Parallel agents - Agent Development Kit

Source URL: https://google.github.io/adk-docs/agents/workflow-agents/parallel-agents/

Parallel agents¶

The `ParallelAgent` is a workflow agent that executes its sub-agents *concurrently*. This dramatically speeds up workflows where tasks can be performed independently.

Use `ParallelAgent` when: For scenarios prioritizing speed and involving independent, resource-intensive tasks, a `ParallelAgent` facilitates efficient parallel execution. **When sub-agents operate without dependencies, their tasks can be performed concurrently**, significantly reducing overall processing time.

As with other workflow agents, the `ParallelAgent` is not powered by an LLM, and is thus deterministic in how it executes. That being said, workflow agents are only concerned with their execution (i.e. executing sub-agents in parallel), and not their internal logic; the tools or sub-agents of a workflow agent may or may not utilize LLMs.

Example¶

This approach is particularly beneficial for operations like multi-source data retrieval or heavy computations, where parallelization yields substantial performance gains. Importantly, this strategy assumes no inherent need for shared state or direct information exchange between the concurrently executing agents.

How it works¶

When the `ParallelAgent`'s `run_async()` method is called:

1. **Concurrent Execution:** It initiates the `run_async()` method of *each* sub-agent present in the `sub_agents` list *concurrently*. This means all the agents start running at (approximately) the same time.
2. **Independent Branches:** Each sub-agent operates in its own execution branch. There is ***no* automatic sharing of conversation history or state between these branches** during execution.
3. **Result Collection:** The `ParallelAgent` manages the parallel execution and, typically, provides a way to access the results from each sub-agent after they have completed (e.g., through a list of results or events). The order of results may not be deterministic.

Independent Execution and State Management¶

It's *crucial* to understand that sub-agents within a `ParallelAgent` run independently. If you *need* communication or data sharing between these agents, you must implement it explicitly. Possible approaches include:

- **Shared `InvocationContext`:** You could pass a shared `InvocationContext` object to each sub-agent. This object could act as a shared data store. However, you'd need to manage concurrent access to this shared context carefully (e.g., using locks) to avoid race conditions.
- **External State Management:** Use an external database, message queue, or other mechanism to manage shared state and facilitate communication between agents.
- **Post-Processing:** Collect results from each branch, and then implement logic to coordinate data afterwards.

Parallel Agent

Full Example: Parallel Web Research¶

Imagine researching multiple topics simultaneously:

1. **Researcher Agent 1:** An `LlmAgent` that researches "renewable energy sources."

2. **Researcher Agent 2:** An `LlmAgent` that researches "electric vehicle technology."
3. **Researcher Agent 3:** An `LlmAgent` that researches "carbon capture methods."

``` ParallelAgent(sub_agents=[ResearcherAgent1, ResearcherAgent2, ResearcherAgent3])

```

These research tasks are independent. Using a `ParallelAgent` allows them to run concurrently, potentially reducing the total research time significantly compared to running them sequentially. The results from each agent would be collected separately after they finish.

Full Code

PythonJava

```
 # Part of agent.py --> Follow https://google.github.io/adk-docs/get-s
# --- 1. Define Researcher Sub-Agents (to run in parallel) ---

# Researcher 1: Renewable Energy
researcher_agent_1 = LlmAgent(
    name="RenewableEnergyResearcher",
    model=GEMINI_MODEL,
    instruction="""You are an AI Research Assistant specializing in e
Research the latest advancements in 'renewable energy sources'.
Use the Google Search tool provided.
Summarize your key findings concisely (1-2 sentences).
Output *only* the summary.
""",
    description="Researches renewable energy sources.",
    tools=[google_search],
    # Store result in state for the merger agent
    output_key="renewable_energy_result"
)
```

```
# Researcher 2: Electric Vehicles
researcher_agent_2 = LlmAgent(
    name="EVResearcher",
    model=GEMINI_MODEL,
    instruction="""You are an AI Research Assistant specializing in t
Research the latest developments in 'electric vehicle technology'.
Use the Google Search tool provided.
Summarize your key findings concisely (1-2 sentences).
Output *only* the summary.
""",
    description="Researches electric vehicle technology.",
    tools=[google_search],
    # Store result in state for the merger agent
    output_key="ev_technology_result"
)

# Researcher 3: Carbon Capture
researcher_agent_3 = LlmAgent(
    name="CarbonCaptureResearcher",
    model=GEMINI_MODEL,
    instruction="""You are an AI Research Assistant specializing in c
Research the current state of 'carbon capture methods'.
Use the Google Search tool provided.
Summarize your key findings concisely (1-2 sentences).
Output *only* the summary.
""",
    description="Researches carbon capture methods.",
    tools=[google_search],
    # Store result in state for the merger agent
    output_key="carbon_capture_result"
)

# --- 2. Create the ParallelAgent (Runs researchers concurrently) ---
# This agent orchestrates the concurrent execution of the researchers
# It finishes once all researchers have completed and stored their re
parallel_research_agent = ParallelAgent(
```

```
    name="ParallelWebResearchAgent",
    sub_agents=[researcher_agent_1, researcher_agent_2, researcher_ag
    description="Runs multiple research agents in parallel to gather
)

# --- 3. Define the Merger Agent (Runs *after* the parallel agents) -
# This agent takes the results stored in the session state by the par
# and synthesizes them into a single, structured response with attrib
merger_agent = LlmAgent(
    name="SynthesisAgent",
    model=GEMINI_MODEL,  # Or potentially a more powerful model if ne
    instruction="""You are an AI Assistant responsible for combining

Your primary task is to synthesize the following research summaries,

**Crucially: Your entire response MUST be grounded *exclusively* on t

**Input Summaries:**

*   **Renewable Energy:**
    {renewable_energy_result}

*   **Electric Vehicles:**
    {ev_technology_result}

*   **Carbon Capture:**
    {carbon_capture_result}

**Output Format:**

## Summary of Recent Sustainable Technology Advancements

### Renewable Energy Findings
(Based on RenewableEnergyResearcher's findings)
[Synthesize and elaborate *only* on the renewable energy input summar
```

```
### Electric Vehicle Findings
(Based on EVResearcher's findings)
[Synthesize and elaborate *only* on the EV input summary provided abo

### Carbon Capture Findings
(Based on CarbonCaptureResearcher's findings)
[Synthesize and elaborate *only* on the carbon capture input summary

### Overall Conclusion
[Provide a brief (1-2 sentence) concluding statement that connects *o

Output *only* the structured report following this format. Do not inc
""",
    description="Combines research findings from parallel agents into
    # No tools needed for merging
    # No output_key needed here, as its direct response is the final
)

# --- 4. Create the SequentialAgent (Orchestrates the overall flow) -
# This is the main agent that will be run. It first executes the Para
# to populate the state, and then executes the MergerAgent to produce
sequential_pipeline_agent = SequentialAgent(
    name="ResearchAndSynthesisPipeline",
    # Run parallel research first, then merge
    sub_agents=[parallel_research_agent, merger_agent],
    description="Coordinates parallel research and synthesizes the re
)

root_agent = sequential_pipeline_agent
```

```
import com.google.adk.agents.LlmAgent;
import com.google.adk.agents.ParallelAgent;
import com.google.adk.agents.SequentialAgent;
import com.google.adk.events.Event;
```

```
import com.google.adk.runner.InMemoryRunner;
import com.google.adk.sessions.Session;
import com.google.adk.tools.GoogleSearchTool;
import com.google.genai.types.Content;
import com.google.genai.types.Part;
import io.reactivex.rxjava3.core.Flowable;

public class ParallelResearchPipeline {

  private static final String APP_NAME = "parallel_research_app";
  private static final String USER_ID = "research_user_01";
  private static final String GEMINI_MODEL = "gemini-2.0-flash";

  // Assume google_search is an instance of the GoogleSearchTool
  private static final GoogleSearchTool googleSearchTool = new Google

  public static void main(String[] args) {
    String query = "Summarize recent sustainable tech advancements.";
    SequentialAgent sequentialPipelineAgent = initAgent();
    runAgent(sequentialPipelineAgent, query);
  }

  public static SequentialAgent initAgent() {
    // --- 1. Define Researcher Sub-Agents (to run in parallel) ---
    // Researcher 1: Renewable Energy
    LlmAgent researcherAgent1 = LlmAgent.builder()
        .name("RenewableEnergyResearcher")
        .model(GEMINI_MODEL)
        .instruction("""
                     You are an AI Research Assistant specializing in
                     Research the latest advancements in 'renewable en
                     Use the Google Search tool provided.
                     Summarize your key findings concisely (1-2 senten
                     Output *only* the summary.
                     """)
        .description("Researches renewable energy sources.")
```

```
        .tools(googleSearchTool)
        .outputKey("renewable_energy_result") // Store result in stat
        .build();

    // Researcher 2: Electric Vehicles
    LlmAgent researcherAgent2 = LlmAgent.builder()
        .name("EVResearcher")
        .model(GEMINI_MODEL)
        .instruction("""
                    You are an AI Research Assistant specializing in
                    Research the latest developments in 'electric veh
                    Use the Google Search tool provided.
                    Summarize your key findings concisely (1-2 senten
                    Output *only* the summary.
                    """)
        .description("Researches electric vehicle technology.")
        .tools(googleSearchTool)
        .outputKey("ev_technology_result") // Store result in state
        .build();

    // Researcher 3: Carbon Capture
    LlmAgent researcherAgent3 = LlmAgent.builder()
        .name("CarbonCaptureResearcher")
        .model(GEMINI_MODEL)
        .instruction("""
                    You are an AI Research Assistant specializing in
                    Research the current state of 'carbon capture met
                    Use the Google Search tool provided.
                    Summarize your key findings concisely (1-2 senten
                    Output *only* the summary.
                    """)
        .description("Researches carbon capture methods.")
        .tools(googleSearchTool)
        .outputKey("carbon_capture_result") // Store result in state
        .build();
```

```
// --- 2. Create the ParallelAgent (Runs researchers concurrently
// This agent orchestrates the concurrent execution of the resear
// It finishes once all researchers have completed and stored the
ParallelAgent parallelResearchAgent =
    ParallelAgent.builder()
        .name("ParallelWebResearchAgent")
        .subAgents(researcherAgent1, researcherAgent2, researcher
        .description("Runs multiple research agents in parallel t
        .build();

// --- 3. Define the Merger Agent (Runs *after* the parallel agen
// This agent takes the results stored in the session state by th
// and synthesizes them into a single, structured response with a
LlmAgent mergerAgent =
    LlmAgent.builder()
        .name("SynthesisAgent")
        .model(GEMINI_MODEL)
        .instruction(
            """
                  You are an AI Assistant responsible for combini
                  Your primary task is to synthesize the followin
                  **Crucially: Your entire response MUST be groun
                  **Input Summaries:**

                  *   **Renewable Energy:**
                      {renewable_energy_result}

                  *   **Electric Vehicles:**
                      {ev_technology_result}

                  *   **Carbon Capture:**
                      {carbon_capture_result}

                  **Output Format:**

                  ## Summary of Recent Sustainable Technology Adv
```

```
                    ### Renewable Energy Findings
                    (Based on RenewableEnergyResearcher's findings)
                    [Synthesize and elaborate *only* on the renewab

                    ### Electric Vehicle Findings
                    (Based on EVResearcher's findings)
                    [Synthesize and elaborate *only* on the EV inpu

                    ### Carbon Capture Findings
                    (Based on CarbonCaptureResearcher's findings)
                    [Synthesize and elaborate *only* on the carbon

                    ### Overall Conclusion
                    [Provide a brief (1-2 sentence) concluding stat

                    Output *only* the structured report following t
                    """)
            .description(
                "Combines research findings from parallel agents into
            // No tools needed for merging
            // No output_key needed here, as its direct response is t
            .build();

    // --- 4. Create the SequentialAgent (Orchestrates the overall fl
    // This is the main agent that will be run. It first executes the
    // to populate the state, and then executes the MergerAgent to pr
    SequentialAgent sequentialPipelineAgent =
        SequentialAgent.builder()
            .name("ResearchAndSynthesisPipeline")
            // Run parallel research first, then merge
            .subAgents(parallelResearchAgent, mergerAgent)
            .description("Coordinates parallel research and synthesiz
            .build();

    return sequentialPipelineAgent;
```

```
  }

  public static void runAgent(SequentialAgent sequentialPipelineAgent
    // Create an InMemoryRunner
    InMemoryRunner runner = new InMemoryRunner(sequentialPipelineAgen
    // InMemoryRunner automatically creates a session service. Create
    Session session = runner.sessionService().createSession(APP_NAME,
    Content userMessage = Content.fromParts(Part.fromText(query));

    // Run the agent
    Flowable<Event> eventStream = runner.runAsync(USER_ID, session.id

    // Stream event response
    eventStream.blockingForEach(
        event -> {
          if (event.finalResponse()) {
            System.out.printf("Event Author: %s \n Event Response: %s
          }
        });
  }
}
```