CIEL : A Universal Execution Engine

G Roshan Lal

What is CIEL?

- Computation Engine
- Distributed Data-Flow
- Universal

- Other Distributed Execution Engines….Data-Flow
  - MapReduce…….Bi-Partite Graph
  - Dryad……..DAG

Why CIEL?

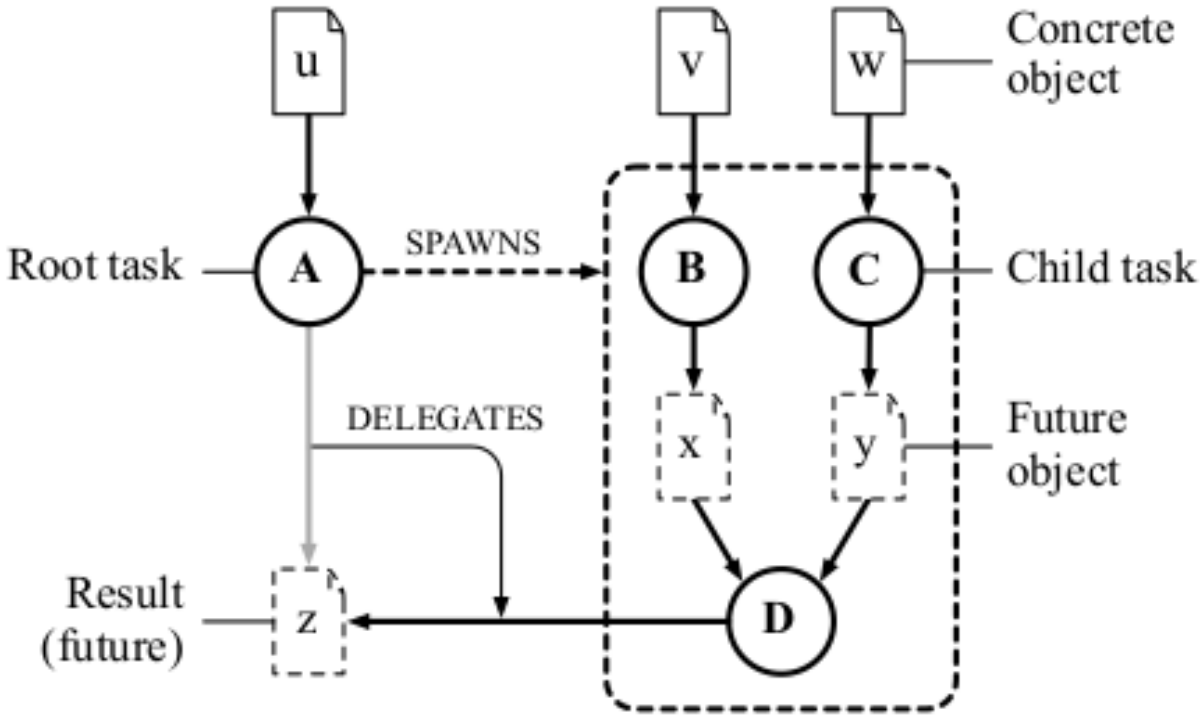
- MapReduce/Dryad is

- Good For:
  - Batch-Oriented
  - Eg. Info Retrieval
  - High Throughput

- Bad For:
  - Iterative
  - Eg. ML Training
  - Low Latency, chained events

How CIEL works?

- Dynamic Task Graphs



(a) Dynamic task graph

How CIEL works?

- Task / Object Tables

Task ID	Dependencies	Expected outputs
A	{ u }	~~z~~
B	{ v }	x
C	{ w }	y
D	{ x, y }	z

Object ID	Produced by	Locations
u	–	{ host19, host85 }
v	–	{ host21, host23 }
w	–	{ host22, host57 }
x	**B**	∅
y	**C**	∅
z	~~**A**~~ **D**	∅

(b) Task and object tables

How CIEL works?

- Objects : Named collection of Bytes
- References : Obj Name, Physical Location (maybe empty….not yet created)
- Tasks
  - Publish output objects (or)
  - Spawn new Tasks depending on output objects
  - Dynamic DAG…. Prevents Deadlock
  - Lazy Evaluation… Easier fault Tolerance

CIEL Architecture

- Single Master
  - Keeps record of obj and task tables
  - Dispatches tasks to workers
- Multiple Workers
  - HeartBeat messages for availability to Master
  - Update Master with Spawn/Publish

CIEL Fault Tolerance

- Workers:
  - HeartBeat from Workers/ Response to Master
  - Re-schedule
- Master:
  - Persistent Logging of Tables

CIEL Optimizations

- Long tails of recursion:
  - Assume deterministic behaviour of tasks
  - Memoization of output for given input
- Persistent logging of (Single) Master:
  - Secondary Masters
  - Reconstruct from Workers
- Skywriting:
  - Scripting Language for CIEL

Conclusion

- CIEL perfroms well on all types of loads
- MapReduce Types: grep
- Iterative Types: k-means
- Compute Intensive Types: Smith-Waterman