DISCUSSION POINT · CIEL, NSDI 2011 · JOSHUA FEHN

From fixed graphs to graphs that follow the data

2004
MapReduce
Fixed two-stage graph

2007
Dryad
Any DAG, fixed before the run

2011
CIEL
Tasks spawn tasks: the graph grows at run time

2012
Spark
Driver loop, in-memory data, lineage

2018
Ray
“Dynamic task graph, similar to CIEL”

Today
AI workloads
RL, LLM serving, agent workflows

> The open trade-off: a **static** graph can be optimised before it runs; a **dynamic** graph can react to the data while it runs.

RELEVANCE TODAY

CIEL’s ideas, then and now

1 **2011:** the graph is built while the job runs

> Today: PyTorch builds its graph on the fly; torch.compile recaptures static parts for speed

2 **2011:** its own language, Skywriting

> Today: Ray runs CIEL’s task model in plain Python

3 **2011:** one master dispatches every task

> Today: one orchestrator agent plans every sub-agent

4 **2011:** recovery by re-running deterministic tasks

> Today: agent steps are not deterministic, so durable workflows record results instead

OPEN QUESTIONS

Four discussion points

1 **Static or dynamic?**

What do we lose when the graph is only known at run time? Where should the line be today?

2 **Why did Spark win?**

Spark’s plain Python loops beat Skywriting; Ray revived CIEL’s model in 2018. Right idea, wrong time? Do coding agents change that?

3 **One coordinator or many?**

CIEL’s single master fell behind at fine granularity. Is one orchestrator agent the same bottleneck?

4 **Fault tolerance for agents?**

CIEL recovers by re-running deterministic tasks. LLM calls are not deterministic and have side effects. What replaces replay?