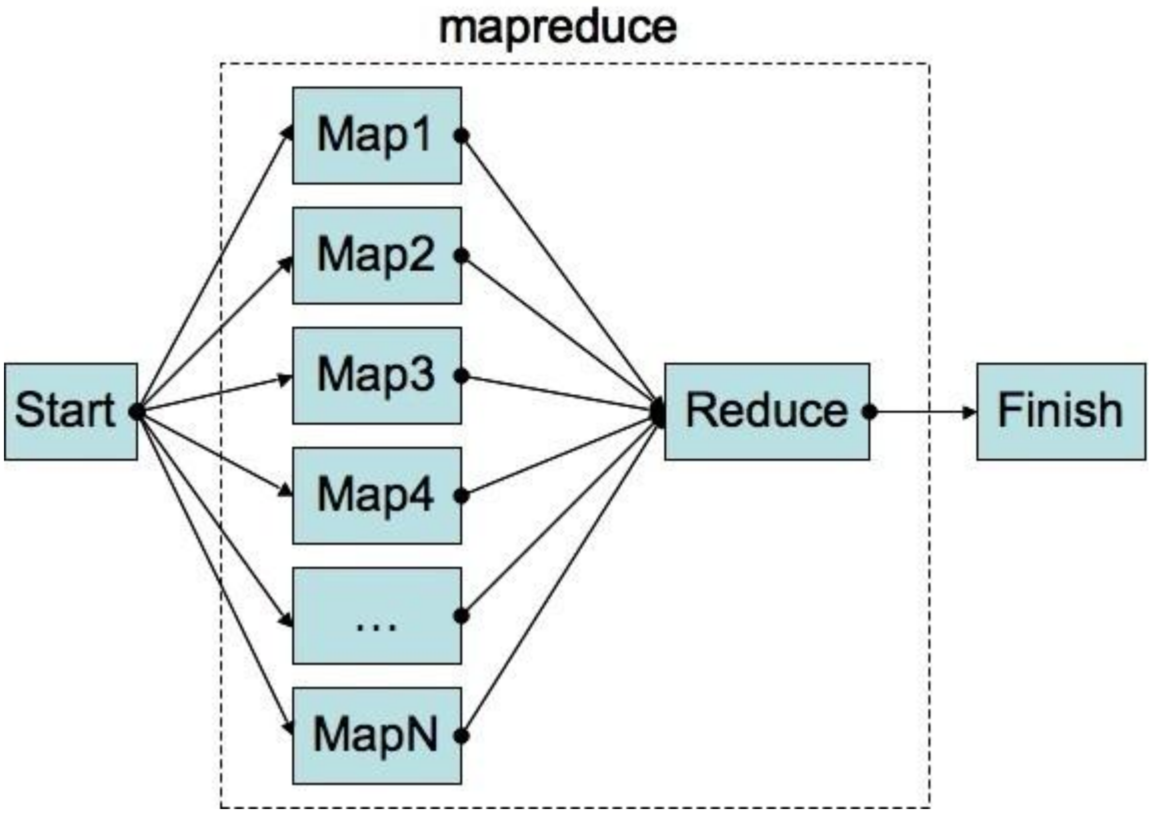
Leveraging LangGraph's Send API for Dynamic and Parallel Workflow Execution

Introduction

In AI-driven applications, managing dynamic and parallel workflows efficiently is crucial. LangGraph's **Send API** is a powerful feature that enables dynamic state distribution across multiple node instances, making it an ideal tool for handling complex, unpredictable workloads. Whether for parallel execution, **map-reduce** operations, or conditional task routing, the Send API enhances flexibility and scalability.



Key Features of LangGraph's Send API

1. Handling Unknown Object Counts

One of the biggest challenges in workflow automation is dealing with varying numbers of objects. In traditional workflows, predefined paths restrict flexibility. However, LangGraph's Send API dynamically manages scenarios where the number of elements is unknown in advance.

Example: User Trip Booking

Consider a user booking a trip. They may choose:

- A **flight** only
- A **flight and car rental**
- A **flight, car rental, and hotel**

Since we don't know the user’s choices beforehand, the Send API dynamically adapts, creating and distributing processing nodes for each selection in real time.

2. Dynamic State Distribution

The Send API enables parallel execution by distributing different states to multiple instances of a node. This means that each part of a workflow can be processed independently, significantly improving efficiency.

How it works in trip booking:

- If a user books a flight and car, two separate nodes handle **flight processing** and **car rental processing** in parallel.
- If the user adds a hotel, another instance is created to process the **hotel reservation** independently.

3. Flexible State Management

Unlike traditional static workflows, where state management is rigid, the Send API allows sent states to differ from the core graph’s state. This means that each workflow instance can have its own unique data while still being part of the larger execution framework.

4. Utilization of Conditional Edges

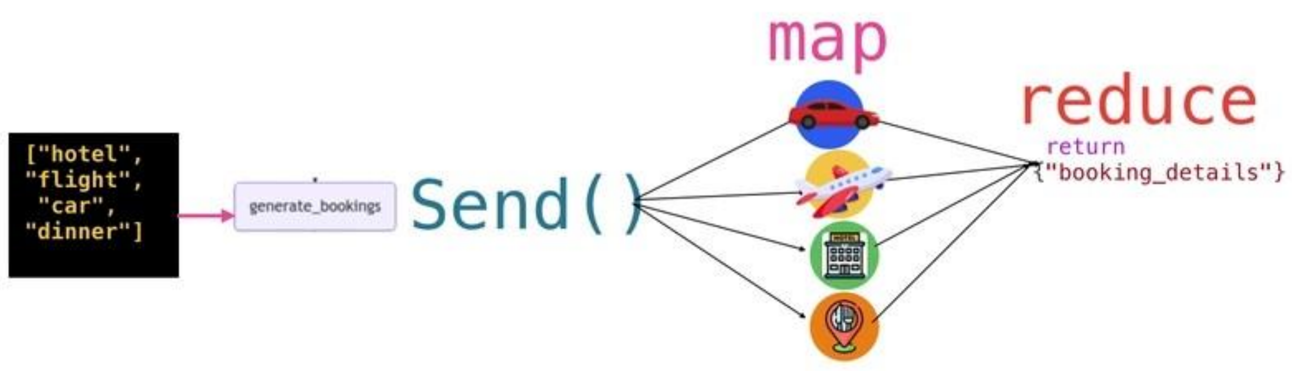
The API uses **conditional edges** to route tasks dynamically based on workflow states. If a user cancels a hotel booking but keeps the flight and car rental, only the relevant nodes process the required tasks, ensuring efficiency.

Use Case: Map-Reduce in AI Workflows

The Send API is particularly useful in **map-reduce** operations, where tasks need to be distributed across multiple nodes for parallel processing. In AI workflows, this can be used for:

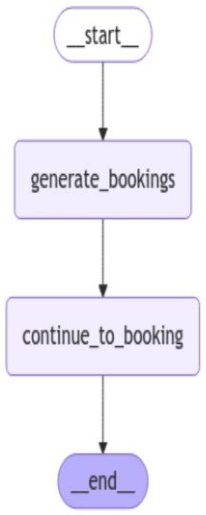
- Distributed data processing Multi-step
- reasoning tasks ● Workflow orchestration in
LLM-based applications

Implementation

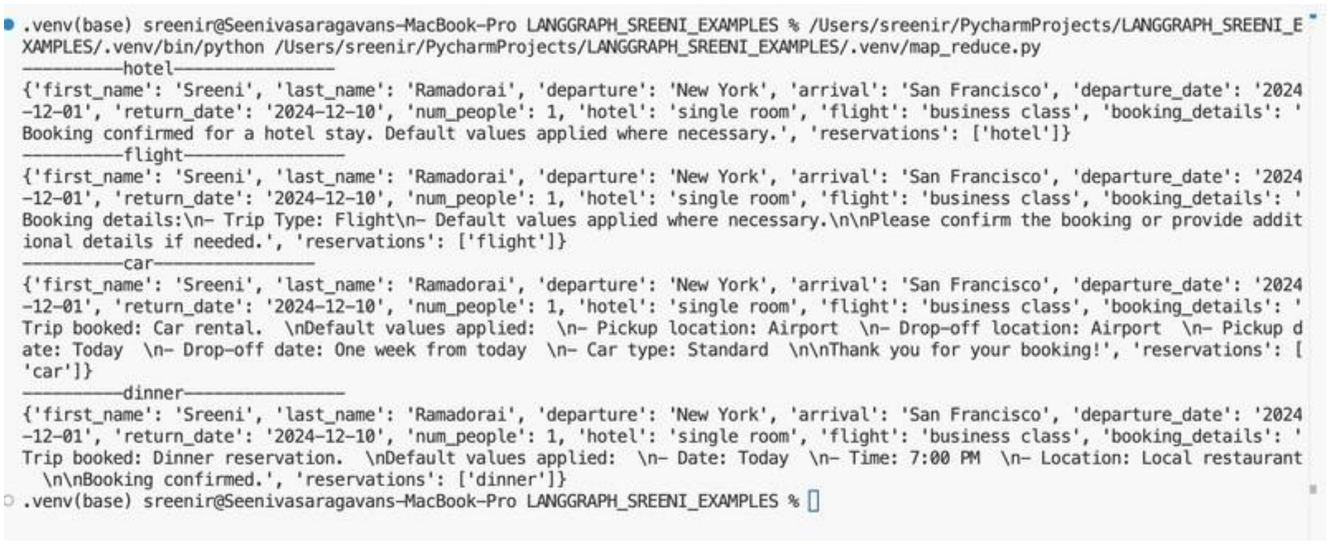


```
from typing import TypedDict
from langgraph.graph import START, END,StateGraph
from langchain_openai.chat_models import AzureChatOpenAI
from dotenv import load_dotenv
from langgraph.types import Send
import os

load_dotenv()

llm = AzureChatOpenAI(
   azure_deployment="gpt-4o-mini",
   api_version="2024-08-01-preview",
   temperature=0,
   max_tokens=None,
   timeout=None,
   max_retries=2
)

booking_prompt = """Book the trip for the customer based on the
 following trip booking information: {reservation}. Please fill
 missing requirements with default values if not provided by
 the user. Return only minimal information."""

class TripBookingState(TypedDict):
   first_name: str
   last_name: str
   departure:str
   arrival:str
   departure_date:str
   return_date:str
   num_people: int
   hotel: str
   flight: str
   booking_details: dict
   reservations: list

class Bookings(TypedDict):
   bookings: list[str]


def generate_bookings(state: TripBookingState):
   prompt=booking_prompt.format(reservation=state["reservations"])
   print(f"----------{reservation}----------------")
   response = llm.invoke(prompt)
   return {"booking_details":response.content}
```

```
def continue_to_booking(state: TripBookingState) -> dict:
   return {"send": [Send("generate_bookings",
                        {"reservation": reservation})
           for reservation in state["reservations"]]
   }


graph = StateGraph(TripBookingState)
graph.add_node("generate_bookings",generate_bookings)
graph.add_node("continue_to_booking",continue_to_booking)
graph.add_edge(START,"generate_bookings")
graph.add_edge("generate_bookings","continue_to_booking")
graph.add_edge("continue_to_booking",END)
app = graph.compile()

image = app.get_graph().draw_mermaid_png()
with open("map_reduce.png","wb") as file:
   file.write(image)

for reservation in ["hotel", "flight", "car", "dinner"]:
   response = app.invoke({
      "first_name": "Sreeni",
      "last_name": "Ramadorai",
      "departure": "New York",
      "arrival": "San Francisco",
      "departure_date": "2024-12-01",
      "return_date": "2024-12-10",
      "num_people": 1,
      "hotel": "single room",
      "flight": "business class",
      "booking_details": {},
      "reservations": [reservation]
   })
   print(response)
```



Output



```
.venv(base) sreenir@Seenivasaragavans-MacBook-Pro LANGGRAPH_SREENI_EXAMPLES % /Users/sreenir/PycharmProjects/LANGGRAPH_SREENI_E
XAMPLES/.venv/bin/python /Users/sreenir/PycharmProjects/LANGGRAPH_SREENI_EXAMPLES/.venv/map_reduce.py
-----------hotel-----------------
{'first_name': 'Sreeni', 'last_name': 'Ramadorai', 'departure': 'New York', 'arrival': 'San Francisco', 'departure_date': '2024
-12-01', 'return_date': '2024-12-10', 'num_people': 1, 'hotel': 'single room', 'flight': 'business class', 'booking_details': '
Booking confirmed for a hotel stay. Default values applied where necessary.', 'reservations': ['hotel']}
-----------flight----------------
{'first_name': 'Sreeni', 'last_name': 'Ramadorai', 'departure': 'New York', 'arrival': 'San Francisco', 'departure_date': '2024
-12-01', 'return_date': '2024-12-10', 'num_people': 1, 'hotel': 'single room', 'flight': 'business class', 'booking_details': '
Booking details:\n- Trip Type: Flight\n- Default values applied where necessary.\n\nPlease confirm the booking or provide addit
ional details if needed.', 'reservations': ['flight']}
-----------car-----------------
{'first_name': 'Sreeni', 'last_name': 'Ramadorai', 'departure': 'New York', 'arrival': 'San Francisco', 'departure_date': '2024
-12-01', 'return_date': '2024-12-10', 'num_people': 1, 'hotel': 'single room', 'flight': 'business class', 'booking_details': '
Trip booked: Car rental.  \nDefault values applied:  \n- Pickup location: Airport  \n- Drop-off location: Airport  \n- Pickup d
ate: Today  \n- Drop-off date: One week from today  \n- Car type: Standard  \n\nThank you for your booking!', 'reservations': [
'car']}
-----------dinner----------------
{'first_name': 'Sreeni', 'last_name': 'Ramadorai', 'departure': 'New York', 'arrival': 'San Francisco', 'departure_date': '2024
-12-01', 'return_date': '2024-12-10', 'num_people': 1, 'hotel': 'single room', 'flight': 'business class', 'booking_details': '
Trip booked: Dinner reservation.  \nDefault values applied:  \n- Date: Today  \n- Time: 7:00 PM  \n- Location: Local restaurant
  \n\nBooking confirmed.', 'reservations': ['dinner']}
.venv(base) sreenir@Seenivasaragavans-MacBook-Pro LANGGRAPH_SREENI_EXAMPLES %
```

Conclusion

LangGraph's Send API empowers developers to build **scalable, flexible, and efficient** workflows for AI-driven applications. Whether managing dynamic trip bookings or distributing computational tasks, this feature significantly enhances workflow automation. By leveraging its ability to handle unknown object counts, enable dynamic state distribution, and utilize conditional routing, developers can create intelligent applications that adapt to real-world complexities.

Thanks
Sreeni Ramadorai

H Company PROMOTED

...