Beam Summit 2026

Danny McCormick

Agentic Workflows in Beam: The Opportunity Ahead of Us

Deploying autonomous, reliable, and production-ready AI agent architectures inside scalable, streaming, and distributed data pipelines.

The Shift to Agentic Workflows

From Chatbots to Agents

Traditional LLM interactions are passive and instruction-response bound. Modern AI architectures are moving rapidly towards autonomous agents that reason, plan, execute multi-step tools, and interact dynamically with systems.

The Production Mandate

To scale agentic prototypes out of sandboxes, developers need frameworks that unify deterministic logic with generative models. Production-grade systems demand low latency, stateful memory, and cost controls.

Defining Agentic Workflows

An agentic workflow structures model execution through recurring loops of planning, memory retention, and tool orchestration.

Workflow Orchestration

Rather than single-turn prompts, workflows direct execution logic through state-dependent steps, evaluation loops, and feedback mechanism paths.

Tool Integration Blocks

Enables the orchestrator to query indexes, fetch real-time data elements, and execute programmatic actions through external system APIs.

Challenge 1: The Scaling Cost Trap

Raw Ingestion Volume = API Cost Exhaustion

Executing an LLM inference call or triggering an agentic runtime directly for every incoming message creates a direct linear dependency. Without middleware optimization, your daily API budget is bound tightly to your event stream volume.

Linear Cost Growth

When every message triggers a multi-step agent execution, the daily compute bill scales linearly with data volume.

Without high-performance batching, caching, and strategic pre-filtering, model costs will easily overwhelm any product's economic viability.

Challenge 2: Context Fragmentation

The Real-Time Context Gap

Agents require rich user sessions, historical context, and global state to act correctly. However, streaming events arrive naked, with none of this data attached.

If an agent must sequentially call slow external databases to fetch history for each run, performance crashes and cost climbs.

> Unprepared state and missing history make intelligent reasoning slow and expensive.

Assembling History

Tracking and sorting out-of-order session histories across highly distributed clusters is a massive infrastructure hurdle.

Context Bloat & Latency

Loading massive contexts repeatedly without optimization wastes active memory and spikes prompt token costs.

Data Pipelines: Scalability by Design

Modern data pipeline systems naturally master the orchestration of massive parallel workloads across workers.

Elastic Scale and Resource Management
Data pipelines inherently handle worker scaling, auto-provisioning, and queue backpressure seamlessly under peak loads.

Asynchronous & Batched Execution
Rather than sending records one-by-one, streaming runtimes organize tasks to parallelize API requests efficiently.

Built for the Heaviest Workloads

By deploying agents deep inside a distributed execution paradigm, we inherit battle-tested clustering mechanisms that keep high-throughput applications running reliably.

Data Pipelines: Cheap Data Prequalification

Optimize Token Budgets

Executing cheap rule-based data filtering and processing early in a pipeline instantly blocks invalid, duplicate, or empty payloads from consuming costly model cycles.

Stop wasting expensive, rate-limited tokens on unvetted, messy stream payloads.

Prequalification Step

Use deterministic expressions to validate structures and discard garbage rows cheaply in JVM/Go environments before they reach agents.

Cost-to-Value Optimization

Ensures expensive LLM inference triggers only for records that absolutely require deep semantic processing or dynamic tool reasoning.

Data Pipelines: Hydrating Rich Context

An agent is only as capable as the prompt context provided at runtime.

Stateful Stream Enrichment
Hydrate incoming event data with historical states and user sessions using stateful joins and side inputs directly inside the pipeline.

Pre-Assembled Context Packets
Instead of letting the agent make 5 separate slow API database lookups, the pipeline pre-loads the context packet, making model calls single-hop.

Pre-Hydration Architecture

Moving hydration from the Agent loop into high-performance stream ingestion drops latencies significantly, organizing and packing parameters before model execution.

Data Pipelines: Operational Guardrails

A containerized pipeline runner serves as an unbreachable sandbox for non-deterministic code.

PII Masking & Data Redaction
Enforce strict enterprise data governance standards by stripping sensitive fields inside secure runners before prompting external LLMs.

Strict Rate-Limiting & Throttle Handlers
Centralized traffic policing prevents runaway agent loops from overwhelming backends or crashing API token budgets.

Beam's Foundation for AI Agents

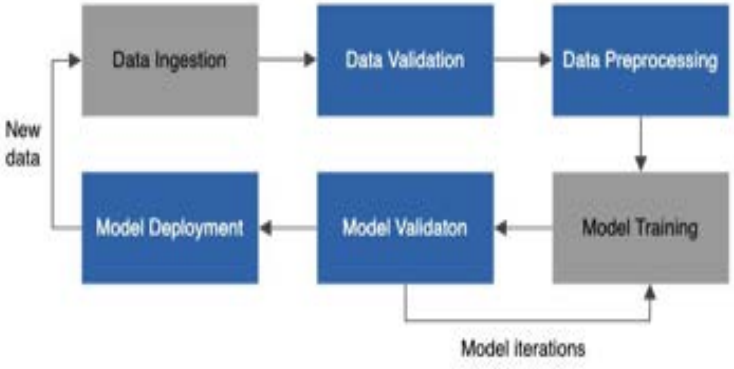
Apache Beam provides a native, low-latency pathway to execute model inference and agentic orchestration at scale.

RunInference Infrastructure

Leverages Beam's production-tested ML inference framework to cleanly host models, manage workers, and handle batch optimizations automatically.

Extensible Model Handlers

Features specific orchestration wrappers like AdkAgentModelHandler as a blueprint to run code-first frameworks within pipelines.



Beam RunInference: Providing batching, resource reuse, and thread-safe distributed orchestration for AI workflows.

Live Demonstration

Demo

Walking through AdkAgentModelHandler pipeline execution.

The Gaps: Areas for Contribution

EASY LEVEL

Framework Integrations

Extend Apache Beam ModelHandlers to natively encapsulate other popular agent frameworks such as LangChain, CrewAI, AutoGen, and LangGraph.

MEDIUM LEVEL

Deep SDK Adaptation

Enhance external agent SDKs to understand and natively scale tool executions over distributed Beam pipeline nodes directly.

HARD LEVEL

Stateful Iterative Loops

Develop native back-edge stream processing techniques in Beam to dynamically support feedback cycles and loop state without pipeline blockages.

Call to Action: Scale Smarter

1. Contribute to Beam ML

Help shape the future of AI/ML on Apache Beam. Join the community to implement next-gen ModelHandlers, expand multi-agent runtime support, or tackle the hard challenge of streaming execution loops.

2. Accelerate Your Business

Stop sending unvetted data straight to expensive LLMs. Refactor your production architectures to prequalify, filter, and enrich context within a high-throughput, cheap preprocessing tier first.

Questions & Answers

Agentic Workflows in Beam: The Opportunity Ahead of Us

Danny McCormick • Beam Summit 2026 | github.com/damccorm