Agentics Tutorial

Lecture 4: *Scaling Out Agentic Workflows for Real-World Applications*

Guest lectures at Columbia University Class on Agentic AI, Fintech, and the Data Economy. - Prof. Agostino Capponi

Alfio Massimiliano Gliozzo

IBM research

gliozzo@us.ibm.com

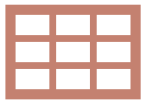
Outline

- Scaling out GenAI with State Graphs: Langgraph
- Scaling out GenAI with MapReduce: Agentics
- Example Applications
  - Multiple Choice QA (Decision Making)
  - Text2SQL
- Hands on: explaining market volatility

Scaling out GenAI workflows

Problems:

LLM calls are often executed in the order of seconds

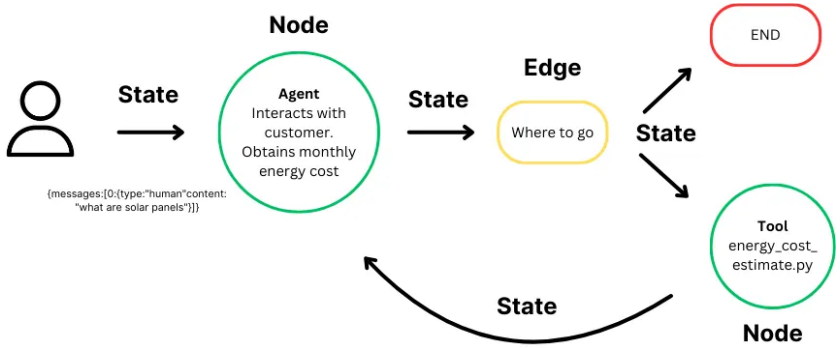


It takes 1 h to execute 1000 llm calls, which is a small size for financial analytics

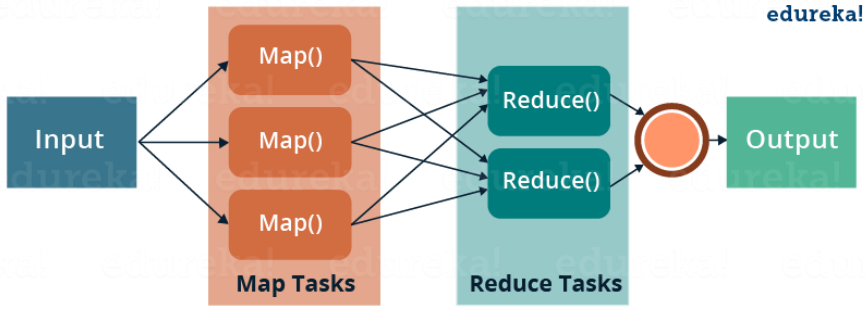
Natural Language based IO will trigger dozens of exceptions

Solutions:

State Graphs



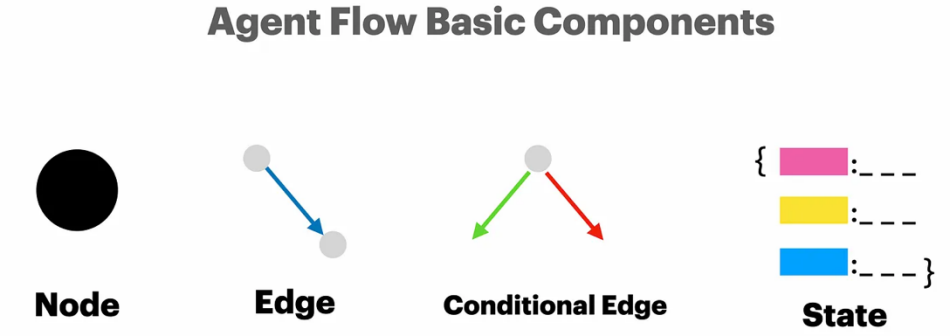
Map Reduce[1]

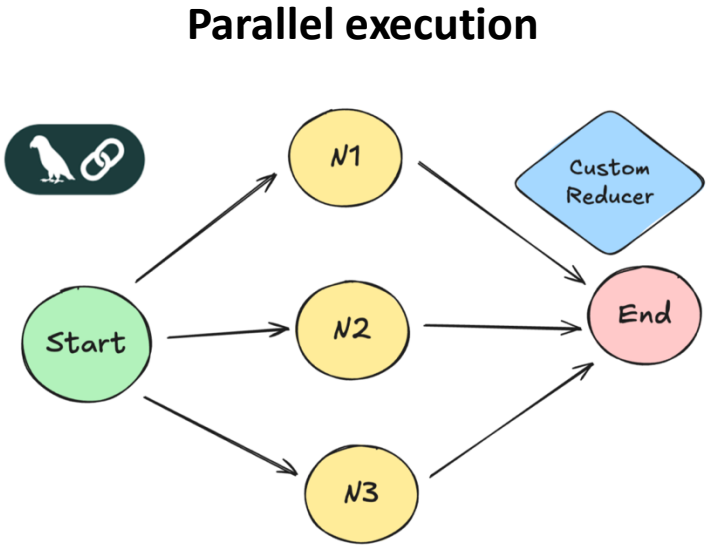


[1]Source: https://www.edureka.co/blog/mapreduce-tutorial/

State Graphs in LangGraph

- **State**: Pydantic Objects
- **Nodes**: functions transforming states into other states
- **Edges**: Connections between nodes
- **Graph**: Set of Node and Edges
- **Parallel execution:** implemented using multiple (dynamic) edges from a single node. Custom logic has to be defined.

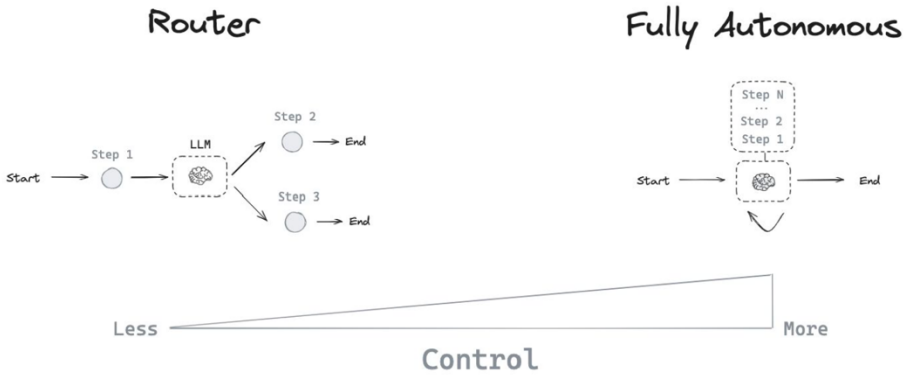




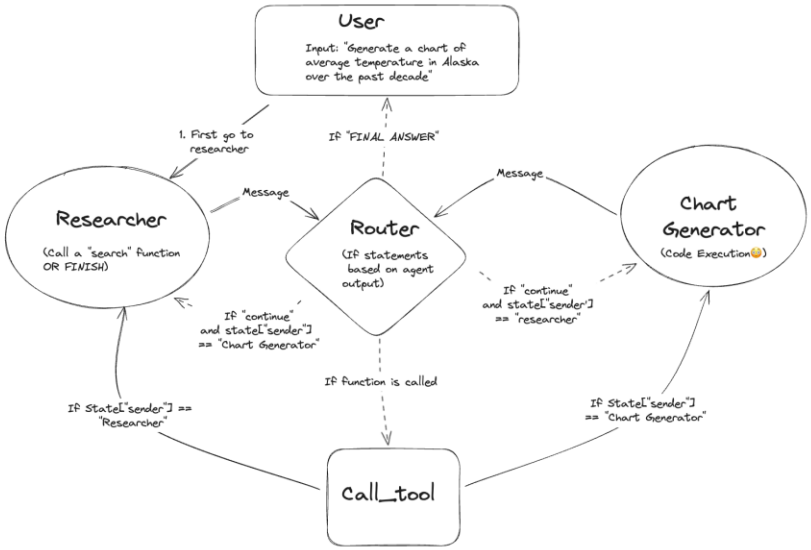
Orchestrating Agents, Function Calls, Tools and LLM

From https://blog.langchain.com/langgraph-multi-agent-workflows/

Many kinds of agents!



Example Agent



Developing agents in LangGraph: pros and cons

✅ Pros

- **Scale-out**: Enables parallel execution of nodes in state graphs.
- **Persistence**: Each step in the graph can be persisted in memory or a database.
- **Multithreading / Concurrency**: Supports multiple concurrent sessions and users.
- **Ecosystem Integration**: Reuses LangChain tools, data loaders, and agentic design patterns.

⚠️ Cons

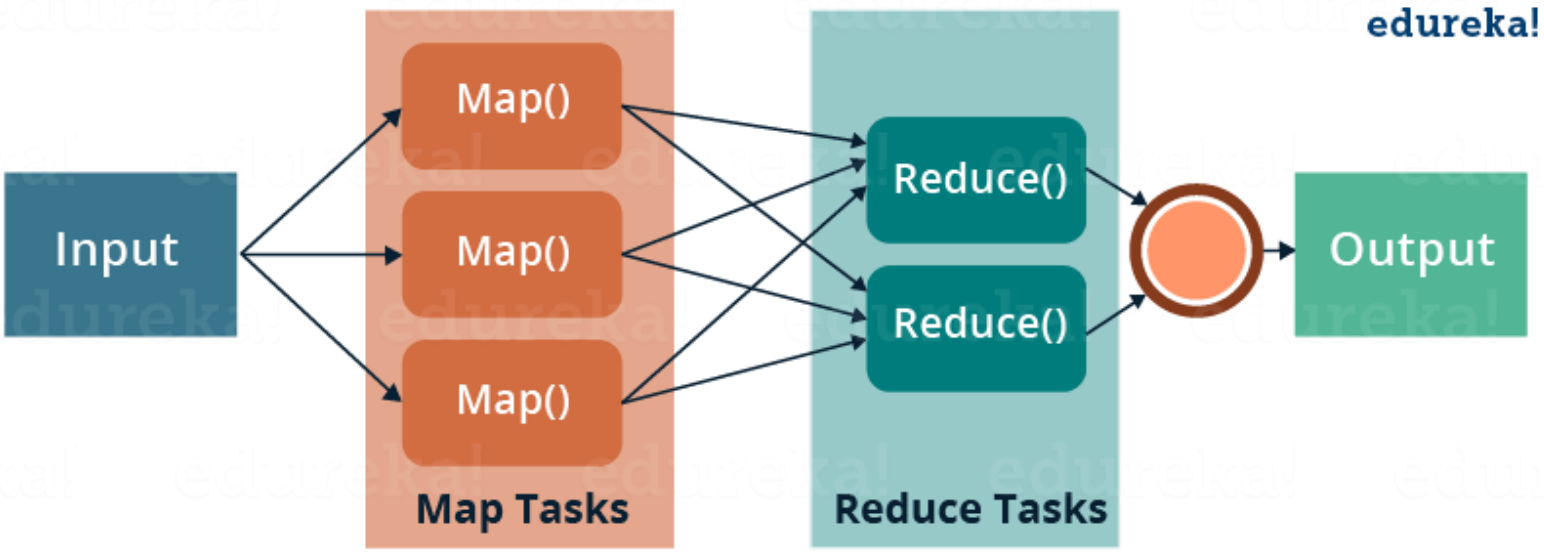
- **Low-Level Abstraction**: Developers need to manage fine-grained details.
- **Difficult Development Cycle**: Harder to develop, debug, and test compared to higher-level frameworks.
- **Conversational Bias**: State graphs were primarily designed for conversational loops
- **Workflow Complexity**: While they can model workflows, they introduce significant complexity during development and maintenance.

Outline

- Scaling out GenAI with State Graphs: Langgraph
- Scaling out GenAI with MapReduce: Agentics
- Example Applications
  - Multiple Choice QA (Decision Making)
  - Text2SQL
- Hands on: explaining market volatility

Map Reduce

map	(k1,v1)	→ list(k2,v2)
reduce	(k2,list(v2))	→ list(v2)



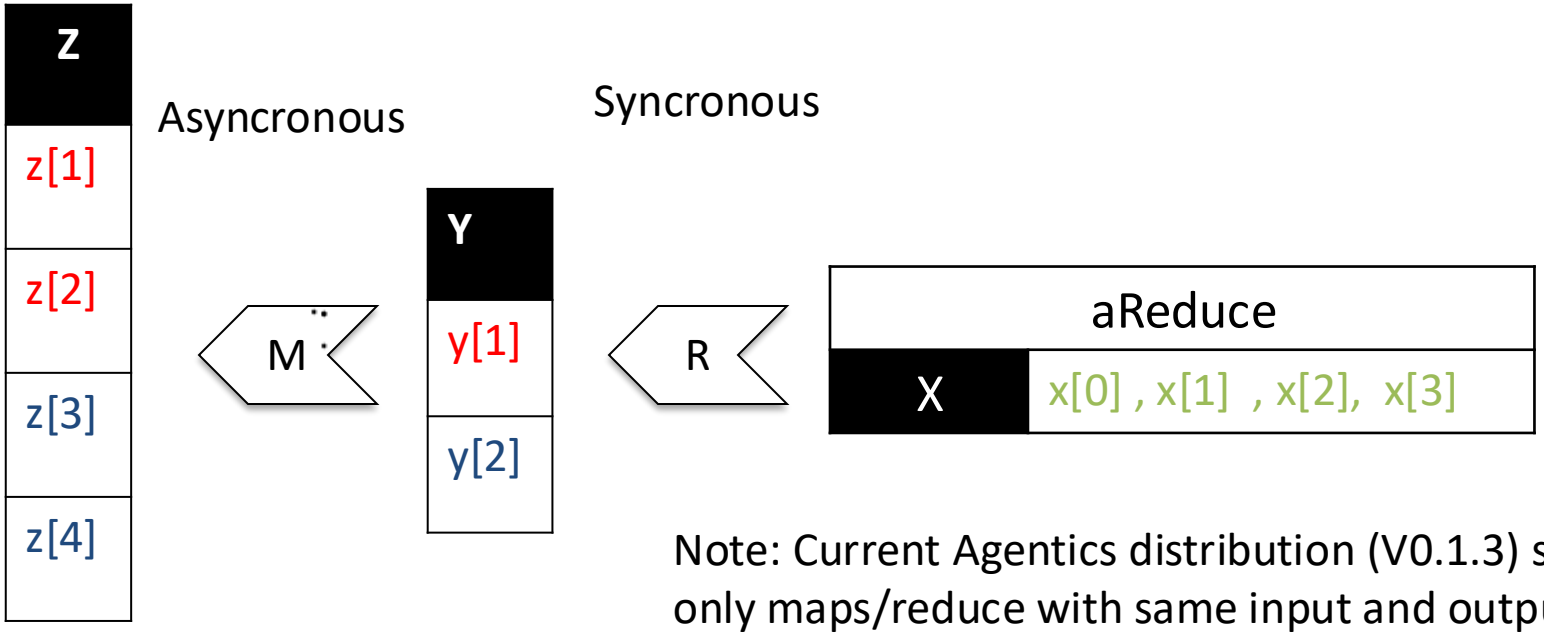
Source https://www.edureka.co/blog/mapreduce-tutorial/

aMapReduce Framework

Asynchronous MapReduce. To scale beyond single-step transduction, LTA provides two higher-order operators:

$$\texttt{aMap} : (AG[X], f) \to AG[Y], \qquad \texttt{Reduce} : (AG[X], g) \to AG[Y],$$

where $f : X \to \text{List}[Y]$ is applied independently to each x_i (filter/transform/fan-out), and $g : \text{List}[X] \to Y$ aggregates many states (summaries, rankings, joins). Because $\ll$ is stateless,



Note: Current Agentics distribution (V0.1.3) supports only maps/reduce with same input and output type and 1:1 amaps

Examples

aMap

```python
## Define a function to get news for a given date using the DDGS search engine
## Note that the similar functionalities can be implemented using MCP tools in AGs
async def get_news(state):
    state.news=str(DDGS().text(
        f"What happended to the stock market and dow jones on {state.date}",
        max_results=10))
    return state

# Now get news for the top 10 days with highest volatility using amap
highest_volatility_days = await highest_volatility_days.amap(get_news)
```

aReduce

```python
async def get_highest_volatility_days(
    states: list[StockMarketState],
) -> list[StockMarketState]:

    # sort the states by volatility and return the top 10, define a new AG with these states
    return sorted(
        states,
        key=lambda x: abs(x.daily_range) if x.daily_range is not None else 0,
        reverse=True,
    )[:10]

# apply the reduce function to get the top 10 days with highest volatility
highest_volatility_days = await dj_data.areduce(get_highest_volatility_days)
```

Outline

- Scaling out GenAI with State Graphs: Langgraph
- Scaling out GenAI with MapReduce: Agentics
- Example Applications
  - Multiple Choice QA (Decision Making)
  - Text2SQL
- Hands on: explaining market volatility

Outline

- Scaling out GenAI with State Graphs: Langgraph
- Scaling out GenAI with MapReduce: Agentics
- Example Applications
  - Multiple Choice QA (Decision Making)
  - Text2SQL
- Hands on: explaining market volatility

Failure Sensor IQ benchmark

- **What?** FailureSensorIQ: Multiple-Choice QA for testing the Language Model's knowledge and reasoning on relationships between failure modes and related sensors.
- **Why?** Application for Maximo Application Suite to enable agents in Industry 4.0 perform asset failure diagnosis, anomaly detection, failure prediction, and energy optimization
- **Research Question:** Models already have a lot of knowledge, but can they successfully combine this knowledge and reason to come to the right answer?

Example question:

When a electric motor has bearing damage, which sensor out of the choices should be the sensor to be monitored for this failure if I want to build an anomaly detection model

A: axial flux
B: vibration
C: voltage
D: cooling gas
E: partial discharge

FailureSensorIQ link

Failure Sensor IQ : (non-AG) Baseline

1) Present each question and prompt it to return a dictionary format with the fields "reasoning" and "answer".
2) Retry maximum 3 times if the answer is not in an acceptable dict format

```
{"reasoning":"<your reasoning>",
"answer":["<answer letter>"]}
```

Failure Sensor IQ : AG Based Solution

Answer

id: B

confidence: 0.9

assessment: The vibration sensor is the most relevant to monitor for bearing damage in an electric motor, as bearing damage typically generates unique vibration patterns that can be detected and analyzed for anomaly detection.

```python
class Answer(BaseModel):
    id:Optional[str]=Field(None,description=
        "The id of the selected answer")
    confidence:Optional[float]=None
    assessment:Optional[str]= Field(None,
        description= """The rationale why
        you believe the answer is correct""")
```

```python
class FailureSensorIQ(BaseModel):
    id: Optional[int] = None
    question: Optional[str] = None
    options: Optional[list[str]] = None
    option_ids: Optional[list[str]] = None
    question_first: Optional[bool] = None
    correct: Optional[list[bool]] = None
    text_type: Optional[str] = None
    asset_name: Optional[str] = None
    relevancy: Optional[str] = None
    question_type: Optional[str] = None
    subject: Optional[str] = None
    system_answer  Optional[Answer] = None
```

Domain-Specific Efficacy of Agentics framework on FailureSensorIQ benchmark

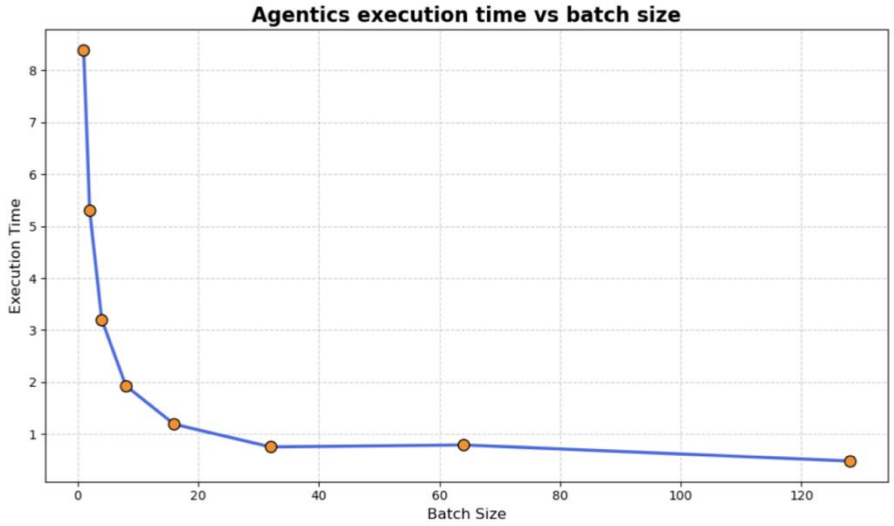
Best performance for small open source model (8b) is comparable to Fronter models
OpenAI-o1 60.4

OpenAI-o1 (Frontier Model) 60.4

Model	# Params	Baseline	Agentics
Qwen3-8B	8B	45.86	60.18 (+14.32)
Llama-3.3-70B	70B	41.69	50.73 (+9.04)
Mistral-Large	123B	50.09	58.41 (+8.32)
Llama-3-405B	405B	51.26	52.90 (+1.64)

Table 6: Accuracy (%) of on 2,667 FailureSensorIQ instances.

Note that the backend LLM supports no more than 20 parallel call, agentics speeds up by ~20x which is optimal



Outline

- Scaling out GenAI with State Graphs: Langgraph
- Scaling out GenAI with MapReduce: Agentics
- Example Applications
  - Multiple Choice QA (Decision Making)
  - Text2SQL
- Hands on: explaining market volatility

```
{
  "question_id": 0,
  "db_id": "california_schools",
  "question": "What is the highest eligible free rate for K-12 students in the schools in Alameda County?",
  "evidence": "Eligible free rate for K-12 = `Free Meal Count (K-12)` / `Enrollment (K-12)`",
  "SQL": "SELECT `Free Meal Count (K-12)` / `Enrollment (K-12)` FROM frpm WHERE `County Name` = 'Alameda' ORDER BY (CAST(`Free Meal Count (K-12)` AS REAL) / `Enrollment (K-12)`) DESC LIMIT 1",
  "difficulty": "simple"
},
{
  "question_id": 1,
  "db_id": "california_schools",
  "question": "Please list the lowest three eligible free rates for students aged 5-17 in continuation schools.",
  "evidence": "Eligible free rates for students aged 5-17 = `Free Meal Count (Ages 5-17)` / `Enrollment (Ages 5-17)`",
  "SQL": "SELECT `Free Meal Count (Ages 5-17)` / `Enrollment (Ages 5-17)` FROM frpm WHERE `Educational Option Type` = 'Continuation School' AND `Free Meal Count (Ages 5-17)` / `Enrollment (Ages 5-17)` IS NOT NULL ORDER BY `Free Meal Count (Ages 5-17)` / `Enrollment (Ages 5-17)` ASC LIMIT 3",
  "difficulty": "moderate"
},
{
```


Text-to-SQL Task

Given a **natural language question** and a **target database schema**, generate an SQL query that can be executed correctly against the target database.

https://bird-bench.github.io/

Bird Benchmark

12,751 unique question-SQL pairs

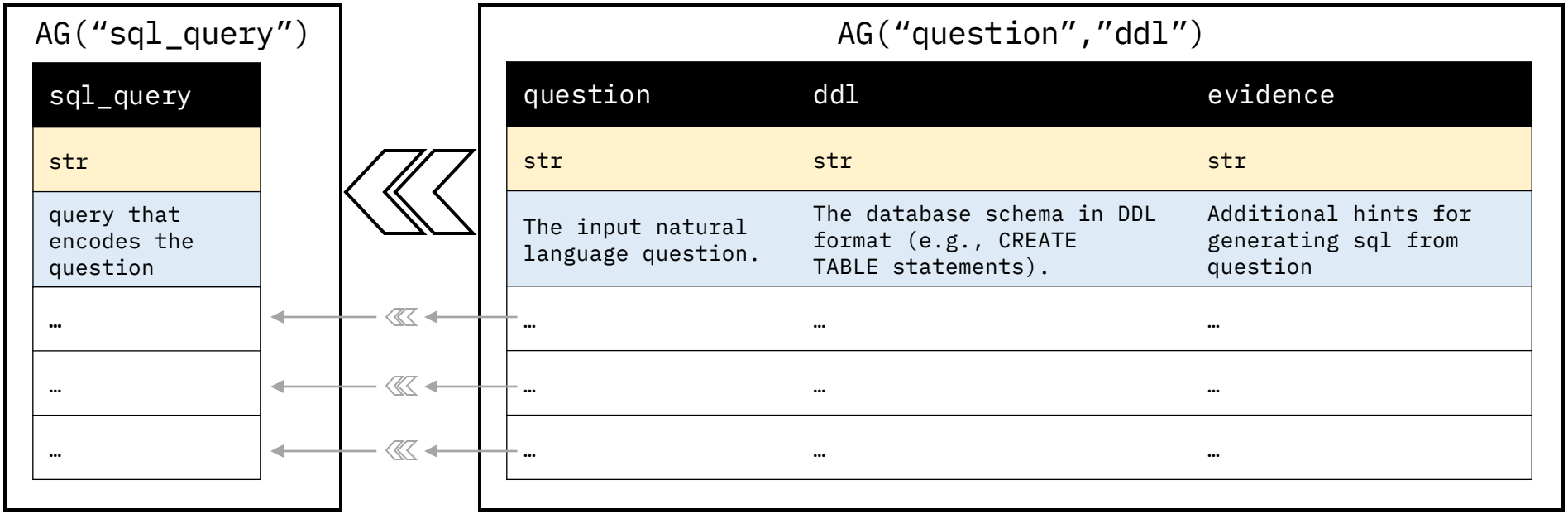
95 big databases

33.4 GB total size

37 professional domains

Agentics Baseline: Transduction

```
class Text2SQLTask(BaseModel):
    question: str = Field(description="The input natural language question.")
    ddl: str = Field(description="The database schema in DDL format (e.g., CREATE TABLE
        statements).")
    sql_query: Optional[str] = Field(description="The SQL query to be generated from the
        question.")
    execution_result: Optional[List[Dict[str, str]]] = Field(description="The resulting table
         from executing the SQL query.")
```



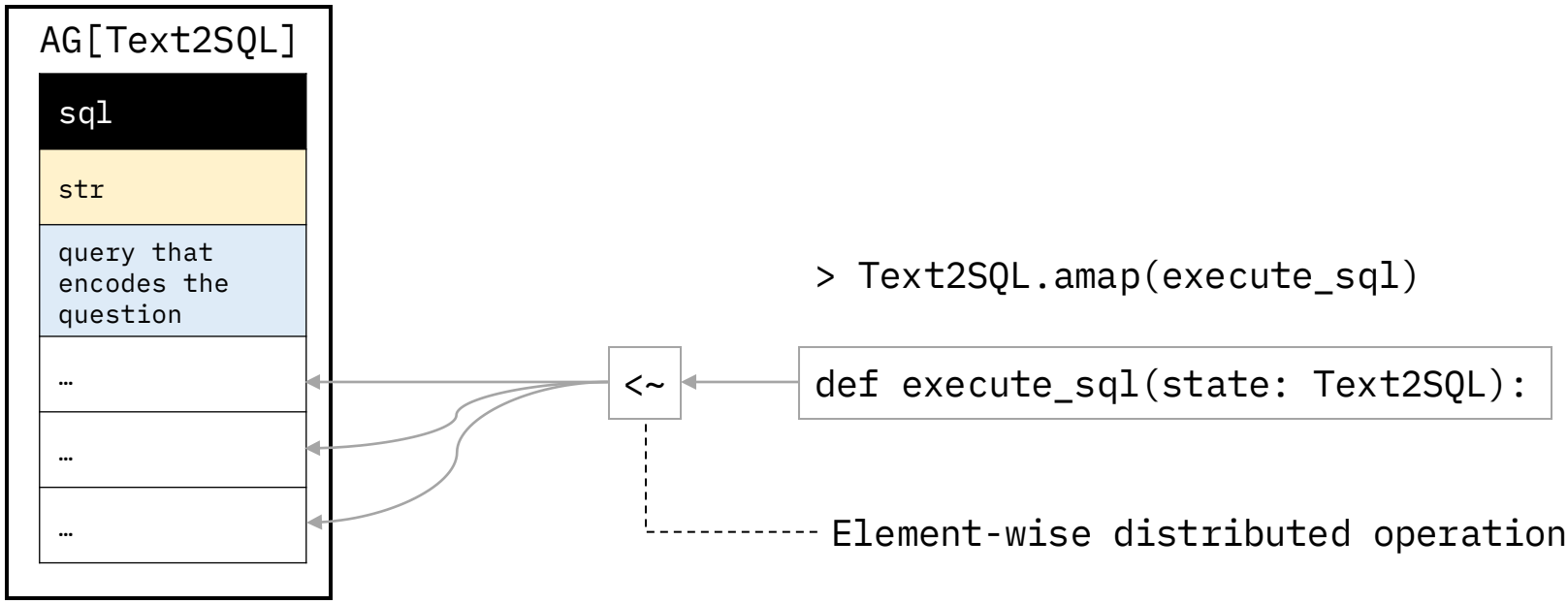
Example DDL

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Department VARCHAR(50),
    HireDate DATE);
```

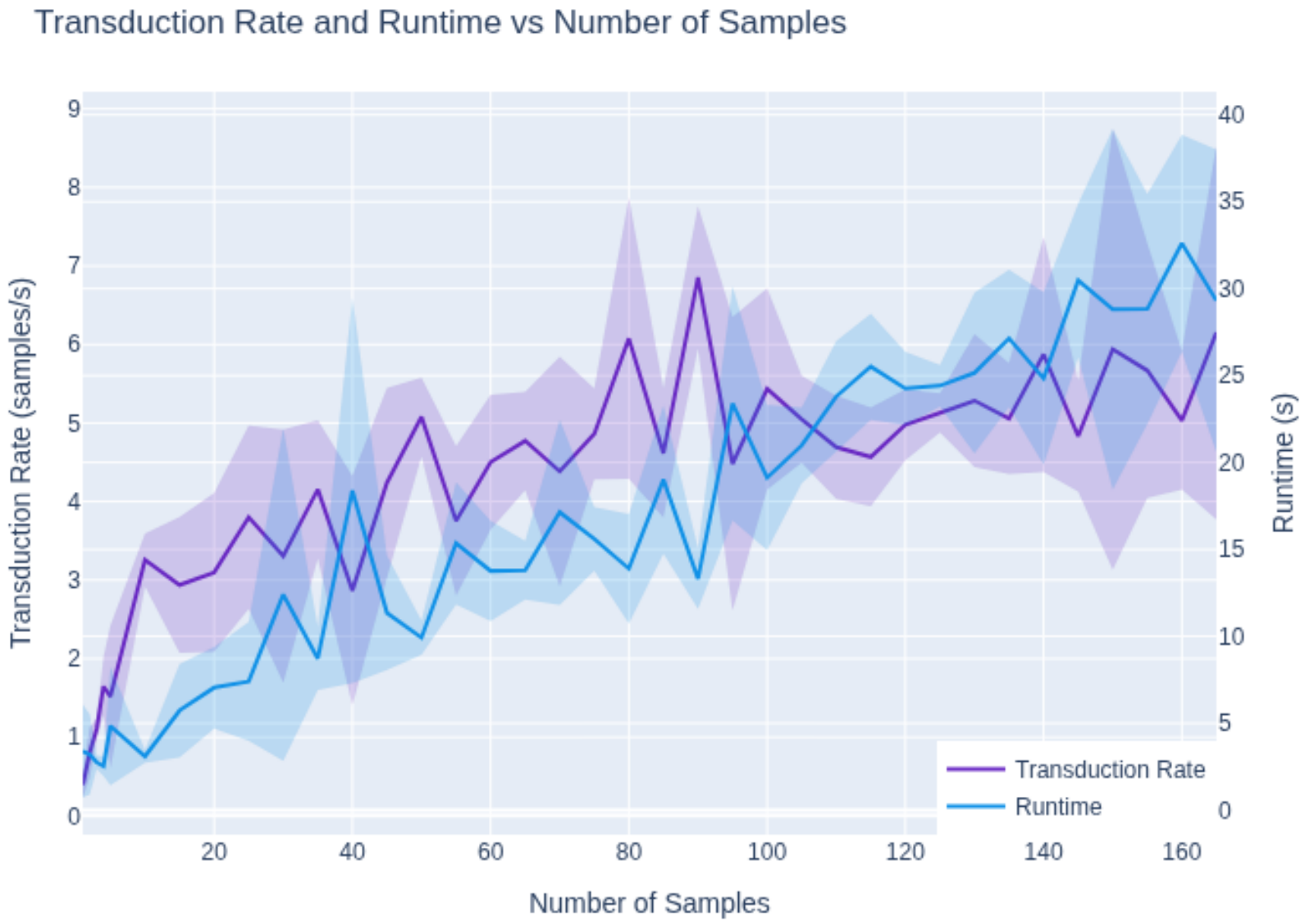
Agentics Baseline: aMap

```
SELECT name, grade, age
FROM students
WHERE age > 14 AND grade > 85
ORDER BY grade DESC;
```

name	grade	age
Bob	92	15
David	85	15



Evaluation via `amap`


Transduction Rate and Runtime vs Number of Samples
Transduction Rate (samples/s)
Runtime (s)
Number of Samples
Transduction Rate
Runtime
0
1
2
3
4
5
6
7
8
9
0
5
10
15
20
25
30
35
40
20
40
60
80
100
120
140
160

Speed

Outline

- Scaling out GenAI with State Graphs: Langgraph
- Scaling out GenAI with MapReduce: Agentics
- Example Applications
  - Multiple Choice QA (Decision Making)
  - Text2SQL
- Hands on: explaining market volatility

Students Projects

Agostino Capponi, Alfio Gliozzo

Using Git for Project Management

- **Create a Fork**
  - Each team should create a fork of the repository and name it after their project:
  - 👉 Example: student_project1
- **Maintain a Wiki Page**
  - Document the project details in the repository wiki:
  - 👉 Project Description Wiki
- **Manage Tasks with GitHub Projects**
  - Use GitHub Projects to define, assign, and track tasks:
  - 👉 Project Board

References

- **LangChain / LangGraph**
  - LangGraph Official Site
  - AI Agents in LangGraph – DeepLearning.AI Short Course
- **Dean, J., & Ghemawat, S.** (2008). *MapReduce: Simplified data processing on large clusters*. *Communications of the ACM, 51*(1), 107–113. https://doi.org/10.1145/1327452.1327492
- **Gliozzo, A., Khan, N., Constantinides, C., Mihindukulasooriya, N., Defosse, N., & Lee, J.** (2025). *Transduction is All You Need for Structured Data Workflows*. *arXiv preprint* arXiv:2508.15610.