Continuous Prompts: LLM-Augmented Pipeline Processing over Unstructured Streams

Shu Chen
Brown University

Deepti Raghavan
Brown University

Ugur Cetintemel
Brown University

"How can we express continuous, stateful LLM computations over unstructured streams?"

1 Motivation

Monitoring **unstructured streams** (e.g. clinical notes, financial news) increasingly requires **persistent**, **semantic-aware** computation

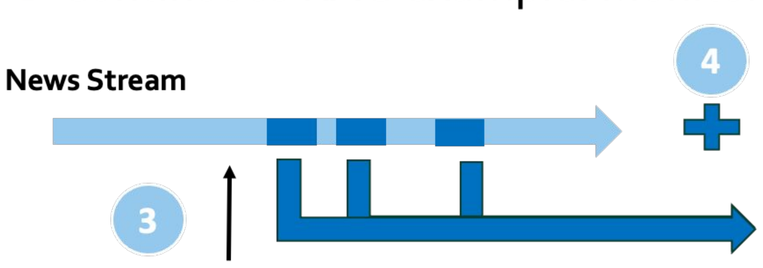
2 Goal

Elevate LLM reasoning from **one-shot, stateless** execution to **continuous, stateful** stream processing.

3 Contributions

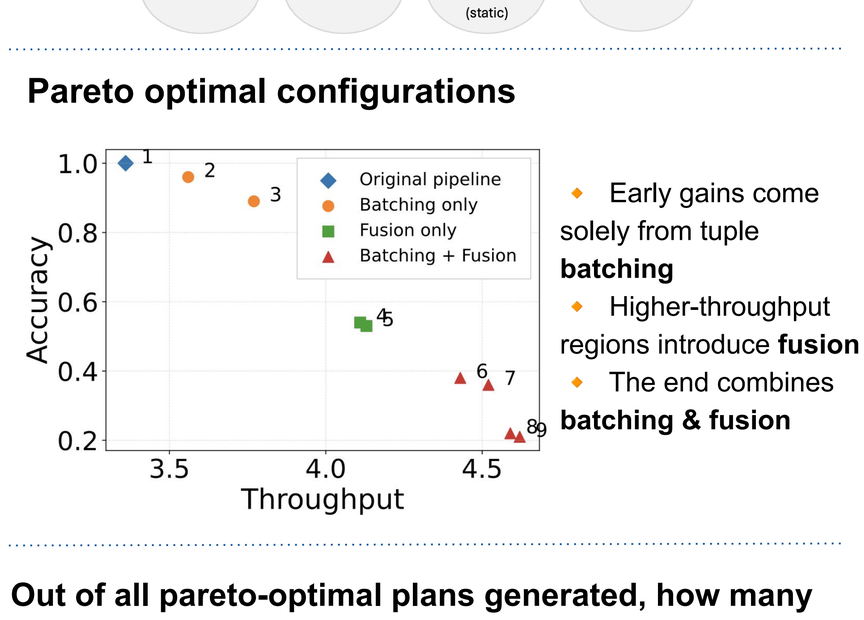
- **Continuous Prompts Framework**: stateful, streaming-native LLM operators
- **LLM-specific Optimizations**: batching & fusion
- **Dynamic planning** to adapt throughput under high arrival rates
- **Cost-aware MOBO** (Multi-objective Bayesian Optimization) for efficient frontier learning and plan selection

Continuous RAG example: Monitoring Stock Portfolio



4 Streaming Semantic Ops

- **Semantic Windows**: Maintains context-aware windows that evolve based on context drift
- **Semantic Group-By**: Dynamically groups stream elements by semantic attributes inferred by LLMs
- **Continuous RAG**: Continuously retrieves relevant data from input streams based on their relevance to a continuous prompt



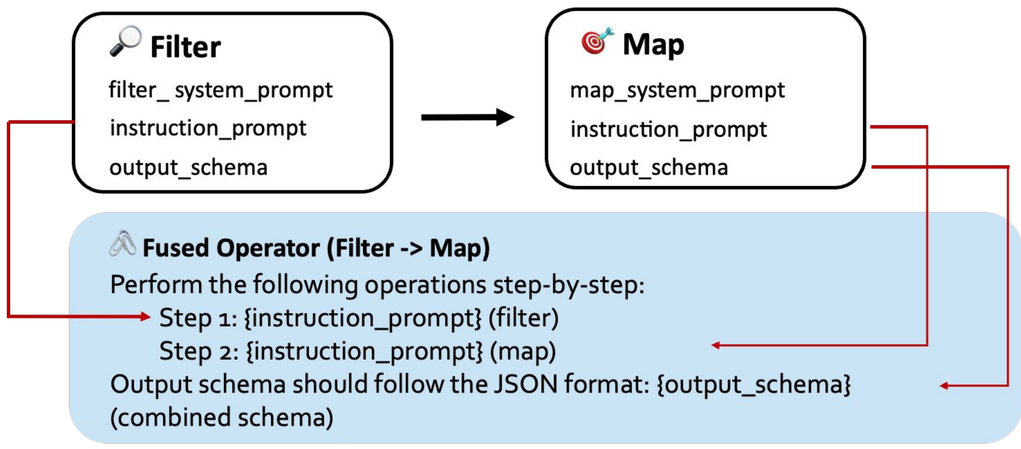
5 LLM-centric Optimizations

Tuple Batching

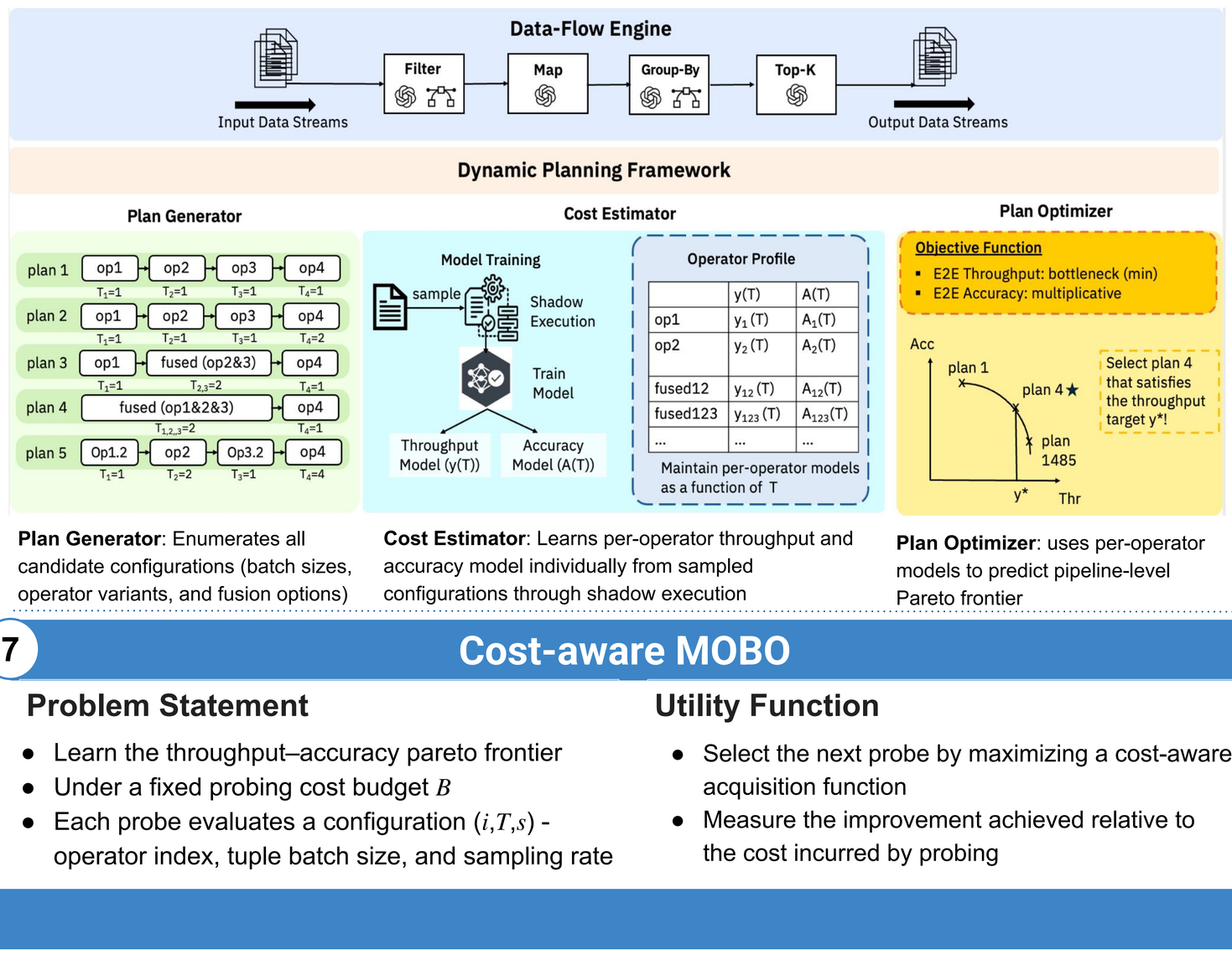
Process multiple input tuples within **one single prompt** to reduce **# of LLM calls**

- Shared Prefix
- Tuple Enumeration
- Output Specification

Operator Fusion



6 Dynamic Planning Framework



Plan Generator: Enumerates all candidate configurations (batch sizes, operator variants, and fusion options)

Cost Estimator: Learns per-operator throughput and accuracy model individually from sampled configurations through shadow execution

Plan Optimizer: uses per-operator models to predict pipeline-level Pareto frontier

7 Cost-aware MOBO

Problem Statement

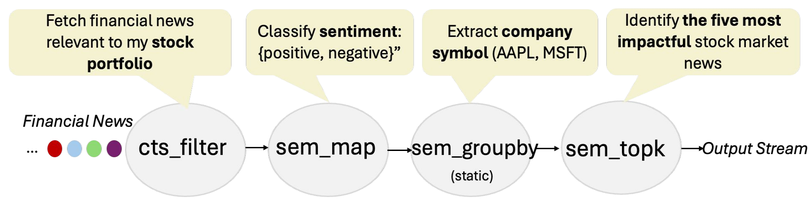
- Learn the throughput–accuracy pareto frontier
- Under a fixed probing cost budget B
- Each probe evaluates a configuration (i,T,s) - operator index, tuple batch size, and sampling rate

Utility Function

- Select the next probe by maximizing a cost-aware acquisition function
- Measure the improvement achieved relative to the cost incurred by probing

8 Results

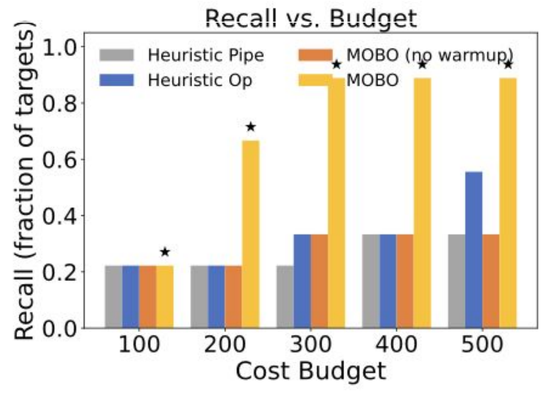
Use Case



Pareto optimal configurations

- Early gains come solely from tuple **batching**
- Higher-throughput regions introduce **fusion**
- The end combines **batching & fusion**

Out of all pareto-optimal plans generated, how many of them are actual pareto-optimal plans?



- MOBO consistently achieves higher recall across all budgets, saturating after B=300