Qwen

Architectural Blueprint for a Self-Improving Liquidation Arbitrage Agent on the Polygon Network

The Liquidation Arbitrage Opportunity Landscape on Polygon

The transition from a simple transaction-spending model on Hedera to a sophisticated Maximal Extractable Value (MEV) hunting strategy on Polygon requires a fundamental reorientation of the agent's purpose and capabilities [2]. On Polygon, the primary target is not merely spending a native token but capturing value from the competitive and dynamic environment of decentralized finance (DeFi), with liquidation arbitrage representing one of the most lucrative and technically demanding strategies [3] [13]. The economic viability of this endeavor is predicated on understanding the unique characteristics of the Polygon ecosystem, its dominant DeFi protocols, and the intricate balance between potential profit and operational risk. The Polygon PoS chain has demonstrated substantial throughput, processing over 4.1 billion transactions by the second quarter of 2024, with a daily active address count of 1.2 million and a total value locked (TVL) in DeFi reaching
1 billion during the same period <sup><a class="ref-sup" href="#ref-21" target="_sel
936 million, underscoring the significant capital pool available for MEV extraction [22]. This high volume of activity, combined with low average transaction fees of around $0.01, creates fertile ground for MEV bots that can operate efficiently and profitably [21].

The cornerstone of the liquidation arbitrage landscape on Polygon is the Aave v3 lending protocol, which stands as the market leader in terms of TVL, holding over $430 million on the network [22]. Its deployment on Polygon, launched in March 2022, was a clean, new installation rather than an upgrade from Aave v2, ensuring ABI and event compatibility across all its network deployments, including Ethereum Mainnet [56]. This standardization is critical for the agent, as it allows for consistent off-chain telemetry ingestion using the same logic that would work on other chains [56]. The core mechanic of Aave v3 involves maintaining overcollateralization, where the value of a user's collateral must exceed their borrowed amount plus interest to prevent insolvent debt [19]. When this

ratio deteriorates due to adverse price movements, the position becomes undercollateralized and eligible for liquidation [18]. The trigger for liquidation is the Health Factor falling below 1, a state that indicates the borrower is unable to service their loan obligations [18] [19]. Once a position is deemed unhealthy, external third-party liquidators, like the proposed agent, can step in to清算 the position [19]. The protocol's liquidation logic is encapsulated in the `LiquidationLogic` library, which performs several key functions: it validates the health factor condition, calculates the precise amount of debt to cover, applies a liquidation bonus to the collateral received by the liquidator, and deducts any applicable protocol fees from the bonus portion before transferring the remaining assets to the liquidator [19]. This structured and predictable mechanism provides a clear and reliable data stream for an automated agent to exploit.

Beyond Aave, the Polygon ecosystem hosts other significant DeFi lending platforms that must be monitored to maximize opportunity capture. Granary Finance is explicitly identified as a lending protocol operating on Polygon with active loan agreements, indicating its presence in the liquidation hunting arena [36]. Similarly, Compound III (also known as Comet) is deployed on Polygon alongside other major chains like Ethereum and Arbitrum [37]. The implementation on Polygon uses isolated markets, Chainlink oracles exclusively, and features gradual liquidation mechanics designed to mitigate cascading failures, although it still emits events analogous to a `LiquidationCall` [38]. While the provided context lacks granular details on the event structures of Granary and Compound III, their existence confirms the necessity of a multi-protocol monitoring architecture. The agent cannot afford to ignore any viable source of liquidatable positions. Furthermore, the Aave v3 protocol itself is continuously evolving, introducing features like 'Isolation Mode' for new assets, which restricts borrowing permissions to specific stablecoins and imposes debt ceilings, and 'Risk Admins', which allow DAOs or automated agents to dynamically adjust risk parameters without formal governance votes [41]. These developments directly impact the agent's ability to detect, validate, and execute liquidations, necessitating a flexible and adaptable monitoring system capable of parsing updated event logs and handling new protocol states, such as those introduced in Aave v3.3, which includes refined logging and burn functionality for undercollateralized positions [54].

The economic calculus of liquidation arbitrage on Polygon is a delicate balance of maximizing profits while minimizing costs. The primary source of profit is the liquidation bonus, a penalty applied to the collateral seized from the undercollateralized position [17]. This bonus ranges from 5% to 10%, depending on the asset's risk parameters as determined by Aave's weekly updated dynamic risk assessment system [17]. For example, common assets like ETH or stablecoins might carry a 5% bonus, while more volatile

assets like KNC could have a 10% bonus [17]. This bonus represents the gross profit margin before any execution costs are deducted. The principal costs involved are gas fees, DEX swap fees, and any fees charged by relays or flashloan providers. Research on multi-chain MEV bots provides valuable quantitative benchmarks applicable to the Polygon environment. Gas fees on Polygon are generally very low, averaging around $0.01, which makes it highly attractive for MEV extraction compared to networks like Ethereum [21]. However, during periods of intense market volatility, when multiple liquidation bots compete for the same opportunity, gas prices can spike dramatically, eroding profitability [2]. Flashloan fees, typically sourced from Aave V3, are approximately 0.09% of the borrowed amount [31]. DEX swap fees on platforms like Uniswap or SushiSwap are also a factor, though they are relatively small at around 0.3% [31]. To be profitable, the net gain from the liquidation bonus minus these cumulative costs must be positive. Based on similar analyses, a minimum price discrepancy or liquidation bonus of approximately 1% is often required to make a trade worthwhile after accounting for all expenses [31]. High-performance bots achieve success rates of 85–95%, meaning they successfully execute profitable trades in the vast majority of attempts, which is crucial for sustained profitability [31] [32]. Stress-testing simulations conducted by firms like Gauntlet highlight the extreme conditions that can arise; sudden volatility spikes, cascading liquidations, and network congestion can severely impact profitability and require robust safety mechanisms, such as circuit breakers and variable gas price distributions, to be built into the agent's core logic [34]. Therefore, the agent must not only identify opportunities but also perform a real-time net present value calculation, factoring in potential gas wars and execution latency, to determine if an opportunity is truly worth pursuing.

Metric	Description	Estimated Value / Range
Dominant Lending Protocol	Primary DeFi protocol for liquidation hunting on Polygon.	Aave v3 [22]
Aave v3 TVL on Polygon	Total Value Locked in Aave v3 on Polygon, indicating capital density.	Over $430 Million [22]
Polygon Average Tx Fee	Low-cost environment favorable for MEV.	~$0.01 [21]
Primary Liquidation Trigger	Condition that makes a position eligible for liquidation.	Health Factor < 1 [18] [19]
Liquidation Bonus	Profit for the liquidator, a percentage discount on seized collateral.	5% - 10% [17]
Flashloan Fee (Aave V3)	Cost of capital used to cover debt in an atomic transaction.	~0.09% [31]
Minimum Profitability Threshold	Minimum net gain required after all costs.	⩾1% [31]
High-Performance Success Rate	Percentage of trades that are successfully executed and profitable.	85% - 95% [31] [32]

This landscape reveals that building a successful agent on Polygon is not merely about code; it is about navigating a complex economic and competitive environment. The agent must be architected to handle the nuances of multiple protocols, react with extreme

speed to fleeting opportunities, and manage its finances with precision to survive in a market where even minor inefficiencies can lead to losses. The low-latency, high-throughput nature of Polygon PoS, combined with its substantial TVL, makes it a prime target for such an advanced autonomous agent, provided it is equipped with the right tools and strategies for the challenges ahead.

Protocol Integration and Monitoring Strategy

To effectively hunt for liquidation opportunities across the Polygon ecosystem, the agent must employ a sophisticated and multi-layered monitoring strategy that combines broad-spectrum scanning of public data with targeted analysis of private, high-priority feeds. This dual-pronged approach is essential for maximizing detection speed and ensuring the agent can act before competitors do. The foundation of this strategy rests on accessing real-time on-chain data, which can be achieved through several channels. The most direct method is interacting with a full node via a JSON-RPC provider. Services like Infura, Alchemy, and OMNIA provide access to PolyScan-compatible block explorers, offering a gateway to the raw data flowing through the network [8] [23]. OMNIA, in particular, offers dedicated RPC endpoints for Polygon with features like privacy proxies and mixnets, which can enhance data retrieval efficiency [8]. By subscribing to WebSocket notifications for new blocks and pending transactions, the agent can scan the mempool for any calls directed to the core logic contracts of lending protocols like Aave, Granary, and Compound III [23]. This raw mempool scanning provides the widest possible coverage but comes with the inherent risks of frontrunning, as described previously [12].

A more structured and reliable approach involves leveraging subgraphs, which are GraphQL APIs that index and organize on-chain data for easy querying. The Aave protocol provides official, production-ready subgraphs for its Polygon V2 and V3 deployments [28]. These subgraphs expose a wealth of information, including historical and real-time events related to liquidations. Specifically, they index the `LiquidationCall` event emitted by the Aave Pool contract whenever a liquidation occurs [28] [29]. This event contains critical data points necessary for the agent to analyze the opportunity, such as the addresses of the collateral and debt assets, the amounts involved (`collateralAmount`, `principalAmount`), the user being liquidated, and the liquidator who executed the call [28] [29]. By running a dedicated consumer application that queries this subgraph, the agent can move beyond passive mempool scanning to actively poll for newly created liquidation opportunities. This method is significantly

faster and more reliable than waiting for a transaction to appear in a mined block. The agent can set up continuous queries to look for new `LiquidationCall` events, parse the data, enrich it with real-time token metadata and USD prices fetched from Aave's Oracle V3, and then calculate the potential profitability of the opportunity [29]. It is important to note that the data within subgraphs may require post-processing; for instance, fields like `totalLiquidity` are snapshots taken at the time of indexing, not query time, and interest rates are stored in RAY units (10^27), requiring conversion formulas to be meaningful [28]. Utilizing libraries like `aave-utilities` can help with this interpretation to ensure accuracy [28]. Developing a similar subgraph-based monitoring layer for other protocols like Granary Finance and Compound III would be a critical next step, as their event structures may differ from Aave's, requiring custom ABIs and indexing logic.

Once an opportunity is detected, the agent must quickly assess its viability and construct a valid execution plan. This requires not only monitoring but also a deep understanding of the specific contract interfaces of each lending protocol. For Aave v3 on Polygon, the main registry contract, `PoolAddressesProvider`, serves as the entry point to discover all other protocol addresses, including the current proxy address for the `Pool` contract itself [40]. Hardcoding contract addresses is a fragile practice, as governance proposals can update implementations over time; therefore, fetching the latest addresses programmatically is the recommended approach [40]. The main Aave v3 Pool contract on Polygon Mainnet has a well-documented address: `0xc4dCB5126a3AfEd129BC3668Ea19285A9f56D15D` [43]. This contract's interface exposes the `liquidateCall` function, which is the target of the agent's transaction. The function signature requires five inputs: the address of the user to be liquidated, the amount of debt to cover, the amount of collateral to seize, the address of the liquidator, and a flag indicating whether the liquidator wants to receive aTokens or the underlying assets [19]. The agent's Solidity smart contract will need to interact with this function to initiate the liquidation. For Granary Finance and Compound III, similar discovery mechanisms likely exist, involving registry or factory contracts that return the addresses of the active lending pools. The agent must maintain a registry of these addresses and ABIs for all monitored protocols to correctly format its transaction calls.

The table below outlines the key components for integrating with the primary lending protocols on Polygon, based on the available information.

Component	Aave v3 (Polygon)	Granary Finance (Polygon)	Compound III (Polygon)
Main Pool Contract Address	`0xc4dCB5126a3AfEd129BC3668Ea19285A9f56D15D` [43]	Information not available in provided sources	Deployed as cUSDCv3 market [39]
Event Signatures	`LiquidationCall` (Standardized ABI) [29] [56]	Implied to emit liquidation events	Emits `LiquidationCall`-like events [38]
Data Source	Official Subgraphs [28]	Not Available	Not Available
Oracle Provider(s)	Aave Oracle V3 [29]	Information not available in provided sources	Chainlink [38] [39]
Key Risk Factors	Dynamic risk parameters, Isolation Mode [41] [53]	Information not available in provided sources	Gradual liquidation mechanics, supply caps [38] [39]

For Granary Finance and Compound III, since detailed ABI and event information is not provided in the context, the development process would involve reverse-engineering the contracts on PolyScan to understand their internal logic and event emissions [23]. This is a standard procedure for interacting with any new smart contract on an EVM-compatible chain. The agent must be modular enough to load different ABIs and interpret different event structures for each protocol it supports. This flexibility is paramount for fulfilling the requirement to "monitor every defi lending platofrm on Polygon." The agent's monitoring module should be designed as a plugin-based system, where each plugin corresponds to a specific protocol, containing its own logic for discovering contract addresses, indexing events, and calculating profitability. This modular design ensures that adding support for a new lending protocol in the future is a manageable task of developing a new plugin rather than rewriting the entire monitoring engine. By combining the broad coverage of raw mempool scanning with the targeted, high-speed data access from subgraphs and direct contract interaction, the agent can build a comprehensive and resilient surveillance system capable of detecting and analyzing liquidation opportunities across the entire Polygon DeFi landscape with minimal latency.

Advanced Execution Infrastructure and MEV Mitigation

Executing a liquidation arbitrage transaction on Polygon is a race against time and competing agents, making the choice of execution infrastructure a critical determinant of success and profitability. The agent's core challenge is twofold: first, to construct a complex, atomic transaction that successfully covers debt, seizes discounted collateral,

swaps assets, and repays a flash loan within a single block; and second, to submit this transaction to the network in a way that minimizes exposure to frontrunning attacks, where competitors copy the agent's intent and pay higher gas fees to steal the opportunity [12]. The foundation of the execution engine is a self-contained Solidity smart contract that orchestrates the entire liquidation process. This contract's logic would follow a standardized sequence: upon receiving a request to liquidate a specific position, it would first call the Aave v3 Pool contract's `flashLoan` function, passing itself as the receiver and providing a callback function selector [17]. The flash loan contract would then transfer the required amount of debt tokens to the agent's contract. Inside the callback function, the agent's contract would immediately call the Aave Pool's `liquidateCall` function with the necessary parameters (user, debt asset, collateral asset, etc.) [19]. Upon successful execution of the liquidation, the contract would receive the discounted collateral. It would then proceed to swap this collateral for the desired profit asset (typically a stablecoin) using a DEX like Uniswap or QuickSwap [23]. Finally, it would repay the flash loan plus its fee and send any remaining profit to the owner's wallet [17]. This entire sequence must be contained within a single transaction to ensure atomicity, preventing failure states where part of the operation succeeds and another fails.

The submission of this atomic transaction is where the agent's infrastructure choices become paramount. Using a standard, public JSON-RPC endpoint means the transaction enters the public mempool, visible to anyone, including rival bots [1]. Frontrunning is almost guaranteed in a competitive environment, especially during periods of high volatility, leading to failed transactions and wasted gas fees [2]. The primary defense against this threat is the use of private transaction relays, which shield the transaction from the public mempool until it is ready to be included in a block [12]. Polygon supports several MEV mitigation technologies that can be integrated into the agent's workflow. One option is Blocknative's bundle RPC endpoint, which enables interaction with builders and supports Flashbots-style bundles [1]. Another is `mev-bor`, a fork of Polygon's Bor client that allows for MEV bundles, enhancing privacy and validation [5]. Perhaps the most potent native solution on Polygon is FastLane (PFL). As of late 2024, PFL operates across approximately 40% of the Polygon network and is designed specifically to prevent sandwich attacks by ensuring fair ordering of transactions [9]. Integrating with the PFL relay would provide a significant competitive advantage by giving the agent a more reliable path to execution without being subject to the highest bidder in the public auction. The agent's workflow would be modified to construct its transaction, sign it, and submit it directly to a private relay instead of broadcasting it to the public network. This shifts the agent from a reactive participant in a public auction to a proactive member of a private order flow, drastically improving its chances of success.

Beyond just submitting the transaction privately, the agent must optimize its bidding strategy to outcompete others. This is achieved through careful management of priority fees, which became a standard feature with the adoption of EIP-1559 [3]. The agent needs to estimate the optimal priority fee to include in its transaction to ensure timely inclusion in a block without overpaying. This estimation can be dynamic, adjusting based on network congestion and the perceived competitiveness of the opportunity. Some advanced bots use heuristic models to predict the necessary gas price, sometimes assuming a miner tip based on historical data, such as a 95% assumption mentioned in one case study [1]. A crucial step in this process is pre-execution simulation. Before committing to a transaction, the agent should simulate its intended actions on a testnet or via a provider's simulation endpoint. For example, the Polygon Liquidator bot uses Alchemy Transact's Asset Changes JSON RPC endpoint to simulate arbitrage transactions and check for profitability before submission [47]. This allows the agent to verify that the transaction will succeed and yield a positive return, avoiding costly failures. This simulation step is vital for managing risk and ensuring that the agent only attempts to execute profitable opportunities.

Finally, to free up capital and reduce operational friction, the agent can leverage meta-transactions. Meta-transactions decouple the account that intends to perform an action from the account that pays for the gas [4]. The agent's control wallet would sign a message expressing its intent to execute a specific function on the agent's smart contract. This signed message is then submitted to a relayer service, which wraps it into a standard Ethereum transaction, pays for the gas, and submits it to the contract. The contract, which inherits from a `NativeMetaTransactions` base contract, validates the signature and executes the original intent [4]. This pattern is natively supported on Polygon PoS. Services like Biconomy offer a managed meta-transaction API that can be easily integrated [4]. By using a meta-transaction relayer, the agent can operate effectively without needing to hold large reserves of MATIC for gas payments, allowing it to deploy its capital more aggressively towards finding and executing MEV opportunities. This combination of private bundling, EIP-1559 fee optimization, pre-execution simulation, and gasless operation via meta-transactions forms a robust and highly effective execution infrastructure, positioning the agent to compete successfully in the challenging and rewarding world of MEV on Polygon.

Implementing Self-Improvement Through Permanent Memory

The user's requirement for a "self improving" agent introduces a critical architectural consideration: the need for a durable, tamper-proof, and permanently accessible memory system. This system is the foundation upon which the agent's learning, strategy refinement, and historical performance records are built. Any strategy that relies on mutable or ephemeral storage is inherently fragile and cannot fulfill the promise of true self-improvement. The user expressed openness to exploring alternative storage strategies native to Polygon, which necessitates a comparative analysis of the available options to determine the most efficient and reliable solution for long-term use. The primary contenders are InterPlanetary File System (IPFS), Arweave, and various on-chain or Polygon-native off-chain storage solutions.

IPFS is a decentralized file system that allows files to be stored and retrieved based on their content identifier (CID). While it offers decentralization, IPFS has a significant drawback for long-term archival: it does not have a built-in economic incentive for nodes to store files indefinitely. Data permanence relies on third-party pinning services paying nodes to keep copies online. If a file is not pinned by someone, it can be removed from the network, making it unreliable for storing critical strategy versions or audit trails that must be immutable and always accessible [52]. This reliance on recurring fees and human coordination makes IPFS unsuitable for the agent's long-term memory requirements, despite its popularity in other web3 contexts.

Arweave, in contrast, is purpose-built for permanent storage with its "pay once, store forever" model [49]. It achieves this through a novel consensus mechanism called Proof of Access (PoA), where miners must prove they can retrieve random pieces of old data to mine new blocks, economically incentivizing them to retain the entire historical dataset [50]. All data is cryptographically verified and recorded on the blockweave, creating an immutable and censorship-resistant ledger of information [52]. Storage is paid for upfront with AR tokens, and the initial payment funds a perpetual endowment pool that earns yield to subsidize miner rewards for centuries [51]. This model guarantees data persistence for at least 200 years, providing a quantifiable and robust guarantee of permanence that is ideal for an agent's memory [49]. Every piece of data uploaded to Arweave is verifiably permanent, unlike IPFS, which makes it the superior choice for storing the agent's evolving strategy logic and performance history.

While Polygon itself offers some storage capabilities, they are generally inefficient for this use case. Writing large datasets directly onto the blockchain is prohibitively expensive in terms of gas fees. Storing pointers or hashes of data on-chain is feasible, but the actual data would need to be hosted off-chain. Polygon-native solutions like Verida DID demonstrate this hybrid approach: a DID document's pointer is written on-chain to Polygon, but the full document is stored off-chain on HTTP servers [14]. While Verida is designed for identity management, its architecture highlights a common pattern. However, the reliability of such systems depends entirely on the uptime and integrity of the off-chain storage providers. An attacker or a provider's server failure could render the data inaccessible. Arweave, by contrast, embeds the economic incentives for long-term storage directly into its protocol, eliminating the dependency on third-party pinning services and providing a much stronger guarantee of availability [52].

Given this analysis, the most efficient and robust architecture for the agent's self-improvement is a hybrid model that leverages the strengths of both Polygon and Arweave. This three-tiered approach combines on-chain verification with off-chain permanence. First, the agent's core logic and strategy files, which can be large, would be stored on Arweave. Each version of the strategy would be uploaded, resulting in a unique, immutable Transaction ID (TxID) that serves as its permanent hash. Second, a simple registry contract would be deployed on the Polygon network. This contract would act as an immutable ledger, storing the TxID of the currently active strategy version along with a timestamp and the signature of the entity that approved the change. Writing this lightweight metadata to Polygon is extremely cheap and fast. Third, the agent's runtime logic would be configured to fetch the latest strategy TxID from the Polygon registry contract. Upon startup or upon receiving a governance signal, the agent would retrieve the full strategy file from the Arweave gateway using the TxID. This architecture is optimal because it uses Polygon for rapid, low-cost, and verifiable on-chain interactions (updating the registry), while relying on Arweave for the guaranteed, permanent, and unchangeable storage of the actual strategy content. This directly addresses the user's need for a system that is both efficient and secure for the long term. The on-chain registry provides a canonical, auditable history of every strategy update, while Arweave ensures that the content of each strategy is preserved indefinitely, enabling a true cycle of learning, adaptation, and verification.

Core Agent Logic and Economic Viability Analysis

The heart of the autonomous agent lies in its core logic, which translates detected opportunities into profitable actions. This logic must be deterministic, gas-efficient, and robust enough to handle the complexities of the DeFi environment. The primary function of the agent is to execute a liquidation call atomically, a process that requires a series of precise on-chain interactions orchestrated by a dedicated smart contract. The sequence begins with the agent's control logic identifying a viable opportunity, typically by monitoring `LiquidationCall` events from a protocol like Aave v3 [28]. Once an opportunity is confirmed, the agent's smart contract is invoked with the necessary parameters. The first step in the contract's execution is to acquire the required capital via a flash loan from the same Aave v3 pool [17]. The flash loan is arranged by calling the pool's `flashLoan` function, specifying the agent's contract as the receiver. This allows the contract to borrow the exact amount of debt needed to liquidate the position without requiring any upfront capital, a crucial feature for maximizing capital efficiency [32]. The flash loan agreement stipulates that the borrowed amount, plus a small fee (typically around 0.09%), must be returned within the same transaction block [31].

With the borrowed debt tokens now residing in its own contract, the agent proceeds to the core action: triggering the liquidation. It calls the `liquidateCall` function on the Aave Pool contract, providing the addresses of the target user, the debt asset, and the collateral asset, along with the amounts to be liquidated [19]. The Aave protocol's `LiquidationLogic` library handles the rest of the process internally. It verifies that the target position is indeed undercollateralized, calculates the amount of collateral the liquidator is entitled to receive (which includes the 5-10% bonus), and transfers the discounted collateral to the agent's contract [19]. At this point, the agent holds the seized collateral but is still liable for the flash loan. The next step is to convert this collateral into a stable asset, such as USDC or DAI, to ensure a predictable profit. This is accomplished by routing a swap through a decentralized exchange (DEX) like Uniswap or QuickSwap [23]. The agent's contract interacts with the DEX's router contract to execute the swap, carefully managing slippage to avoid unfavorable price execution. After the swap is complete, the agent's contract now holds the stablecoins. It must then repay the flash loan, returning the original borrowed amount plus the 0.09% fee to the Aave pool [31]. The final step is to calculate the net profit—the difference between the value of the collateral seized (plus the bonus) and the sum of the debt covered, the flashloan fee, and any DEX swap fees—and transfer this remainder to the agent's owner wallet. This entire process, from initiating the flash loan to sending the final profit, must be completed within a single, atomic transaction to prevent any intermediate failure states.

The economic viability of this entire operation hinges on a meticulous cost-benefit analysis. The agent must calculate the potential profit before committing to the transaction. The formula for profitability is straightforward:

$$Profit = (CollateralSeized \times (1 + LiquidationBonus)) - (DebtCovered + FlashloanFee + SwapFee + Gas$$

Each component of this equation must be estimated accurately. The `LiquidationBonus` is known from the protocol's risk parameters [17]. The `Collateral_Seized` and `Debt_Covered` are provided by the `LiquidationCall` event data [28]. The `FlashloanFee` is a small percentage of the debt covered, typically 0.09% on Aave V3 [31]. The `SwapFee` is a function of the DEX's fee structure (e.g., 0.3% for Uniswap) and the size of the trade relative to the liquidity pool's depth [31]. The `GasFee` is the most volatile component and is particularly relevant on Polygon. While the base fee is low, the priority fee required to get the transaction included can vary significantly, especially during periods of high competition for MEV opportunities [2] [21]. The agent's logic must include a sophisticated gas estimator that can predict the necessary priority fee based on recent block data. If the calculated profit is negative, the agent must abort the transaction to avoid a loss. Even if the profit is positive, the agent must consider the probability of success. In a congested network, a transaction with insufficient priority fee may simply fail, costing the agent the gas spent on submission. Therefore, the agent's decision-making loop should incorporate a risk-adjusted profitability threshold, ensuring it only attempts transactions it has a high confidence of completing successfully.

Stress-testing and risk management are non-negotiable aspects of the agent's design. The DeFi environment is fraught with potential hazards. Oracle attacks, where an attacker manipulates a price feed to incorrectly mark a healthy position as undercollateralized, pose a significant threat [12]. To mitigate this, the agent's logic should prioritize decentralized oracles like Chainlink or Pyth and, where possible, use TWAP (Time-Weighted Average Price) calculations to smooth out short-term price spikes [12] [27]. Furthermore, the agent must be prepared for unexpected behavior from the lending protocols themselves. For example, Aave v3's introduction of Efficiency Modes (eMode) can alter the liquidation economics for groups of correlated assets, requiring the agent to query the protocol's state to get accurate parameters before acting [30]. The agent should also incorporate circuit breakers, which can pause its operations if certain conditions are met, such as a sharp drop in market value or an unusually high number of failed transactions, protecting it from catastrophic losses during extreme market events [31] [32]. By embedding these layers of financial modeling, risk assessment, and defensive

programming into its core logic, the agent can navigate the treacherous waters of the Polygon DeFi ecosystem and consistently turn liquidation opportunities into sustainable profits.

Synthesis and Strategic Recommendations

In conclusion, the transition of an autonomous agent from the Hedera network to the Polygon ecosystem represents a significant leap in complexity, shifting its role from a simple transaction initiator to a sophisticated, high-stakes MEV hunter. The successful implementation of such an agent is not merely a matter of porting code but requires a holistic architectural overhaul centered on three pillars: deep protocol integration, mastery of advanced execution infrastructure, and the implementation of a robust, permanent memory system for self-improvement. The analysis of the Polygon landscape reveals a vibrant but fiercely competitive environment where success is contingent on speed, precision, and strategic foresight. The agent must be architected to thrive in a world where microseconds can mean the difference between profit and loss, and where malicious actors constantly seek to undermine its operations.

The first pillar, **deep protocol integration**, is foundational. The agent must be capable of monitoring all major DeFi lending platforms on Polygon, with Aave v3 serving as the primary target due to its leadership in TVL [22]. This requires a multi-faceted monitoring strategy that combines broad mempool scanning with targeted analysis of protocol-specific subgraphs and direct contract interactions [23] [28]. By leveraging the standardized `LiquidationCall` event emitted by Aave v3, the agent can reliably detect opportunities across all its network deployments [56]. To fully meet the requirement of monitoring "every defi lending platofrm," the agent's modular design must be extended to support Granary Finance and Compound III, which, while less documented, are active participants in the Polygon ecosystem [36] [37].

The second pillar, **advanced execution infrastructure**, is the agent's lifeline in the competitive MEV market. The single most critical recommendation is the immediate integration of private transaction relays. The agent must abandon public broadcast in favor of submitting transactions through private channels to mitigate the pervasive threat of frontrunning [12]. Among the available options on Polygon, FastLane (PFL) stands out as the dominant and most native solution, offering protection against sandwich attacks and a direct line to validators [9]. Supplementing this with support for Flashbots-style

bundles via `mev-bor` would further solidify its execution capabilities [5]. This infrastructure must be complemented by rigorous EIP-1559 fee optimization, pre-execution transaction simulation to validate profitability, and the use of meta-transactions to enable gasless operation, freeing capital for more aggressive trading [3] [4] [47].

The third and final pillar, **permanent memory for self-improvement**, addresses the agent's long-term viability. The choice of storage technology is critical. While IPFS offers decentralization, its lack of a built-in economic incentive for permanence makes it unsuitable for archival purposes [52]. Arweave, with its "pay once, store forever" model backed by cryptographic proof and an endowment pool, emerges as the unequivocal best choice for storing the agent's immutable strategy versions and performance history [49] [50]. The most efficient and robust architecture is a hybrid model: a simple, on-chain registry contract on Polygon to store the Arweave TxIDs of active strategies, combined with Arweave for the actual, permanent storage of the strategy files. This approach leverages the speed and low cost of Polygon for on-chain governance while guaranteeing the immutability and permanence of Arweave for the agent's core intellectual property.

To summarize, the following actionable recommendations are proposed to guide the project: 1. **Prioritize Private Relay Integration:** Immediately develop and integrate with FastLane (PFL) as the primary channel for submitting private orders. Add support for Flashbots bundles via `mev-bor` for enhanced privacy and capability. 2. **Develop Multi-Protocol Monitoring:** Build a dedicated subgraph consumer for Aave v3 on Polygon to supplement raw mempool scanning. Extend this modular architecture to add plugins for Granary Finance and Compound III to ensure comprehensive coverage of the liquidation landscape. 3. **Implement the Arweave-Polygon Hybrid Memory Model:** Use Polygon for an on-chain registry of strategy version TxIDs and Arweave for the permanent storage of strategy files. This provides the best combination of low-cost on-chain verification and unbreakable off-chain persistence. 4. **Build Robust Risk Management:** Incorporate pre-execution profitability calculations, dynamic gas fee estimators, and circuit breakers to halt operations during extreme market volatility. Ensure the agent's logic is resilient to oracle manipulation by prioritizing decentralized price feeds. 5. **Utilize Meta-Transactions:** Integrate a meta-transaction relayer like Biconomy to allow the agent to operate without holding large reserves of MATIC for gas, optimizing its capital allocation for MEV hunting.

By adhering to this blueprint, the agent can be transformed from a basic tool into a formidable and adaptive force within the Polygon ecosystem, capable of systematically

identifying, evaluating, and exploiting liquidation arbitrage opportunities with a high degree of efficiency and resilience.

Reference

1. MEV Bot Guide: Create an Ethereum Arbitrage Trading Bot https://www.blocknative.com/blog/mev-and-creating-a-basic-arbitrage-bot-on-ethereum-mainnet
2. What is MEV in Crypto: Its Protection and Automation with Bot https://tatum.io/blog/what-is-mev-in-crypto
3. MEV Bot Development: Types, Features and Benefits https://ideausher.com/blog/mev-bot-development/
4. Meta transactions https://docs.polygon.technology/pos/concepts/transactions/meta-transactions/
5. Introducing MEV support for Polygon - Blog https://blog.marlin.org/introducing-mev-support-for-polygon
6. Running a Relayer | Across Documentation https://docs.across.to/relayers/running-a-relayer
7. NethermindEth/polygon-builder: Flashbots MEV-Boost ... https://github.com/NethermindEth/polygon-builder
8. Private and Secure Polygon Endpoints https://omniatech.io/pages/polygon-rpc/
9. MEV - FastLane Update - Polygon Mainnet (PoS) https://forum.polygon.technology/t/mev-fastlane-update/13299
10. A Deep Dive on MEV Market Structure https://portal.vc/mev
11. What is Shared Sequencer? Shared ordering for rollups & ... https://www.cube.exchange/what-is/shared-sequencer
12. 30+ DeFi Attack Vectors & How to Secure Your Assets in ... https://www.quillaudits.com/blog/web3-security/defi-attack-vectors-security-risks
13. What is Maximal Extractable Value (MEV)? https://www.cointracker.io/learn/maximal-extractable-value
14. Introducing “did:vda”: Fast, Cheap Web3 Identity on Polygon https://news.verida.network/introducing-did-vda-a-fast-cheap-web3-identity-solution-on-polygon-5d1487941477

15. A Digital Identity Blockchain Ecosystem: Linking ... https://www.mdpi.com/2076-3417/15/15/8577
16. Setup App Chain zk Rollup w/ Polygon zkEVM & Metamask https://medium.com/coinmonks/how-to-setup-your-own-self-sovereign-app-chain-zk-roll-up-using-polygon-zkevm-and-connect-with-f50b13231123
17. Performing Liquidations on the Aave DeFi Lending Protocol https://blog.amberdata.io/performing-liquidations-on-the-aave-defi-lending-protocol
18. How to Yield Farm with Aave on Polygon https://polygon.technology/blog/how-to-yield-farm-with-aave-on-polygon
19. DeFi: Full Guide [Part II] - Lucas Martin Calderon - Medium https://lucasmartincalderon.medium.com/defi-full-guide-part-ii-9805683973b2
20. Polygon (POL): A comprehensive overview of an ... https://oakresearch.io/en/reports/protocols/polygon-pol-comprehensive-overview-ecosystem-ethereum-scaling-solutions
21. Polygon Ecosystem Overview https://messari.io/report/polygon-ecosystem-overview
22. Polygon: A 2024 Ecosystem Overview https://stakin.com/blog/polygon-a-2024-ecosystem-overview
23. Ultimate Polygon Development Guide 2024 https://www.rapidinnovation.io/post/polygon-blockchain-development-guide-ultimate-resource-for-developers-and-businesses
24. Polygon Review 2025: Updates You Don't Want to Miss! https://coinbureau.com/education/polygon-review/
25. List of 17 Decentralized Lending Dapps on Polygon https://www.alchemy.com/dapps/list-of/decentralized-lending-dapps-on-polygon
26. Lending Protocols on Polygon https://defillama.com/protocols/Lending/Polygon
27. What Are the Top 10 DeFi Lending Protocols to Watch in ... https://bingx.com/en/learn/article/what-are-the-top-defi-lending-protocols-to-watch
28. The code of Aave protocol subgraphs https://github.com/aave/protocol-subgraphs
29. Aave V3 Liquidation Tracker | Quicknode Sample App Library https://www.quicknode.com/sample-app-library/ethereum-aave-liquidation-tracker
30. Advanced Features | Aave Protocol Documentation https://aave.com/docs/aave-v3/markets/advanced
31. sorasuzukidev/ethereum-bnb-mev-bot https://github.com/sorasuzukidev/ethereum-bnb-mev-bot
32. butter991011/ethereum-mev-bot https://github.com/butter991011/ethereum-mev-bot

33. aave-protocol · GitHub Topics https://github.com/topics/aave-protocol?o=desc&s=forks
34. Gauntlet: Simulation-Based Risk Modeling and ... https://medium.com/@gwrx2005/gauntlet-simulation-based-risk-modeling-and-quantitative-research-for-defi-fb736e4392d2
35. Granary v1 — A Retrospective https://medium.com/@granarydefi/granary-v1-a-retrospective-5cbf05273f8a
36. Early Users Rejoice! | lending protocol Granary Finance https://cryptorank.io/news/feed/37cf4-205222-granary-finance-lending-protocol-launched-early
37. Chainrisk & Compound Finance Partnership Announcement https://chainrisk.xyz/blog-posts/chainrisk-compound-finance-partnership-announcement
38. Compound III (Comet) Explained: Multi-Chain DeFi Lending https://levex.com/en/blog/compound-iii-comet-multi-chain-defi-lending
39. Initialize Compound III (USDC on Polygon PoS) https://www.comp.xyz/t/initialize-compound-iii-usdc-on-polygon-pos/3611/11
40. Pool Addresses Provider | Aave Protocol Documentation https://aave.com/docs/aave-v3/smart-contracts/pool-addresses-provider
41. Introducing Aave V3 - Governance https://governance.aave.com/t/introducing-aave-v3/6035
42. [ARFC] stMATIC & MaticX Emission_Admin for Polygon v3 ... https://governance.aave.com/t/arfc-stmatic-maticx-emission-admin-for-polygon-v3-liquidity-pool/10632
43. How to deposit and redeem AAVE V3 in Solidity (Goerli)? https://governance.aave.com/t/how-to-deposit-and-redeem-aave-v3-in-solidity-goerli/9687
44. BGD. Aave v3 cross-chain listing template - Development https://governance.aave.com/t/bgd-aave-v3-cross-chain-listing-template/9354
45. butter991011 https://github.com/butter991011
46. BSC/ETH MEV Bot https://github.com/marksantiago290/Ethereum-MEV-BOT
47. gonzalolater/Compound-Liquidation-Bot https://github.com/gonzalolater/Compound-Liquidation-Bot
48. How to Make a Liquidation Bot https://www.ratherlabs.com/blog/how-to-make-a-liquidation-bot
49. Arweave (AR) - The Permanent Data Storage Protocol https://www.bitcoinnoobs.com/crypto/ar
50. Arweave (AR): Permanent Storage, Permaweb, and ... https://medium.com/thecapital/arweave-ar-permanent-storage-permaweb-and-profiting-from-its-volatility-with-options-791156dbb201

51. Protocol https://docs.arweave.org/developers/development/protocol
52. Why Arweave's Permanent Storage Model Wins https://ar.io/articles/why-arweaves-permanent-storage-model-wins
53. Reserve | Aave Protocol Documentation https://aave.com/docs/aave-v3/concepts/reserve
54. Changelog | Aave Protocol Documentation https://aave.com/docs/resources/changelog
55. Governance | Aave Protocol Documentation https://aave.com/docs/ecosystem/governance
56. BGD. Aave v3 Ethereum. New deployment vs Aave v2 ... https://governance.aave.com/t/bgd-aave-v3-ethereum-new-deployment-vs-aave-v2-upgrade/9990