator Protocol, and why we decided to make $GAT fixed supply.

See Figure 2, a complete diagram of how the Gator Protocol functions.

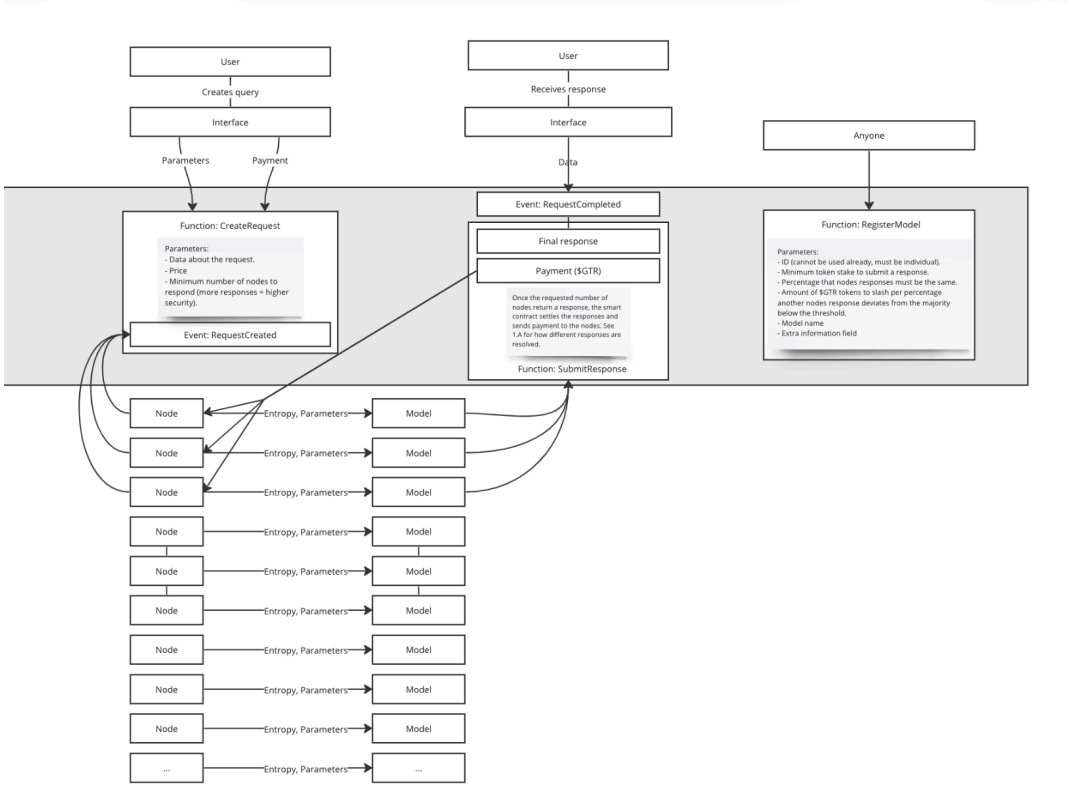


Figure 2.

2. Chain management

GatorChain

Most of the projected use cases of the Gator Protocol will be LLM inference and general inference from large AI models; as opposed to the on-chain use cases already specified. The Gator Protocols competitive, permissionless market will easily facilitate this.

For consumers, speed and low-cost transactions are important. The GatorChain will facilitate this with a custom blockchain, that inherits the security of Ethereum using rollup technology and abstracts gas away by making $GAT the native token of the rollup.

Custom interfaces will be accessible through the website, abstracting away the bridging component by enabling direct payment in various cryptocurrencies from other blockchains and swapping those tokens for $GAT on GatorChain.

The goal of the GatorChain is to be a blockchain that you don’t think about. While it exists and powers the core protocol, the bridges, token, and any other blockchain components are meant to be abstracted away through an interface where devs can load credits onto an API. The UX of the chain will reflect this mission, with low fees and near-instant finality.

AnyGator

AnyGator is infrastructure for connecting AI to on-chain smart contracts. There are various ways to achieve this, and it is still undecided which is the best option to take while developing this software.

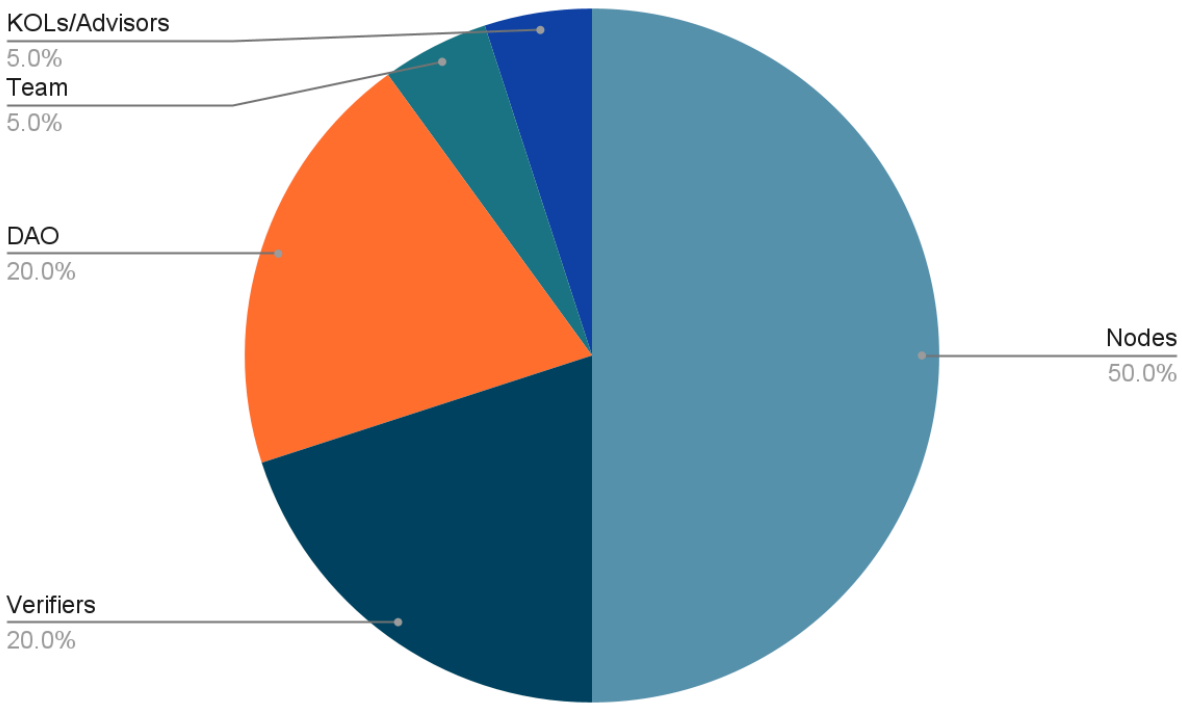
- **Wormhole**. Using Wormhole messaging, distribute bridging contracts across the protocol that swap the provided token for $GAT, bridge it to GatorChain, and send the request to the Gator Protocol.
- **CCIP**. Same approach as above using ChainLink's CCIP.
- **Replicated smart contracts.** The core smart contracts are copied onto the chosen chain. All nodes listen into this blockchain in addition to GatorChain, and fulfill requests normally. The issue with this is fragmentation, which could result in poor UX and/or more complexity for clients.

3. Tokenomics

The native token of the Gator Protocol, $GAT, is a utility token that exists solely for its various utility within the Protocol. Nodes and verifiers are required to stake $GAT to participate in the network and users pay for requests in $GAT.

To preserve the free nature of the protocol, $GAT will be created with no pre-sale, pre-mine, or liquidity mining program.

The tokenomics chart below describes the allocation of $GAT.



For more information about the DAO, see Section 5. DAO.

3.1 Vesting

- Team allocation: completely locked for six months. After six months, 10% is unlocked every quarter (3 months) until all the tokens are unlocked.
- KOLs/Advisor allocation: completely locked for six months. After six months, 10% is unlocked every quarter (3 months) until all the tokens are unlocked.
- Node/verifier rewards are unvested and immediately accessible.

3.2 Node subsidies

For a node to receive a subsidy for completing a request to a certain model, that model must be whitelisted by the DAO to receive rewards. This prevents nodes creating fake models to game the reward system.

Every day,

3.3 Verifier Subsidies

After a verifier submits a valid fraud proof to the DAO

3. DAO

The Gator Protocol will have a DAO, called the Gator DAO. The DAO is responsible for intelligent governance of the Protocol and overseeing 20% of the $GAT token supply.

The DAO's goal is to maintain the protocol and fund open-source decentralized AI projects. The DAO prioritizes its

Similar to Ethereum, the Gator DAO will have a list of proposals called GPIPs. These proposals will relate to the development of the Protocol. Additionally, there will be a separate

2. Security

When nodes return responses, the majority of them must be legitimate for a valid response to be returned to the user. The central tenet of the security of the Gator Protocol is the nodes. More honest nodes equates to more security. We have mechanisms for disincentivizing malicious nodes and incentivizing honest nodes.

To dis-incentivize malicious nodes, we slash their token stake. When a user registers a model, they must provide a parameter of what percentage of the response must be the same and what percentage of a nodes tokens must be slashed per percentage point that the response varies from the majority of nodes response, or the response which the hosted with the most amount of staked $GAT returns. This design dissuades malicious actors from submitting incorrect responses because the punishment will likely be much greater than the reward.

We incentivise honest nodes to complete requests by issuing $GAT tokens. To prevent a kind of attack where a malicious actor registers a model then creates and completes millions of fake requests to earn $GAT tokens, the DAO (read Section 5: Governance) will decide which models get $GAT token incentives.

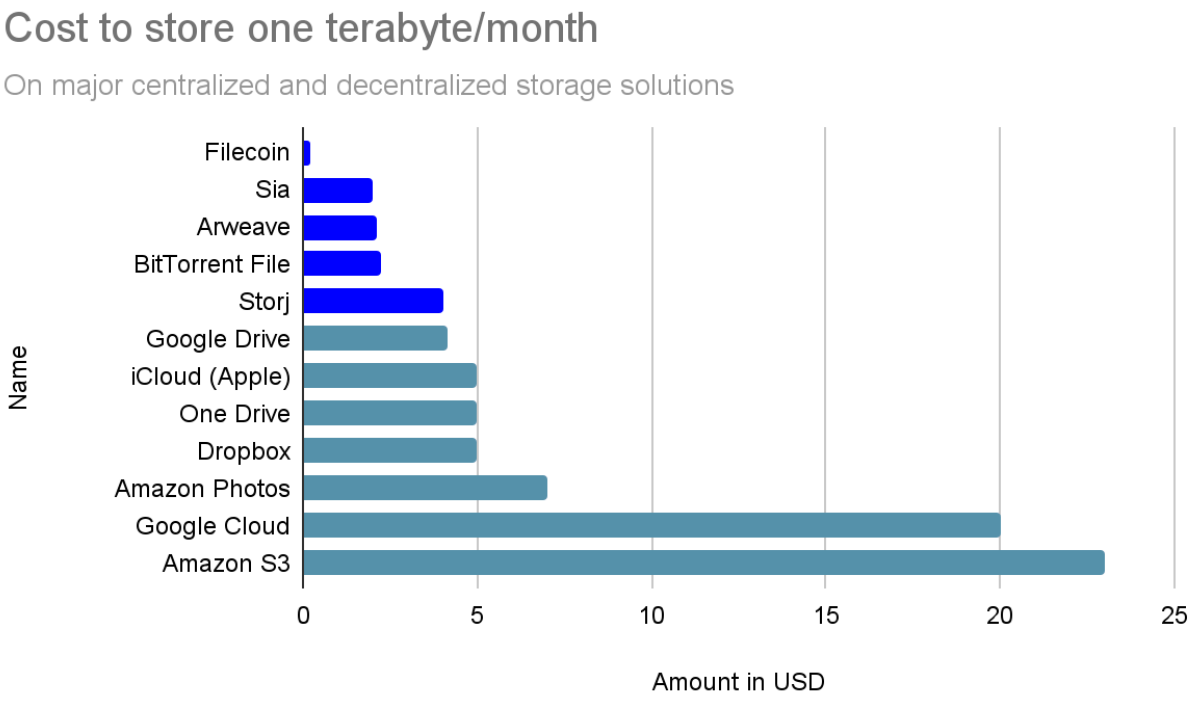
It is important that the Gator Protocol client is distributed to as many users as possible, to ensure the majority of nodes are honest. Thus, making it user-friendly is important. A good client would mean most people with a strong GPU could run an LLM on their desktop and earn $GAT.

4. Cost-effectiveness

In a world where AI and LLMs are commoditized, cost becomes very important. The Gator Protocol's decentralized, permissionless market structure is meant to facilitate competition for the lowest prices and latency. As more nodes join the Gator Protocol, inference prices will decrease because nodes will be forced to squeeze their margins to compete with others.

It is proven that decentralized markets are more efficient than central planning, and decentralized networks are generally cheaper than centralized solutions. The best example of this is decentralized data storage.

CoinGecko reports that as of May 17th, 2023, the more expensive decentralized storage solution (Storj) was cheaper than the cheapest centralized storage solution (Google Drive).



If the infrastructure for interacting with the Gator Protocol is done correctly, switching from centralized solutions to decentralized solutions becomes trivial.

Inference will be cheaper on the Gator Protocol because there are lower overhead costs for providing GPUs than centralized providers. More nodes = lower costs because nodes must compete to win requests. So nodes margins will come down significantly, which again is a big advantage when nodes are commoditized. To get people to switch LLMs is only a matter of changing an API URL and model since OpenAI open-sourced their method of inference.
* https://www.coingecko.com/research/publications/centralized-decentralized-storage-cost

Inference prices can be expressed as

$$P = \alpha \times C + \beta \times U$$

5. Value of our solution

The value of the Gator Protocol can be summarized in five main points:

1. The Gator Protocol is uncensorable, meaning the public inference of any registered model can be done through the Gator Protocol. As we have learned, centralization means censorship. The Gator Protocol eliminates this censorship by having an open marketplace to provide compute and send requests.
2. Users can select the speed of their request by paying more/less in gas and/or paying more/less of the normal fee to nodes, allowing for more flexibility. This constant competition creates incentives for models to always be faster, an important property of most consumer applications.
3. Instead of creating highly closed, restricted custom GPTs through OpenAI, anyone can create a custom GPT that is a smart contract that can create requests from the Gator Protocol. This bypasses the future possibility of payments through OpenAI, (which OpenAI would almost surely take a cut on), and any other arbitrary platform policies which are ultimately harmful for creators of custom GPTs. Take OpenAI as an example of a walled garden that hurts AI composability, and it isn't the last or only example. Smart contracts can integrate with the Gator Protocol and permissionlessly do inferences from a variety of models, or, create their own model and outsource the compute by making it open-source.
4. Anybody can provide their computing power to the Gator Protocol network to run any AI models. This compute is much in demand and the Gator Protocol will be a great place to facilitate that because of the customizability + competitiveness + no downtime. I think creating a desktop application for people to download, create a wallet for the Gator Protocol tokens, and start earning money by proving their useful processing power to run an LLM/LLMs would make it a lot easier for normies to participate in the Gator Protocol network.
5. With lots of people providing their compute, the Gator Protocol will be highly competitive. Fees will become a race for the bottom as model nodes compete for the lowest unused time on their hardware. This will encourage innovation in hardware and model speed/reliability, much in the same way that Bitcoin has encouraged innovation in SHA-256 ASICs except for useful compute that is in demand.

6. Use cases

The Gator Protocol has a wide variety of use cases, here we provide some basic examples.

- Fully decentralized GPT chat interface
- On-chain secure AI trading vaults
- Smart contract autonomous agents

6.A Fully decentralized GPT chat interface

Users load an IPFS-hosted webpage and use Web3.js to send a request to a Llama model through the Gator Protocol. To make it easier for users to get started, interfaces may want to subsidize the cost of sending requests.

6.B On-chain AI trading vaults

Someone could develop advanced AI trading strategies, register their models through the smart contract, open-source their models, and then make requests to the smart contract to decide which assets to buy and sell.

6.C Smart contract autonomous agents

Autonomous agents similar to custom GPTs can be created as smart contracts and interact with various AI models through the Gator Protocol. The possibilities of on-chain autonomous agents are enormous.

7. Governance

Bootstrapping the Gator Protocol with no ICO, pre-sale, or pre-mine presents an issue: incentivizing builders to create new decentralized applications with the protocol. While some of the first boilerplate applications will be built by the Gator Protocol team, we believe sustaining and funding new applications and use cases is very important. Also, there are some parts of the Gator Protocol that require human consensus.

The DAO has the following purposes:

- **Allocating resources and $GAT to fund builders using the Gator Protocol.** The DAO can send $GAT or other tokens to builders.
- **Whitelisting certain models for $GAT issuance.** To prevent a kind of attack where a malicious actor registers a model and spams it with requests to receive request subsidies, models registered through the smart contract will need to be whitelisted for nodes to receive token incentives. The criteria for registering tokens are
- **Managing the treasury.** The DAO controls its own independent treasury and can sell tokens to raise more capital.
- **Creating incentives.** The DAO can decide on other incentives for builders, nodes, and users.
- **Voting on protocol upgrade proposals.**
- **Slashing malicious nodes token stake.**

Inevitably, **the Gator Protocol will be upgraded**. This whitepaper does not give an exhaustive description of the features which may be included in the future.

Some protocol upgrades may require switching smart contracts or editing node clients.

To ensure clarity of consensus about these important upgrades, it is encouraged that an on-chain vote is held.

Appendix A: Ensuring deterministic results from models

When a request is created through the smart contract, entropy is generated through built-in smart contract functions and made public. nodes interested in fulfilling that request use that public entropy as a seed. This is done to ensure that all responses are deterministic. When multiple nodes submit their results to the smart contract, they must be the exact same. If they are not, then one of the nodes may be slashed.

It is important to ensure that the seed is used correctly. Enforcing this seed and making sure there are no deviations in the results is a difficult problem, especially with creating a client that can be used by everyday consumers on a wide variety of machines. Compatibility with a variety of operating system architectures, hardware, and models is important too.

In this section, I will break down how to create deterministic outputs with different pipelines and APIs.

$$P(T = 0) = 1$$

HuggingFace

HuggingFace is a widely used inference pipeline. Its Transformers library allows anyone to load pre-trained models, cache them, and query them locally. HuggingFace's documentation describes a process for creating reproducible outputs with all of their models.

https://huggingface.co/docs/diffusers/en/using-diffusers/reproducibility

In many cases, models can be more unpredictable, and using the same seed phrases doesn't mean responses will always be deterministic. There can still be small fluctuations because of low-level differences. In almost all LLMs, we can set the temperature parameter to 0 so the highest probability next token is always chosen. This changes the output slightly, although not enough for the consumer to notice.

OpenAI

OpenAI models can be used through the Gator Protocol. In this case, the Gator Protocol functions more like an oracle which delivers several responses from an API, compares them,

and returns the final result. Gator Protocol models can become deterministic by setting the temperature parameter to 0. Generally, this affects the results slightly because the next tokens are less varied, but it is trivial and not likely to be noticed.

Existing Solutions

Disclosures

This paper is for general information purposes only. It does not constitute investment advice or a recommendation or solicitation to buy or sell any investment and should not be used in the evaluation of the merits of making any investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations. This paper reflects current opinions of the authors and is not made on behalf of the Gator Protocol, or their affiliates and does not necessarily reflect the opinions of the Gator Protocol, their affiliates or individuals associated with them. The opinions reflected herein are subject to change without being updated.