arize

Arize AI - Agent Mastery Course

arize

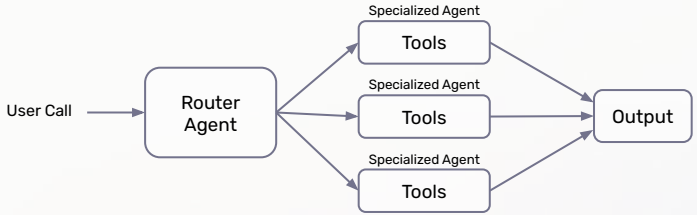
Module 3: Agent Architecture & Frameworks

Agent Architectures

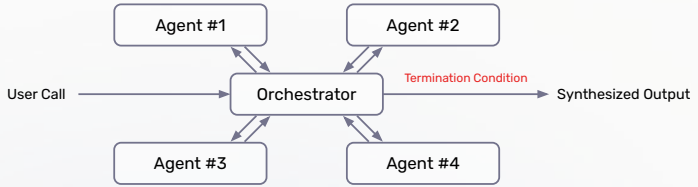
- Foundational patterns that shape how AI agents handle:
  - Reasoning: how agents plan and decide
  - Action: how agents use tools, APIs, and environment
  - Interaction: how agents manage conversations, memory, and feedback

Common Agent Architectures

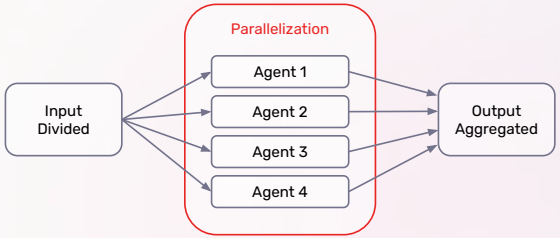
Routing



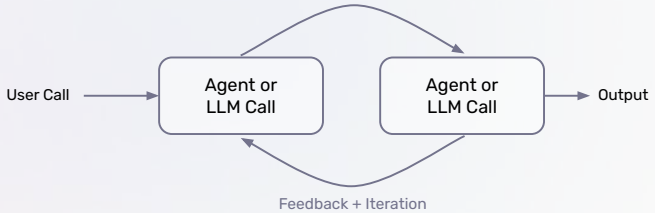
Orchestrator-Worker



Parallelization

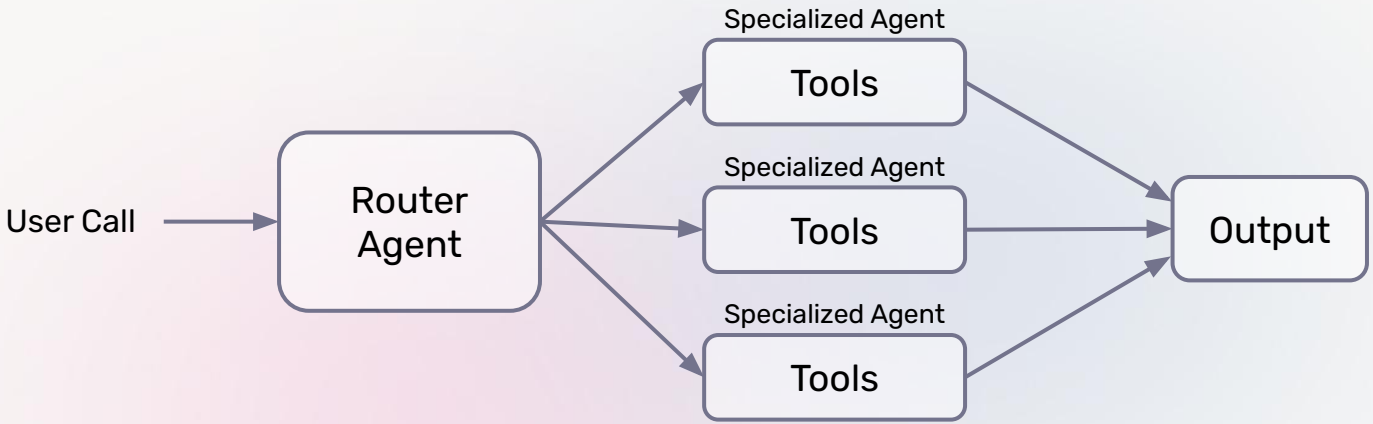


Evaluator-Optimizer



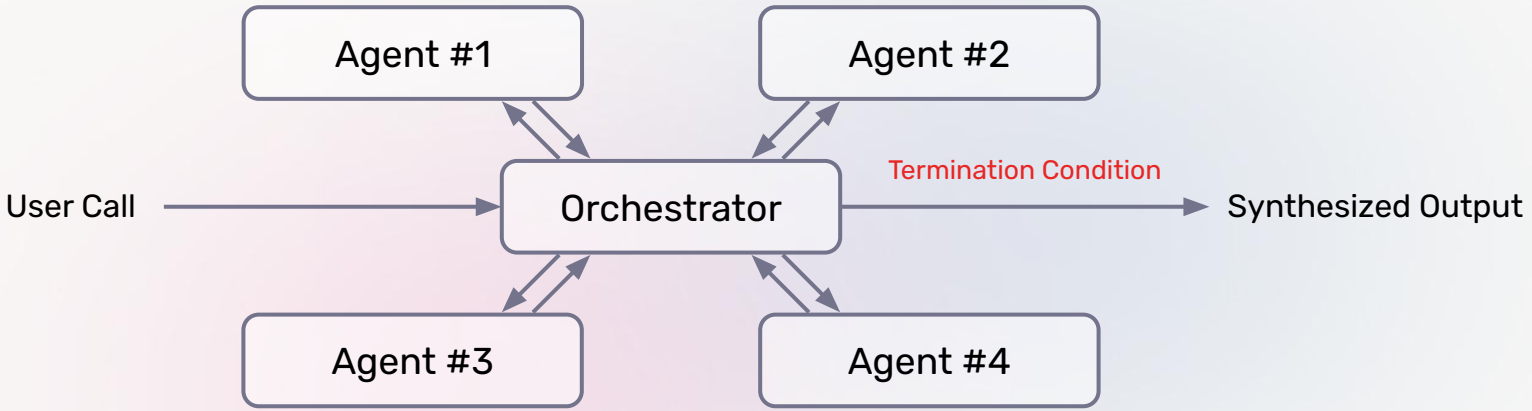
Routing

1. Router agent classifies task
2. Input is routed to a specialized agent
3. Specialized prompts and tools improve performance



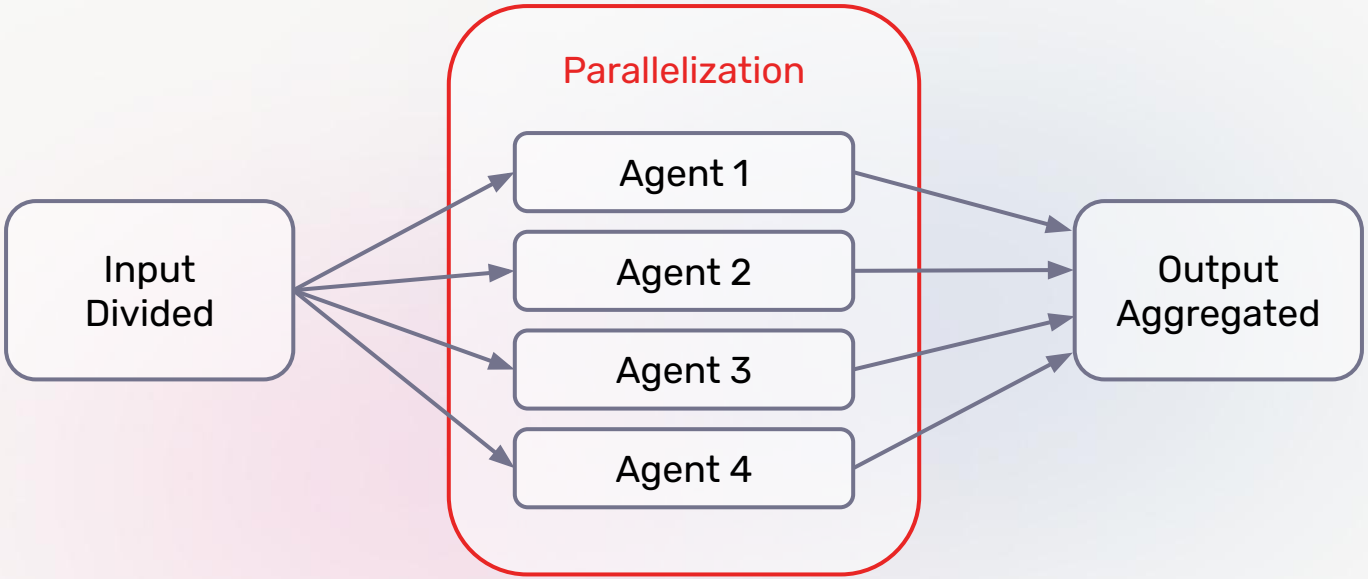
Orchestrator-Worker

1. Orchestrator agent splits tasks
2. Tasks are dynamically assigned to worker agents
3. Results from worker agents are synthesized



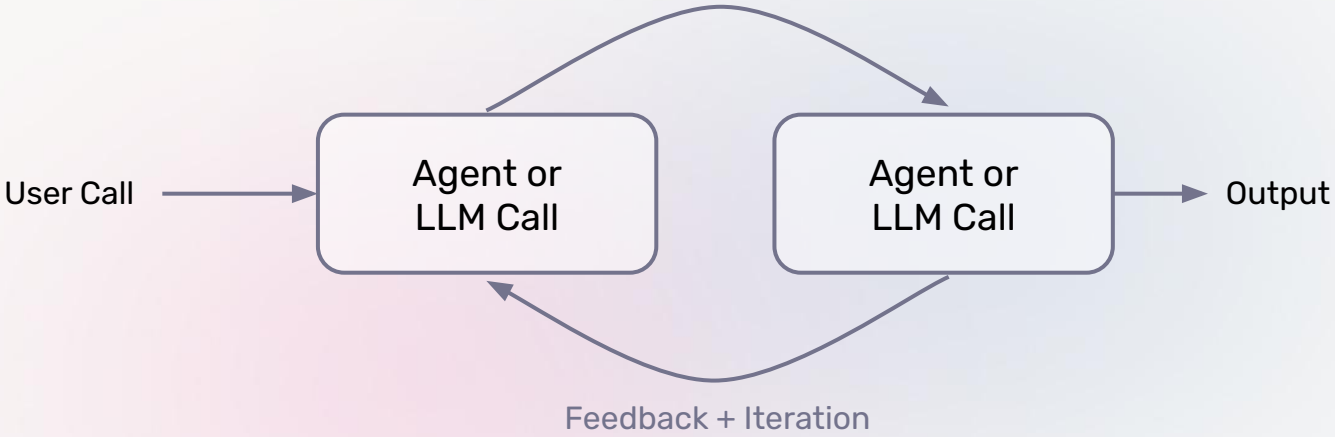
Parallelization

1. Input is divided into independent subtasks
2. Each agent or LLM completes its assigned subtask
3. Results are aggregated together to form final output



Evaluator-Optimizer

1. An LLM or agent generates a response
2. Another LLM evaluates the response and provides feedback
3. The loop continues until evaluator “approves” response



Common Agent Frameworks

Agent frameworks simplify the implementation of these architectures.

Framework	Language	Core Features
CrewAI	Python	Event-driven manager–worker framework with retries, observability, and async parallelism, supporting flexible memory for continuity
Mastra	TypeScript	Built-in streaming, suspend-resume, and tracing; configurable structured memory system with defaults for persistence.
Autogen	Python	Multi-agent orchestration, human-in-the-loop, conversation-driven workflows
Agno	Python	Team-based agents, role assignment, collaboration protocols
LangGraph	Python	Graph-based; controlled parallelism, recursion, and durable, checkpointed memory with resumability

Lab 3: Implement a Orchestrator-Worker Agent Workflow

- Navigate to labs/lab3_agent_architectures