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

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

Bogdan-Alexandru Matican

University of Cambridge

February 5, 2013
Table of contents

1. Research questions

2. Design
   - CIEL
   - Skywriting

3. Technicalities

4. Conclusion
Main considerations

- distributed data-flow computing
- task dependencies
- dynamic coordination

Bonus: transparency (fault tolerance, scaling, locality)

**Figure**: Features of distributed execution engines.
Introduction

The system is primarily focused around the following:

- **Data** objects and references to them
- **Processing** tasks (input and output references)
- **Coordination** dynamic task graph
Managing the graph

Two main rules for dependencies:

- **Input** depend on concrete or future references
- **Output** publish reference **OR** spawn child

Two main evaluation styles:

- **Eager** start with concrete tasks and continue
- **Lazy** start from root moving recursively down
Example state

Figure: A CIEL job example.
Architecture

Figure: Cluster architecture
Introduction

A couple of important primitives:

- **spawn** – parallel task
- **exec** – synchronous executor
- **dereference** – load reference in context
Handling tasks

Figure: Task creation example.
CIEL: a universal execution engine for distributed data-flow computing

- Technicalities

Details

- dereferencing – data / coordination space
- naming and memoisation
- fault tolerance (client / worker / master)
- streaming
Contributions

- system with broader computational model
- dynamic task dependency handling
- transparent distribution and scheduling
Critique and questions

- is Skywriting as a language necessary?
- worker fault tolerance – replication?
- deterministic, terminating computation?
- homogenous machines in cluster?