Compiling and Linking Workflows

Condor Week 2012

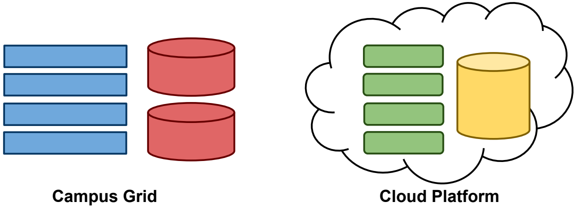
Peter Bui

Cooperative Computing Lab
University of Notre Dame, IN, USA

May 2, 2012

Programming Distributed Systems is Hard

Distributed Systems



Research Challenge

How do we enable both novice and expert users to take advantage of distributed computing resources? (*Particularly for data-intensive scientific applications*).

Cloud Computing Approach - Abstractions

Structured way of combining small executables into parallel graphs that can be scaled up to large sizes.

Examples

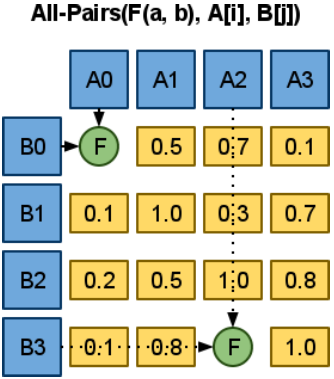
All-Pairs, Wavefront, Map-Reduce

Advantages

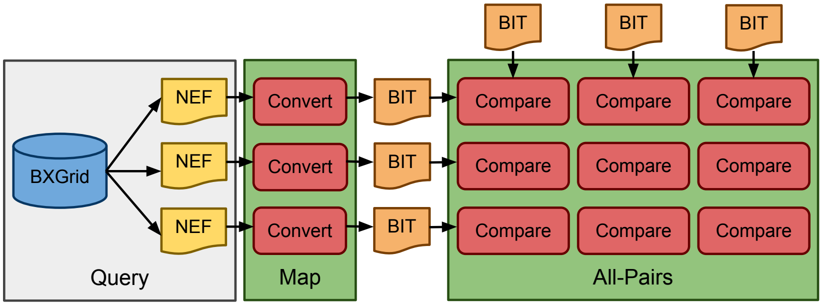
- Simple programming interface.
- Hides details of distributed system.

Disadvantages

- Only addresses one phase of computation.
- Difficult to implement large sophisticated workflows.



Biometrics Experiment



1. **Query:** Select and extract data from scientific repository.
2. **Transcode:** Convert image data to new format suitable for analysis.
3. **Comparison:** Perform All-Pairs computation on new image data.

Grid Computing Approach - Workflows

Organize computation as a directed-acyclic-graph (DAG).

Examples

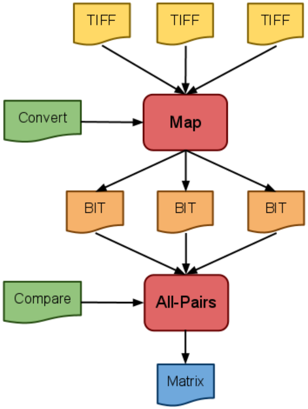
Pegasus, DAGMan, Dryad, Makeflow

Advantages

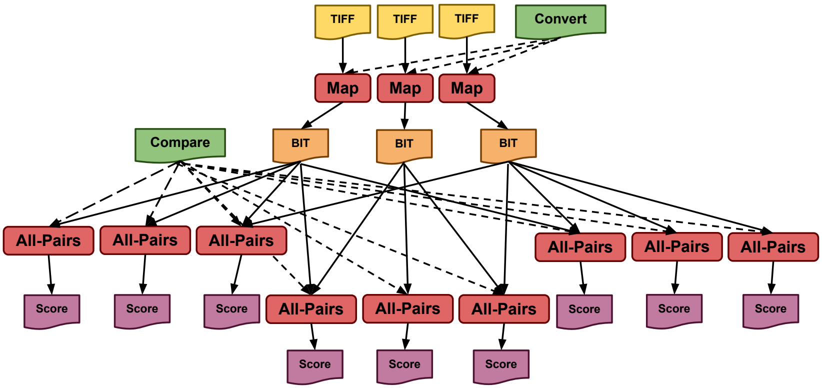
- Exploit natural parallelism.
- Program large applications consisting of multiple phases.
- Embed/implement abstractions as part of DAG.

Disadvantages

- Tedious, difficult to construct DAGs.
- Too low level.



Biometrics Experiment (DAG)



- **Map:** $O(n)$ tasks.
- **All-Pairs:** $O(n^2)$ tasks.
- **Large workflows require many nodes**.

Proposed Approach - Compiler

Proposition

We need a **compiler for distributed workflows** will combines the programming ideas from both grid and cloud computing.

Observations

1. **DAGs** are the **assembly language** of distributed computing.
   Provide mechanism for constructing and executing large distributed applications.
2. **Abstractions** are the **SIMD instructions**.
   Provide powerful compact way to express a common pattern of computation.

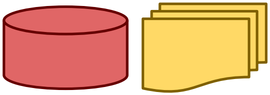
Compiler Overview

Weaver is a high-level compiler framework that allows users to construct distributed workflows.

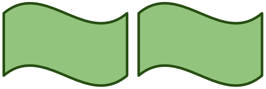
Unique Features

- Built on top of **Python** programming language.
- Enables users to combine abstractions to construct workflows.
- Applies various **compiler techniques** to workflow construction.
- Includes additional utilities such as **linkers** and profilers to provide a complete programming toolchain.

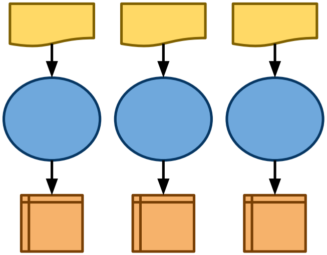
Programming Model



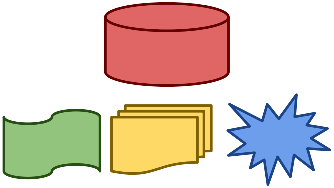
Datasets



Functions



Abstractions

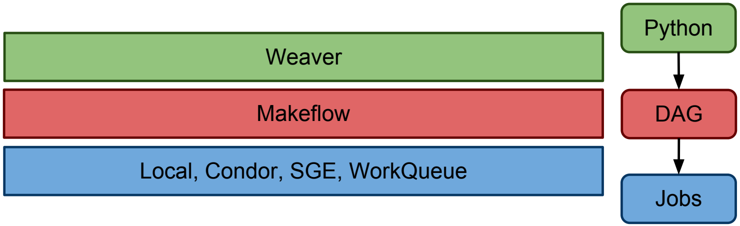


Nests

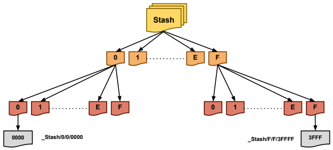
Biometrics Experiment

```
db      = MySQLDataset('db', 'biometrics', 'irises')
irises  = Query(db, db.c.state == 'Enrolled',
                    Or(db.c.color == 'Blue',
                       db.c.color == 'Green'))

convert = ParseFunction(
            'convert_iris_to_template {IN} {OUT}')
compare = ParseFunction(
            'compare_iris_templates {IN} > {OUT}')

bits    = Map(convert, irises, '{BASE_WOEXT}.bit')
results = AllPairs(compare, bits, bits)
table   = Merge(results, 'table.txt')
```

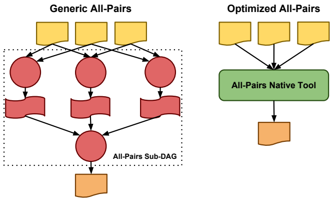
Software Stack



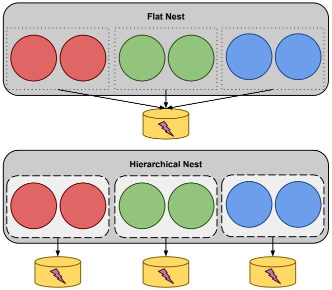
Optimizations



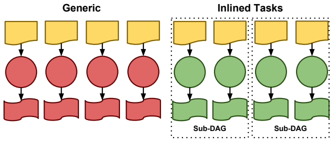
(a) Intermediate Files



(b) Instruction Selection

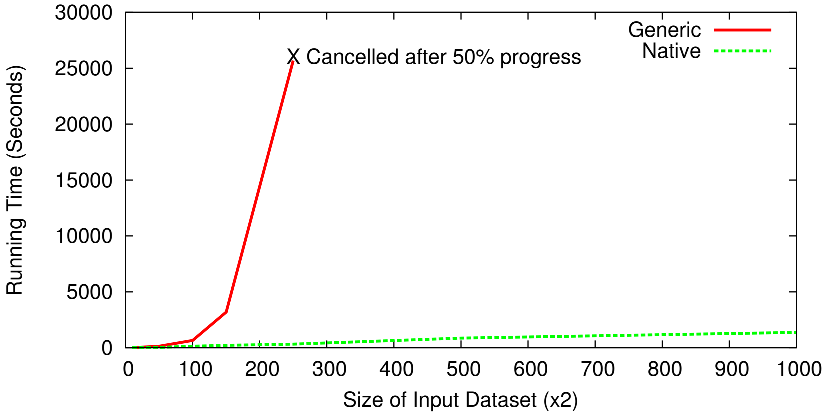


(c) Hierarchical Workflows

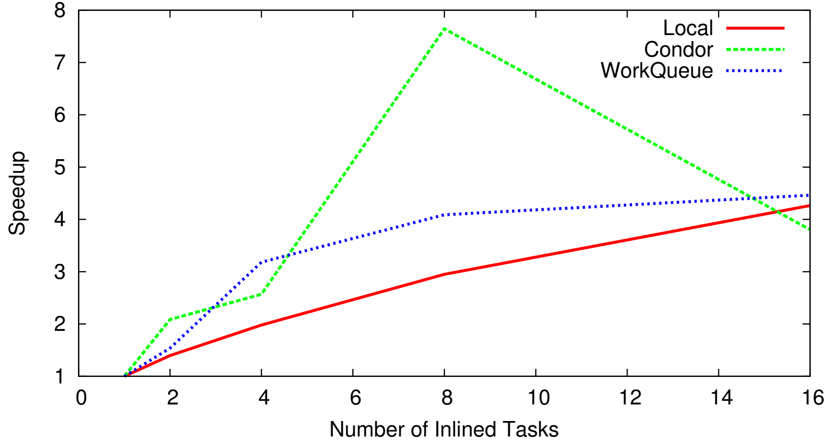


(d) Inlined Tasks

Instruction Selection



Inlined Tasks



Toolchain

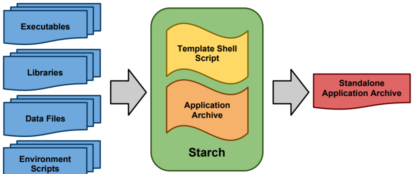
Linking

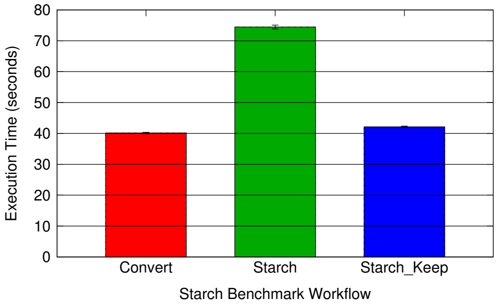
- **Application Linker**: Package applications for portable distribution and execution.
- **Workflow Linker**: Intelligently modify paths in DAG.

Profiling

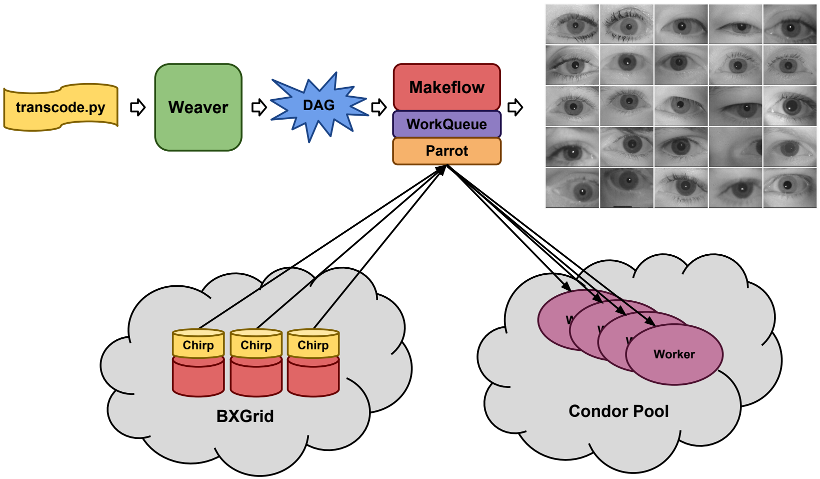
- **Analyze**: Export provenance information into a variety of formats.
- **Monitor**: Report workflow execution information in a user-friendly manner.
- **Report**: Provide statistics and summaries of workflow.

Application Linker





Transcoding Workflow



Summary

Having access distributed computing resources is great, but we must provide tools and support for both novice and expert users to take advantage of these systems.

Weaver

Programming toolchain that includes a **compiler** for translating workflows written in Python DSL into Makeflow DAGs, **linkers** for packaging components and entire workflows, **profilers** for analyzing and monitoring workflows.

CCTools

- **Makeflow**: Workflow manager for parallel and distributed systems.
- **WorkQueue**: Light-weight master-worker framework.
- **Chirp**: Unpriviledged network personal filesystem.
- **Parrot**: Transparent adapter for remote filesystems.

Questions?

CCTools

`http://cse.nd.edu/~ccl/software`

Collection of distributed computing software.

Weaver

`http://bitbucket.org/pbui/weaver`

Distributed Workflow compiler for Makeflow.

python-cctools

`http://bitbucket.org/pbui/python-cctools`

Collection of CCTools utilities in Python.