Methodology and Framework

1. Data Pipeline

What We Are Trying
A Smart Order Router (SOR) relies on accurate, high-frequency market data to make informed decisions. The data pipeline ensures this data is processed, cleaned, and ready for use in backtesting simulations. This includes market data (bid/ask prices, trades) and venue-specific order book details.
Key Considerations

- **Data Sourcing**:
    - **Historical Market Data**: Sources like Alpha Vantage or IEX Cloud can provide historical price and volume data.
    - **Synthetic Data**: In the absence of real data, we generate synthetic market data resembling real-world scenarios.
- **Data Challenges**:
    - **Missing Data**: Gaps in price or volume data due to system failures or API issues can impact simulations. Solutions include interpolation or carrying forward the last known value.
    - **Synchronization**: Bid/ask updates and trade execution logs may arrive out of sync. Use event timestamps to realign them.

Implementation Approach

- Design an ETL (Extract, Transform, Load) pipeline to handle raw market data.
- Normalize all inputs (e.g., scaling prices, volumes).
- Generate synthetic data for testing scalability and robustness.

2. Execution Strategies

What We Are Trying
Execution strategies dictate how orders are broken up and executed in the market. The aim is to minimize costs and slippage while maximizing fill rates.
Simplified Execution Strategies

- **TWAP (Time-Weighted Average Price)**:
    - Divide the total order into equal parts executed at regular time intervals.
    - Suitable for stable markets with minimal volatility.
    - **Example**: If an order of 1,000 shares needs execution over 1 hour, execute 250 shares every 15 minutes.
- **VWAP (Volume-Weighted Average Price)**:
    - Trades are allocated proportionally based on historical or real-time market volume.

    - Adaptive to market liquidity changes but computationally heavier.
    - **Example**: If 30% of the day's trading volume happens in the first hour, execute 30% of the order in that hour.

Market Conditions

- **Volatility**:
    - High volatility increases execution risk and slippage.
    - VWAP performs better in such scenarios as it adapts to volume spikes.
- **Liquidity**:
    - TWAP assumes constant liquidity, making it less effective in illiquid markets.

3. Performance Metrics

What We Are Trying

Performance metrics evaluate the success of execution strategies and their impact on the order's final outcome.

Key Metrics

- **Execution Cost**:
    - The difference between the executed price and a benchmark price (e.g., VWAP).
    - **Example**: If the executed price is $101 and VWAP is $100, the execution cost is $1.
- **Slippage**:
    - The difference between the expected and actual execution price.
    - **Example**: If the expected execution price was $99 but the actual was $101, the slippage is $2.
- **Fill Rate**:
    - Percentage of orders successfully executed versus total orders placed.
    - **Example**: If 950 out of 1,000 shares are filled, the fill rate is 95%.

Implementation Approach

Track metrics in real-time during simulations and output them as part of the post-simulation analysis.

4. Simulation Logic

What We Are Trying

Simulate a realistic market environment to test the performance of SOR strategies under controlled conditions.

Key Components

- **Multi-Venue Routing**:
    - Orders are routed to multiple trading venues based on latency, liquidity, and fees.

    - **Example**: If Venue A has lower fees but higher latency compared to Venue B, split the order between them based on trade-offs.

- **Order Placement**:
    - Orders can be market orders, limit orders, or complex multi-leg orders.
    - **Example**: A multi-leg trade might involve selling stock in Venue A while simultaneously buying in Venue B to hedge.
- **Order Modifications**:
    - Simulate the impact of canceling or modifying orders due to market conditions or partial fills.

Implementation Approach

Use an event-driven simulation engine to mimic market conditions and track the lifecycle of orders.

5. Extensibility and Scalability

What We Are Trying

Design a framework that can adapt to new strategies, handle increased data volumes, and support various asset classes.

Extensibility

- Add support for:
    - **Advanced Strategies**:
        - Machine learning (e.g., reinforcement learning for smart routing).
        - Strategies like dynamic VWAP or adaptive TWAP.
    - **New Asset Classes**:
        - Expand to derivatives, bonds, cryptocurrencies, and forex.

Scalability

- **High-Performance Computing**:
    - Parallel processing for large-scale simulations.
    - Cloud-based architectures for distributed simulations.
- **Data Volume**:
    - Efficient data structures and storage solutions (e.g., Apache Parquet for market data).

Implementation Approach

- Modularize the codebase to allow easy integration of new components.
- Use scalable frameworks like Ray for distributed simulations.

This detailed framework outlines the end-to-end design of a backtesting system for SOR. Each component aligns with real-world trading challenges and is scalable for future enhancements.