Design and Implementation of an Event-Driven Systematic Market-Making Architecture

Thai Koehnle
Cornell University

February 2026

Overview

This document outlines a systematic market-making methodology optimized for the Gemini Prediction Markets over an 8-week tournament horizon. Initial iterations of the architecture deployed a directional, statistical arbitrage strategy; however, empirical analysis of the platform's liquidity profile revealed prohibitive bid-ask spreads that eroded theoretical edge for market takers. Consequently, the system was refactored into a passive, zero-knowledge liquidity provision engine. By utilizing cross-exchange order books (Kalshi) and continuous options pricing models (Black-Scholes) as absolute pricing anchors, the bot calculates a vig-free fair value. It then deploys an Avellaneda-Stoikov inventory skewing model to dynamically manage risk without relying on public order book data. Concurrently, an integrated spot-trading algorithm neutralizes directional crypto exposure via vectorized delta hedging, generating organic volume while strictly constraining tail risk.

1 Introduction & Market Microstructure

The Gemini Prediction Market currently exhibits the classic characteristics of an immature, retail-driven exchange: low resting liquidity, wide bid-ask spreads, and significant pricing inefficiencies.

Initially, this architecture was designed as a directional taker. The system scraped highly liquid "sharp" oracles to identify positive expected value (+EV) discrepancies on Gemini, executing latency-arbitrage trades sized via a Dynamic Fractional Kelly scaler. However, live deployment revealed a critical structural flaw: the bid-ask spreads on Gemini were consistently wider than the mathematical edge extracted from the oracles. Crossing the spread as a market taker subjected the portfolio to severe friction, effectively decaying the compounded Expected Value to sub-zero levels.

To survive the constraints of a fixed-bankroll tournament, the architecture was pivoted. Rather than paying the spread, the bot now captures it by acting exclusively as a passive market maker, dictating prices to retail flow while mathematically isolating itself from adverse selection.

2 The Oracle Engine: Fair Value Anchoring

Because Gemini does not expose a public limit order book (Level 2 data) for its prediction markets via API, the bot cannot "penny" existing quotes. Instead, it operates a zero-knowledge quoting model, relying entirely on external, high-fidelity oracles to establish an absolute baseline probability (p_{true}).

2.1 Discrete Events (Sports)

For mutually exclusive binary events (e.g., NFL, NBA), the bot queries the Kalshi limit order book. To extract the sharpest reflection of market consensus, the engine isolates the *yes_ask* and

no_ask. Because market makers on Kalshi build a spread (vig) into these quotes, the raw implied probabilities inherently exceed 1.00. The engine normalizes these values to strip the platform vig and isolate the true probability:

$$p_{\text{true}} = \frac{\pi_{yes\_ask}}{\pi_{yes\_ask} + \pi_{no\_ask}} \tag{1}$$

2.2 Continuous Events (Crypto)

For crypto price prediction markets, the bot models the outcomes as Cash-or-Nothing European Options, achieving $O(1)$ pricing latency. The engine continuously maps the Gemini strike and expiration to the nearest Deribit options instrument, extracting the live index price (S) and the market maker implied volatility (σ).

$$p_{\text{true}} = \mathcal{N}\left(\frac{\ln(S/K) + (r - \frac{\sigma^2}{2})t}{\sigma\sqrt{t}}\right) \tag{2}$$

3 Inventory Management & Pricing Execution

In a passive market-making regime, directional edge and Kelly sizing are obsolete. The primary risk is no longer event outcome, but rather extreme inventory accumulation (adverse selection). To manage this, the system implements a modified Avellaneda-Stoikov pricing model.

3.1 Avellaneda-Stoikov Inventory Skew

The bot maintains a continuous state ledger of its net inventory (q) for each market, where accumulating "YES" contracts results in $q > 0$. Instead of quoting symmetrically around the oracle's p_{true}, the engine calculates a dynamically shifted Reservation Price (p_{res}) based on a predefined risk aversion parameter (γ) :

$$p_{\text{res}} = p_{\text{true}} - (q \cdot \gamma) \tag{3}$$

As inventory accumulates on one side of the book, the mathematical skew violently shifts the p_{res} away from the oracle value. If the bot absorbs excessive "YES" contracts, p_{res} decreases, subsequently lowering the Ask price to incentivize retail buyers to take the inventory, while lowering the Bid price to prevent further accumulation.

3.2 Synthetic Spread & Execution Routing

With p_{res} established, the bot generates optimal, dual-sided limit quotes using a target half-spread (δ) :

$$P_{\text{bid}} = \max(0.01, p_{\text{res}} - \delta) \tag{4}$$

$$P_{\text{sell}} = \min(0.99, p_{\text{res}} + \delta) \tag{5}$$

Operating in a high-frequency cancel-and-replace loop, the bot posts these standard-lot quotes to the Gemini API. If the net inventory reaches the absolute maximum threshold $(|q| \geq Q_{\text{max}})$, the engine enacts a structural firewall, halting limit orders on the overexposed side until the inventory is naturally unwound by the market.

4 Spot Market Synergy & Risk Neutralization

While the Avellaneda-Stoikov skew protects against short-term toxic flow, holding prediction market inventory introduces systemic beta exposure to the underlying crypto assets. To optimize the tournament's volume-weighted scoring metric, a secondary spot-trading algorithm acts as a dedicated delta-hedging risk desk.

4.1 Vectorized Portfolio Delta Hedging

Upon receiving a fill confirmation on a crypto prediction contract, the *RiskStateManager* evaluates the localized Black-Scholes probability density function (PDF) to extract the contract's Greek Delta (Δ).

$$\Delta_{\text{YES}} = \frac{n(d_2)}{S_i \sigma_i \sqrt{t}} \tag{6}$$

These deltas are aggregated into an orthogonal risk vector $(\mathbf{\Delta})$, grouped strictly by the underlying ticker i:

$$\mathbf{\Delta} = \left[\sum \Delta_1, \sum \Delta_2, ..., \sum \Delta_n \right] \tag{7}$$

4.2 Cross-Hedging and the Beta (β) Fallback

The spot bot continuously polls the $\mathbf{\Delta}$ vector. If $\sum \Delta_i < 0$, it buys the equivalent spot asset. If $\sum \Delta_i > 0$, it sells from a dynamically initialized "Base Inventory."

Due to the "buy-only" constraint of the competition's spot market, Base Inventories may become depleted. When unable to directly sell spot asset i to hedge a positive delta, the engine falls back to a Beta-weighted cross-hedge using a proxy asset j. The optimal required delta in the proxy asset is defined as:

$$\Delta_j^* = \beta_{i,j} \cdot \left(\frac{S_i}{S_j} \right) \cdot \sum \Delta_i \tag{8}$$

5 Conclusion

By pivoting from a directional, latency-arbitrage taker to a passive, zero-knowledge liquidity provider, this architecture fundamentally alters its interaction with exchange friction. Synthesizing discrete Kalshi oracle scraping, continuous options pricing, and Avellaneda-Stoikov inventory skewing allows the engine to strictly control its risk profile without reliance on public Level 2 data. Concurrently, the integrated spot execution engine provides mathematically rigorous beta neutralization, organically generating the required volume to optimize the tournament's Volume $\times (1 + \text{ROE})^2$ scoring function.