Directed Acyclic Graph Scheduling with User Defined Validation and Assimilation

David Coss

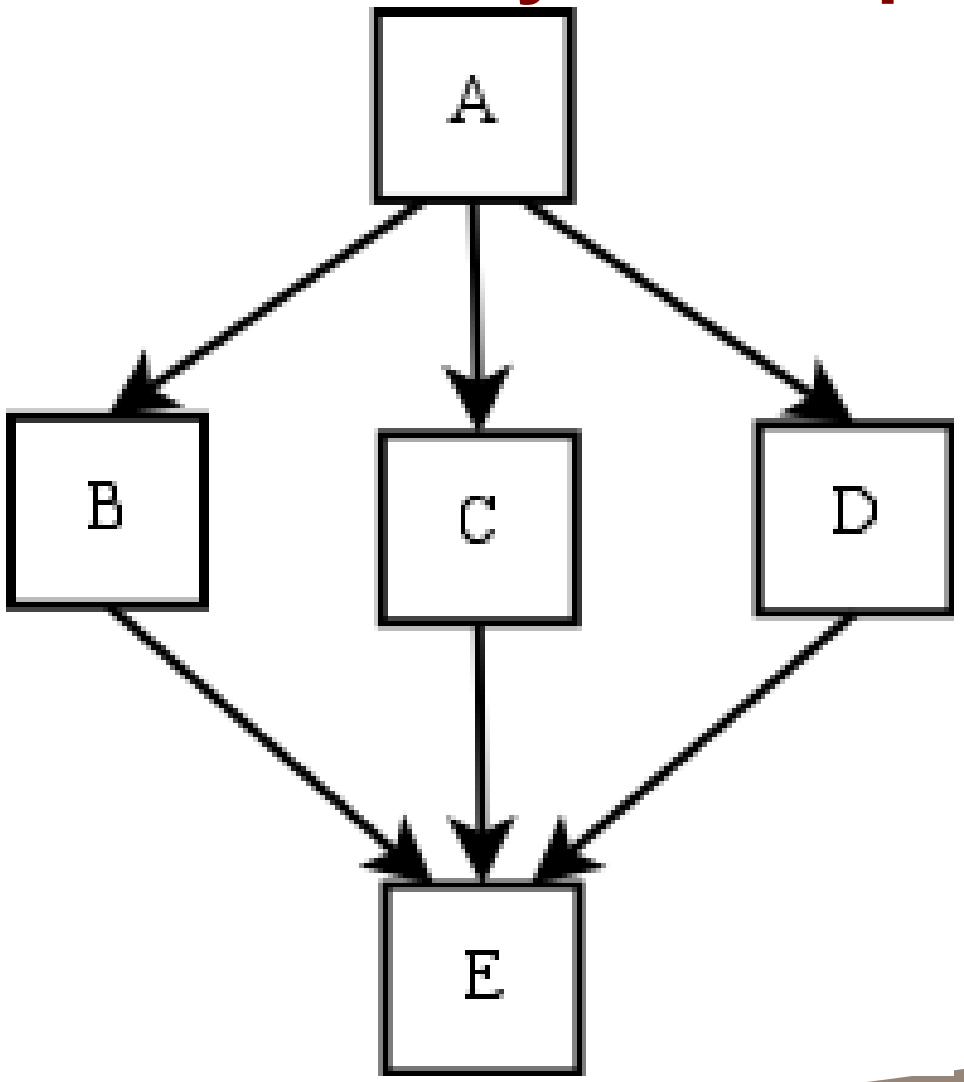
BOINC Workshop

27 Sept 2012

Outline

- Goals and Definitions
- Functionality
- Use Case
- To do

Directed Acyclic Graph



Goals

- Reusable method of parallelization
- Provide dynamic validation and assimilation
- Add Server-Side Job Functionality
- Provide Fault Tolerance/Error Handling

Goals

- Reusable method of parallelization
- Provide dynamic validation and assimilation
- Add Server-Side Job Functionality
- Provide Fault Tolerance/Error Handling

gsub

- Commands to be run written like shell script
- *gsub* parses script
- Commands are mapped to user-defined routines in Parser Dictionary
  - Parser routines create list of Process Objects
- *gsub* creates graph (DAG) of interdependent processes
- Jobs are started (create_work)

gsub

- Run as user
  - DAG creation code is completely user customizable
- Calls create_work
  - Requires no BOINC code to follow during development
- Parsing functions are defined in ~/.boincdag

Validator

- Adds layer to BOINC API
  - init_result, compare_results and clean_result are embedded python functions to setup and run user defined routines based on Command → Routine Map
  - Adds BoincResult Python class

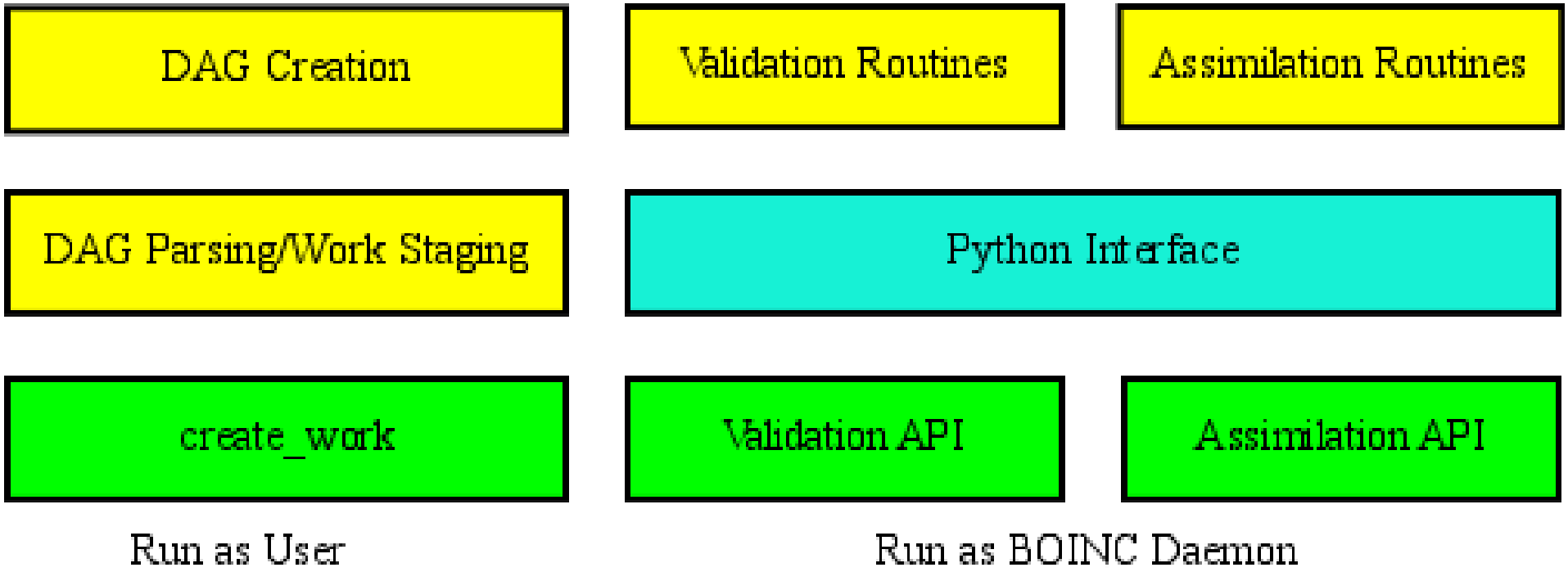
Assimilator

- assimilate_handler
  - Embedded python function calls corresponding user defined code for application
  - Loads DAG object and starts child work units

Design

- Written in a combination of Pure and Embedded Python
  - Allows changes to validation routines without need to restart daemon
  - Adds flexibility in user-defined routines
  - Easier for users with less programming experience to create custom validation code

Design



Use Case

Microrna – DNA Binding Prediction

- Embarrassingly parallel
- Operates on segments of DNA
- Results of Segments are recombined
- End Result is Reformated for Database

Example

```
[dcoss@stjudeathome2 Y]$ cat job.sub
trident hsa_all_mirna-1.fasta chrY.fasta -brief -out chrY_mirna-1.out
trident hsa_all_mirna-2.fasta chrY.fasta -brief -out chrY_mirna-2.out
```

Example

```
[dcoss@stjudeathome2 Y]$ cat ~/.boincdag
parsers['trident'] = 'trident_segmenter.parse'
import trident_segmenter
```

Example

```
[dcoss@stjudeathome2 Y]$ python -m gsub --setup_only job.sub
Running trident_segmenter.parse(parser_args)
Running trident with miRNA hsa_all_mirna-1.fasta and DNA chrY.fasta
with flags -brief -out chrY_mirna-1.out
Created 566 dna files
Running trident_segmenter.parse(parser_args)
Running trident with miRNA hsa_all_mirna-2.fasta and DNA chrY.fasta
with flags -brief -out chrY_mirna-2.out
Created 566 dna files
Saved DAG as jobs.dag
```

Example

```
[dcoss@stjudeathome2 Y]$ python -m update_dag list|head
trident-113812189: trident
trident-542977945: trident
trident-471964252: trident
trident-542491246: trident
trident-134625408: trident
trident-209129499: trident
trident-436703746: trident
trident-048747455: trident
trident-414265868: trident
trident-853879748: trident
```

Example

```
[dcoss@stjudeathome2 depend_tests]$ python -m update_dag print
trident
State: RUNNING(2)
Input: hsa_all_mirna-1.fasta, chrMT.fasta, chrMT.fasta_hsa_all_mirna-1.fasta-job.xml (job.xml)
Output: step1.out
Workunit Name: trident-569267845
Workunit Template: tmppEvglY
Result Template: tmpLRDvpT
args: -brief -out step1.out

Children:
trident-736471555(trident)
```

Example

trident

State: CREATED(0)

Input: hsa_all_mirna-2.fasta, step1.out, step1.out_hsa_all_mirna-2.fasta-job.xml (job.xml)

Output: chrMT_150000-1_mirna-2.out

Workunit Name: trident-736471555

Workunit Template: tmpAo0Leb

Result Template: tmpXhYdAf

args: -brief -out chrMT_150000-1_mirna-2.out

Children:

Depends on: trident

Unfinished Dependencies

Process: trident-569267845

File: step1.out

TODO

- Server-Side Mechanism
  - Indicate that process is Python code or server program
- Explicit dependency
  - Currently only file based dependency
- Conditional Branches

Acknowledgments & Thanks

- Work supported financially by American Lebanese Syrian Associated Charities
- Use case is part of a project with Steve Paugh of St Jude