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

Campus Grid

Cloud Platform

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.
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.
- **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
1 db = MySQLDataset('db', 'biometrics', 'irises')
2 irises = Query(db, db.c.state == 'Enrolled',
3 Or(db.c.color == 'Blue',
   db.c.color == 'Green'))

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

11 bits = Map(convert, irises, '{BASE_WOEXT}.bit')
12 results = AllPairs(compare, bits, bits)
13 table = Merge(results, 'table.txt')
Software Stack

Weaver

Makeflow

Local, Condor, SGE, WorkQueue

Python

DAG

Jobs
Optimizations

(a) Intermediate Files

(b) Instruction Selection

(c) Hierarchical Workflows

(d) Inlined Tasks
Instruction Selection

![Graph showing running time vs. size of input dataset for Generic and Native implementations. The graph indicates that the Generic implementation has a significantly higher running time, especially as the size of the input dataset increases. There is a note that says 'X Cancelled after 50% progress.'](image)
Inlined Tasks

![Graph showing the speedup versus the number of inlined tasks for Local, Condor, and WorkQueue.]

- **Local** (red line)
- **Condor** (green dashed line)
- **WorkQueue** (blue dotted line)

The graph illustrates the speedup achieved with different numbers of inlined tasks for each of the three systems. The speedup is calculated as the ratio of execution time with inlined tasks to the execution time without inlining.
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

Executables
Libraries
Data Files
Environment Scripts
Starch
Template Shell Script
Application Archive
Standalone Application Archive

Execution Time (seconds)

Starch Benchmark Workflow

Convert
Starch
Starch_Keep

Execution Time (seconds)
Transcoding Workflow

- transcode.py
- Weaver
- DAG
- Makeflow
- WorkQueue
- Parrot
- BXGrid
- Chirp Chirp Chirp
- Condor Pool
- Worker
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**: Unprivileged 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.*