Scaling Up Bioinformatic Workflows with Dynamic Job Expansion: A Case Study Using Galaxy and Makeflow

Nicholas Hazekamp, Joseph Sarro, Olivia Choudhury, Sandra Gesing, Scott Emrich and Douglas Thain

Cooperative Computing Lab: http://ccl.cse.nd.edu

University of Notre Dame
Galaxy is an open source, web-based platform for data-intensive biomedical research. If you are new to Galaxy, start here or consult our help resources.

Public Galaxy Servers and still counting

This history is empty. You can load your own data or get data from an external source.
Workflows in Galaxy
Simple Workflow in Galaxy

Problem: As Size increases so does Time
Workflow with Parallelism added in Galaxy

Problem: Tools must be updated every change in Parallelism/Relies on Scientist
Workflow Dynamically Expanded behind Galaxy
User-System Interaction

Execution Engine

Tasks run

SANDBOX

JE

DAG

MF

File location, linking, and storing.

Tool creates sandbox.

Run From Portal

WWW

Repository

Task

Task

Task

Execution Engine

User
Makeflow

• Task Structure
INPUTS : OUTPUTS
COMMAND

• Directed Acyclic Graph (DAG)

• Programatically Generated
Work Queue

- Galaxy
- Makeflow
- Work Queue
- Local Files and Programs
- Local Cluster
- XSEDE Resource
- Condor Pool
- SGE Cluster
Work Queue

- Galaxy
- Makeflow
- Work Queue

Local Files and Programs

Local Cluster

XSEDE Resource

Condor Pool

SGE Cluster
Work Queue

Galaxy

Makeflow

Work Queue

Local Files and Programs

Local Cluster

XSEDE Resource

Condor Pool

SGE Cluster
Job Sandbox – Inputs and Outputs linked
Job Sandbox – Program creates unmanaged files
Job Sandbox – Program creates unmanaged files
Job Sandbox – Program creates unmanaged files
Job Sandbox – Log file creation for cleanup
Managing Environmental Expectations

Environment Variables
- PYTHON_PATH
- DB_LOC

Execution Platforms
- JAVA
- Python
- Perl

Path Programs
- BWA
- GATK
- Add RGs
- Sam->Bam

Task Specific
- Inputs
- Split
- Concat
Small Scale Run

Query: 600MB  Ref: 36MB
Full Scale Run

Query: 32GB   Ref: 36MB
Performance in Real-Life

• 100+ Different runs through Workflow

• Utilizing 500+ Cores with heavy load

• Data sets ranging from >1GB to 50GB+
Real Usage Concurrency Comparison

![Concurrency Achieved vs Concurrency Expressed](chart.png)
Conclusions

• Using Dynamic Job Expansion we were able to scale up a workflow without requiring the huge amount of time to process it

• Found viable solutions for:
  • Using Work Queue we utilized 100s of cores from a Condor Pool
  • Cleaning Sandbox using knowledge of intermediates and logging
  • Explored methods to transmit needed environments such as executables and Java

• 61.5X speed-up on 32 GB dataset utilizing these methods
Acknowledgements

• CCL
  • http://ccl.cse.nd.edu

• University of Notre Dame

• NSF
  • Grant OCI-1148330
Thank You – Questions?

CCL software including CCTools: http://ccl.cse.nd.edu

For more information contact:
Nick Hazekamp: nhazekam@nd.edu