扩展成 Flask 服务版本的 Saga，甚至加事件编排/调度器

ChatGPT said:

下面给你一个基于 **Flask + 内存调度器** 实现的简易 **Saga 模式服务版示例**。它包括：

- 独立的 Saga 事务管理器（调度器）
- 多个步骤的服务端点，分别执行局部事务和补偿事务
- Saga 事务发起接口
- 简单的事件驱动调用和补偿执行流程

项目结构（示意）

bash

CopyEdit

```
saga_demo/
├── saga_manager.py     # Saga 事务调度器（内存版）
├── service_a.py        # 服务 A，模拟订单服务
├── service_b.py        # 服务 B，模拟库存服务
├── service_c.py        # 服务 C，模拟支付服务
└── run_saga.py         # 启动 saga 事务
```

1. saga_manager.py — Saga 调度器（内存状态）

python

CopyEdit

```
from flask import Flask, request, jsonify
import requests
import threading

app = Flask(__name__)

# 事务状态存储（内存）
saga_states = {}

# 事务步骤定义（顺序执行）
```

```
SAGA_STEPS = [
    {
        "name": "create_order",
        "action_url": "http://localhost:5001/create",
        "compensate_url": "http://localhost:5001/cancel"
    },
    {
        "name": "reserve_inventory",
        "action_url": "http://localhost:5002/reserve",
        "compensate_url": "http://localhost:5002/release"
    },
    {
        "name": "process_payment",
        "action_url": "http://localhost:5003/pay",
        "compensate_url": "http://localhost:5003/refund"
    }
]

def call_service(url, txn_id):
    try:
        resp = requests.post(url, json={"txn_id": txn_id},
timeout=5)
        resp.raise_for_status()
        return True
    except Exception as e:
        print(f"调用服务失败 {url}，错误：{e}")
        return False

def execute_saga(txn_id):
    saga_states[txn_id] = {"status": "in_progress", "step":
0}

    for i, step in enumerate(SAGA_STEPS):
        print(f"执行步骤 {step['name']}，事务 {txn_id}")
        success = call_service(step['action_url'], txn_id)
        if not success:
            print(f"步骤失败，开始补偿，事务 {txn_id}")
            # 失败，补偿已执行的步骤
            for j in range(i-1, -1, -1):
                comp_step = SAGA_STEPS[j]
                print(f"补偿步骤 {comp_step['name']}，事务 {txn_id}")
                call_service(comp_step['compensate_url'], txn_id)
            saga_states[txn_id]['status'] = 'failed'
            return
        saga_states[txn_id]['step'] = i + 1

    saga_states[txn_id]['status'] = 'success'

@app.route('/start_saga', methods=['POST'])
def start_saga():
    import uuid
    txn_id = str(uuid.uuid4())
    threading.Thread(target=execute_saga, args=(txn_id,), daemon=True).start()
    return jsonify({"txn_id": txn_id, "message": "Saga started"})

@app.route('/status/<txn_id>', methods=['GET'])
def status(txn_id):
    state = saga_states.get(txn_id, None)
    if not state:
        return jsonify({"error": "unknown txn_id"}), 404
    return jsonify(state)

if __name__ == "__main__":
    app.run(port=5000)
```

2. 示例服务：service_a.py (订单服务)

python
CopyEdit

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/create', methods=['POST'])
def create_order():
    data = request.json
    print(f"订单创建，事务 {data['txn_id']}")
    return jsonify({"status": "created"})

@app.route('/cancel', methods=['POST'])
def cancel_order():
    data = request.json
    print(f"订单取消，事务 {data['txn_id']}")
    return jsonify({"status": "cancelled"})

if __name__ == "__main__":
    app.run(port=5001)
```

3. 示例服务：service_b.py (库存服务)

python
CopyEdit

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/reserve', methods=['POST'])
def reserve_inventory():
    data = request.json
    print(f"库存预留，事务 {data['txn_id']}")
    # 这里可以模拟失败，比如：
    # return jsonify({"error": "库存不足"}), 500
    return jsonify({"status": "reserved"})

@app.route('/release', methods=['POST'])
def release_inventory():
    data = request.json
    print(f"库存释放，事务 {data['txn_id']}")
    return jsonify({"status": "released"})

if __name__ == "__main__":
    app.run(port=5002)
```

4. 示例服务：service_c.py (支付服务)

python
CopyEdit
```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/pay', methods=['POST'])
def process_payment():
    data = request.json
    print(f"支付处理中，事务 {data['txn_id']}")
    # 模拟成功：
    return jsonify({"status": "paid"})
    # 模拟失败测试补偿：
    # return jsonify({"error": "支付失败"}), 500

@app.route('/refund', methods=['POST'])
def refund_payment():
    data = request.json
    print(f"退款，事务 {data['txn_id']}")
    return jsonify({"status": "refunded"})

if __name__ == "__main__":
    app.run(port=5003)
```

5. 测试流程

1. 启动所有服务（每个在独立终端）：

bash
CopyEdit
```bash
python saga_manager.py
python service_a.py
python service_b.py
python service_c.py
```

2. 发起 Saga 事务：

bash
CopyEdit
```bash
curl -X POST http://localhost:5000/start_saga
```

3. 查询事务状态：

```
bash
CopyEdit
curl http://localhost:5000/status/<txn_id>
```

6. 总结

- **SagaManager** 作为事务协调器，顺序调用每个步骤，失败则执行补偿步骤。
- 各微服务通过 REST API 实现各自局部事务和补偿事务。
- 使用 Flask + requests 模拟真实分布式环境。
- 可以扩展加入持久化（数据库或 Redis）、异步消息队列等。

你写一套更完整的 Saga 生产级示例（含持久化状态、消息队列事件驱动）或者详细讲解每个部分实现细节