# CIEL: a universal execution engine for distributed data-flow computing

Derek Murray, Malte Schwarzkopf, Christopher Smowton,
Steven Smith, Anil Madhavapeddy and Steven Hand

Presented by: Deniz Alkan (da626)

22 October 2025

# Background & Context

- Goal: Good distributed computing
- Problem: What is “good”?

# Previous Work: MapReduce

MapReduce: bipartite graph structure to

1. map inputs into intermediate key, value pairs
2. reduction on the values of each key

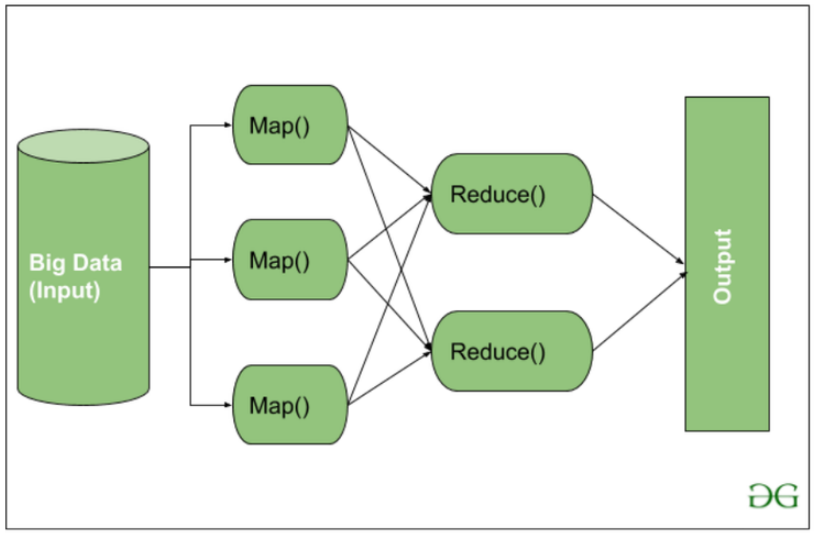


Figure: MapReduce architecture [1]

- Simple, makes sense
- Easily parallelisable
- transparent scaling & fault tolerance

[1] https://www.geeksforgeeks.org/c-sharp/map-reduce-in-hadoop

# Previous Work: Dryad/LINQ

Dryad/LINQ: directed acyclic graph (DAG) structure:

1. inputs and outputs flow between tasks
2. statically generated task ordering/scheduling

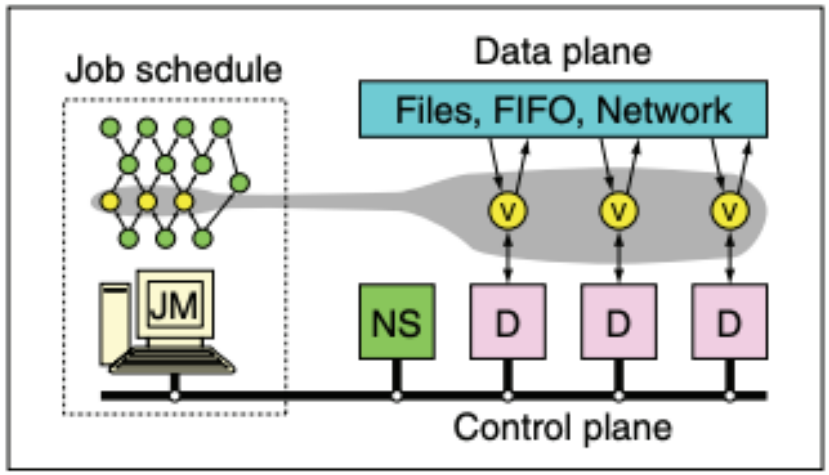


**Figure 1: The Dryad system organization.** The job manager (JM) consults the name server (NS) to discover the list of available computers. It maintains the job graph and schedules running vertices (V) as computers become available using the daemon (D) as a proxy. Vertices exchange data through files, TCP pipes, or shared-memory channels. The shaded bar indicates the vertices in the job that are currently running.

Figure: Dryad architecture[1]

- More complex, allowing for complicated relationships and operations
- streaming

[1] Isard et al. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks

# Previous Work: Pregel & BSP

BSP (Bulk Synchronous Parallel) operates in "supersteps":

1. compute in parallel
2. communicate – exchange data
3. synchronise (barrier) - vote to finish superstep when individually done

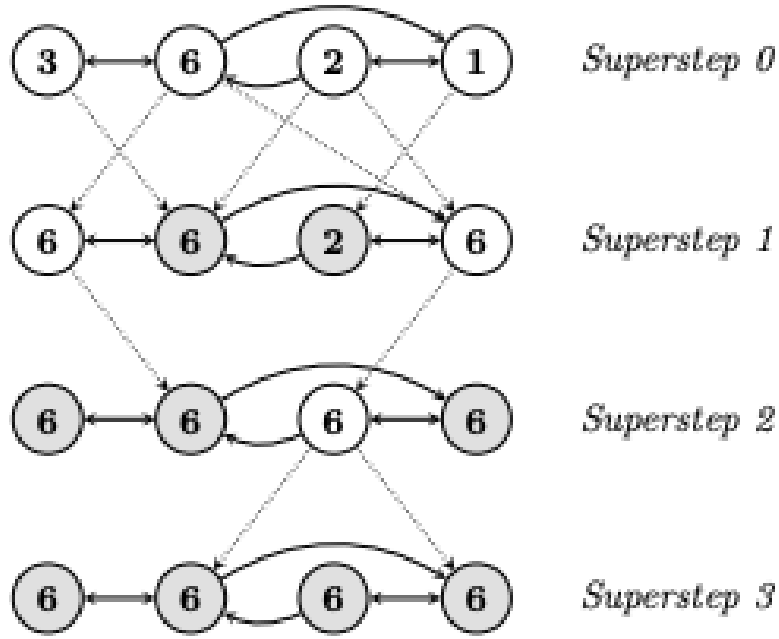


Figure: Pregel implementing BSP[1]

- dynamic control flow
- no online task generation (as with others)

[1] Malewicz et al. Pregel: A System for Large-Scale Graph Processing

# Previous Work – Notable Mentions: Iterative MR, Piccolo

- Iterative MR extends MapReduce to include data-driven control flow
- Piccolo reduces MapReduce into a single layer of "kernel" operations and implements a checkpoint system for fault tolerance, but suffers from non-transparent scaling.

# Identifying the Problem

- We need dynamic control flow, data locality, transparent scaling and fault tolerance
- Pregel has all of these, what is the issue?
- What about recursive algorithms?
- Fifth requirement: flexibility!
- CIEL's goal: unify all five requirements AND preserve transparent automatic distribution $\rightarrow$ evolution of predecessors

# CIEL's Ideas and Solution

- objects, references, tasks
- dynamic task graphs:
  1. tasks publish references or further tasks with reference to originally scheduled output
  2. child tasks depend on non-empty addresses / outputs of pre-existing tasks ⇒ prevent cycles
- lazy ("backward") evaluation selected over eager ("forward") evaluation ⇒ better memoisation & FT
- master-worker structure w/ FT & deterministic naming

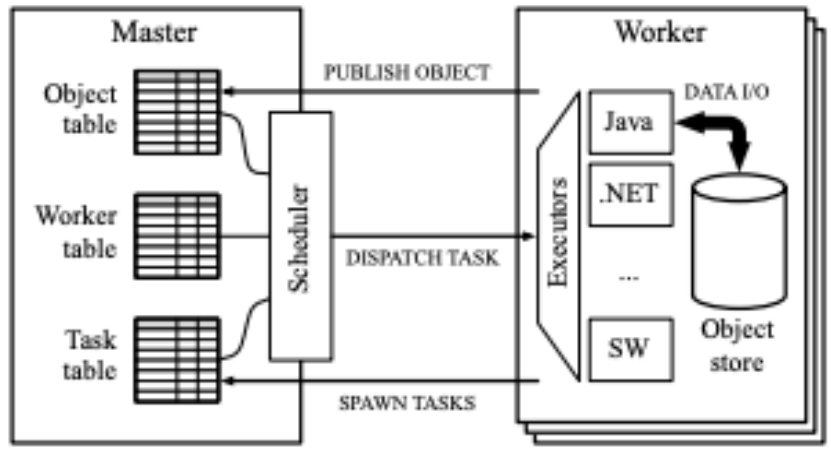


Figure: CIEL master and workers [1]

[1] Murray et al. CIEL: a universal execution engine for distributed data-flow computing

# CIEL & Skywriting

- purpose built, Turing-complete language
- runs on CIEL
- abtracts away the dereference operator's continuation-passing style - which is claimed to be the **main** selling point of Skywriting
- what about other languages?

# Evaluation

- Evaluation against Hadoop (MapReduce implementation) in Grep and k-means - improved results (not enough)
- non-comparative dynamic programming evaluation: disappointingly unscalable results
- why?

# Opinions

- The evaluated results show worse-than-expected performance, CIEL is not as innovative as it advertises: limited scope
- This is demonstrated by the number of citations of CIEL: 413
- This seems little... at least until 2023 CIEL was the only system that supported dynamic task generation, as surveyed by Margara et al.[1]
- How about TensorFlow? 13929 why?
- TF paper[2] doesn't have many bad things to say about CIEL – just grouped in with the others: TF closer to Naiad w/ static, cyclic data flow. CIEL is just a building block / case study
- Yu et al.[3] deduce that CIEL's dynamic task generation causes each task to act as a black box. This curbs optimisation, explainability and granularity.
- Skywriting: barrier to entry – mainstream language instead?

[1] Margara et al. A Model and Survey of Distributed Data-Intensive Systems

[2] Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems

[3] Yu et al. Dynamic Control Flow in Large-Scale Machine Learning

# Questions/Suggestions

- Why hasn't CIEL addressed the problem of stragglers?
- Clearly motivated by MapReduce, its potential and drawbacks
- MapReduce reported 44% increase in runtime due to stragglers on sorting[4]
- CIEL may have more "granular" tasks than a huge reduce function, e.g.
- Not always the case... e.g. we could roughly simulate MapReduce with CIEL

[4] Dean and Ghemawat. MapReduce: Simplified Data Processing on Large Clusters

# Final Remarks

- Unfortunately I can't just say very good just because it was developed in Cambridge, and CIEL sounds like CL (Computer Lab).
- I will, however, give bonus points for using the name CIEL and Skywriting together, as "ciel" means "sky" in French...