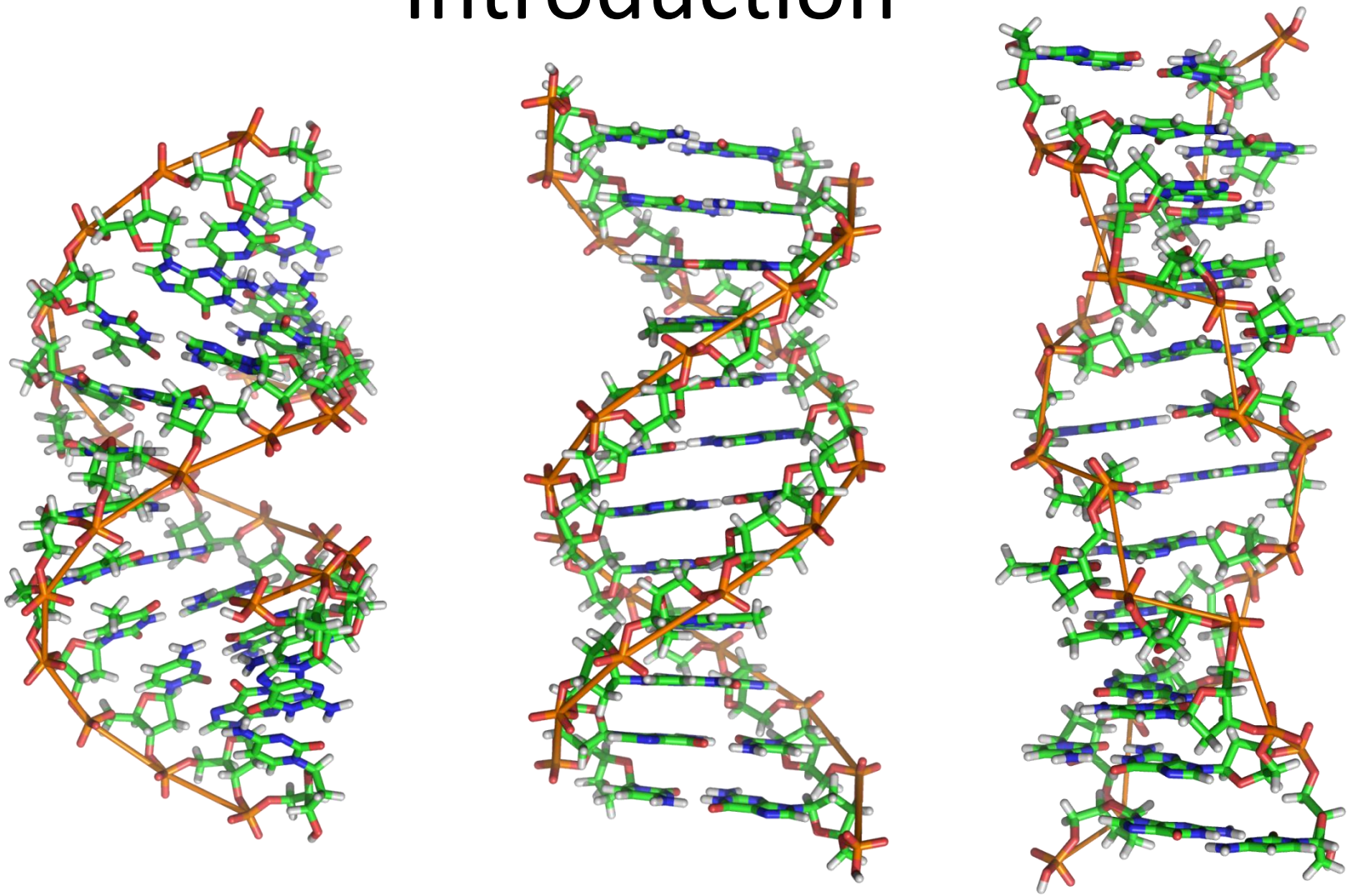
Taming Complex Bioinformatics Workflows with Weaver, Makeflow, and Starch

Andrew Thrasher, Rory Carmichael, Peter Bui, Li Yu, Douglas Thain, and Scott Emrich

November 14, 2010

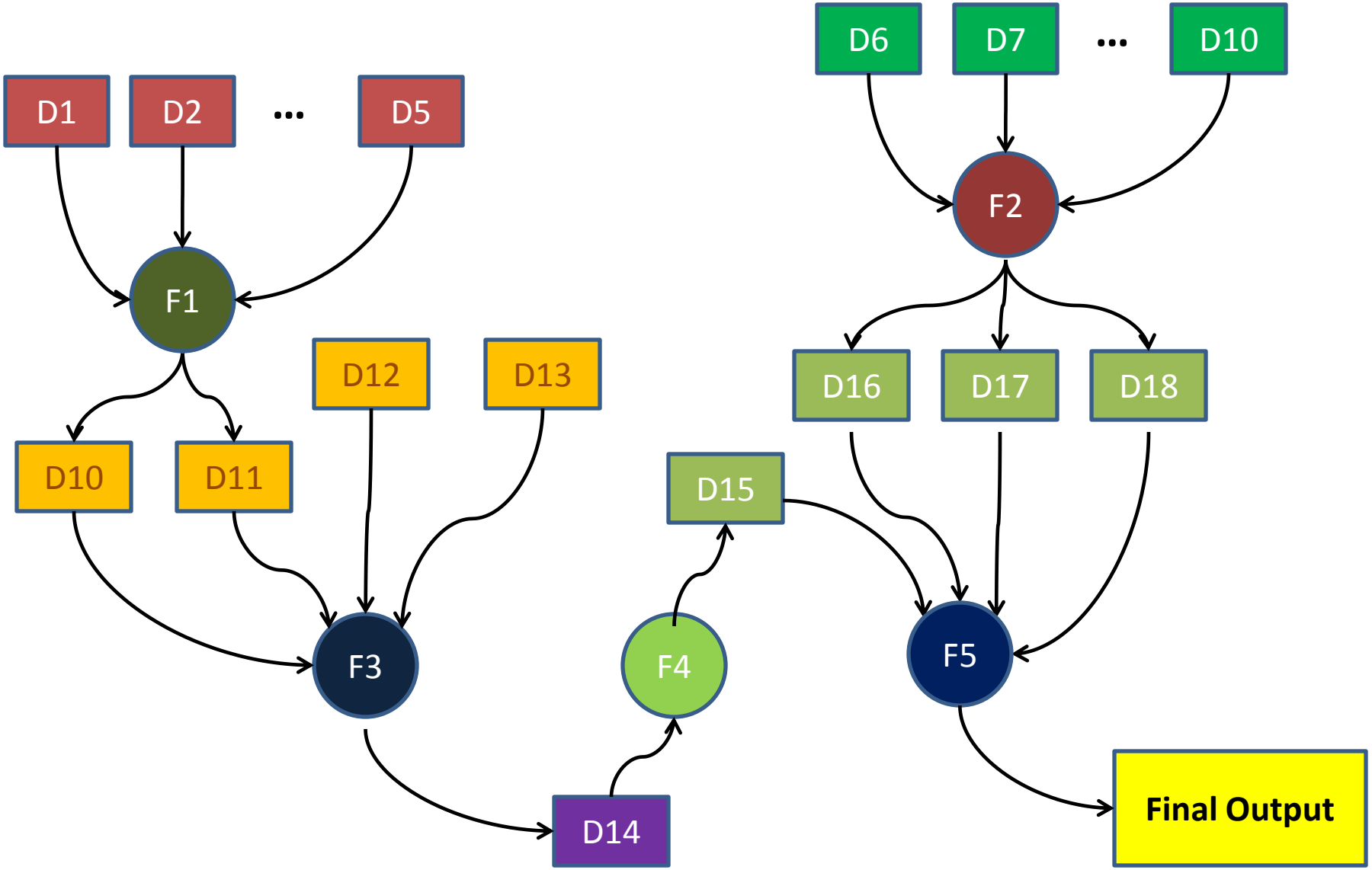
UNIVERSITY OF NOTRE DAME

Introduction



Source: Wikimedia

Introduction



Challenges

- Portability
  - Designed for a particular system
- Software maintainability
  - Goals and hypotheses change over time
- Dependency management
  - Reliance on third party libraries and executables

EST pipeline

Protein Synthesis:

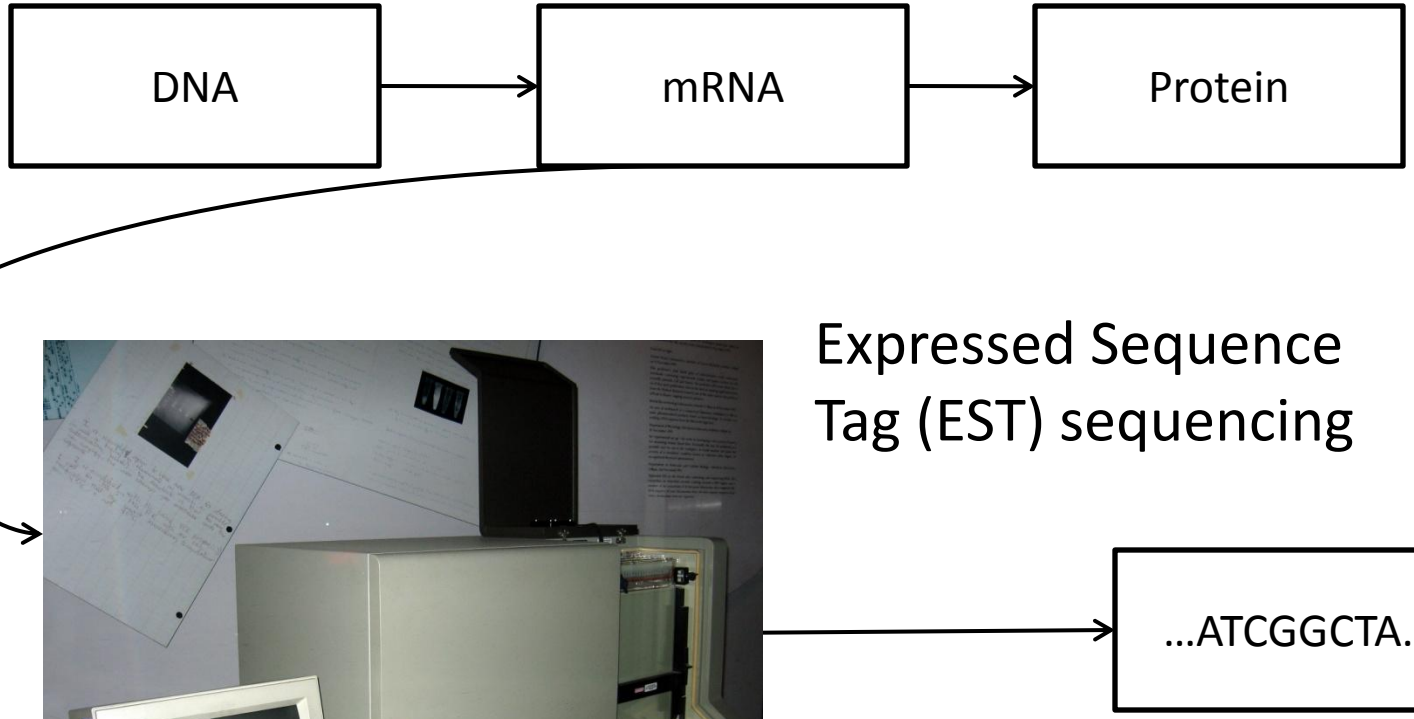
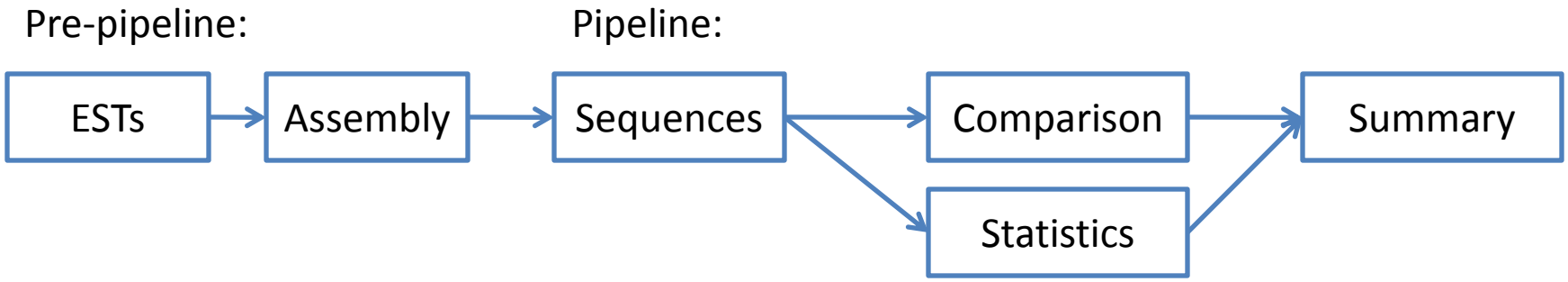
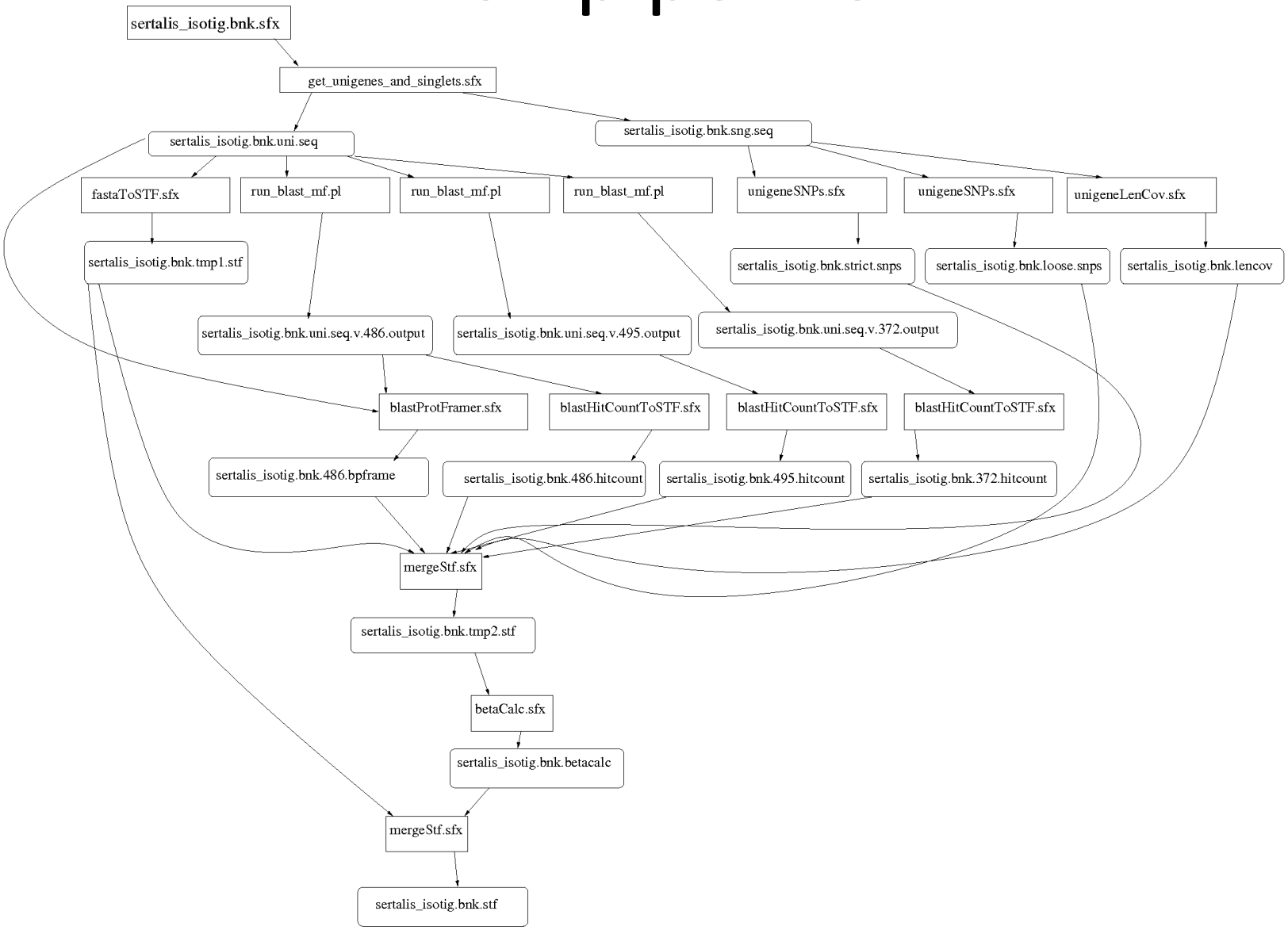


Photo Source: Wikimedia

EST pipeline



EST pipeline

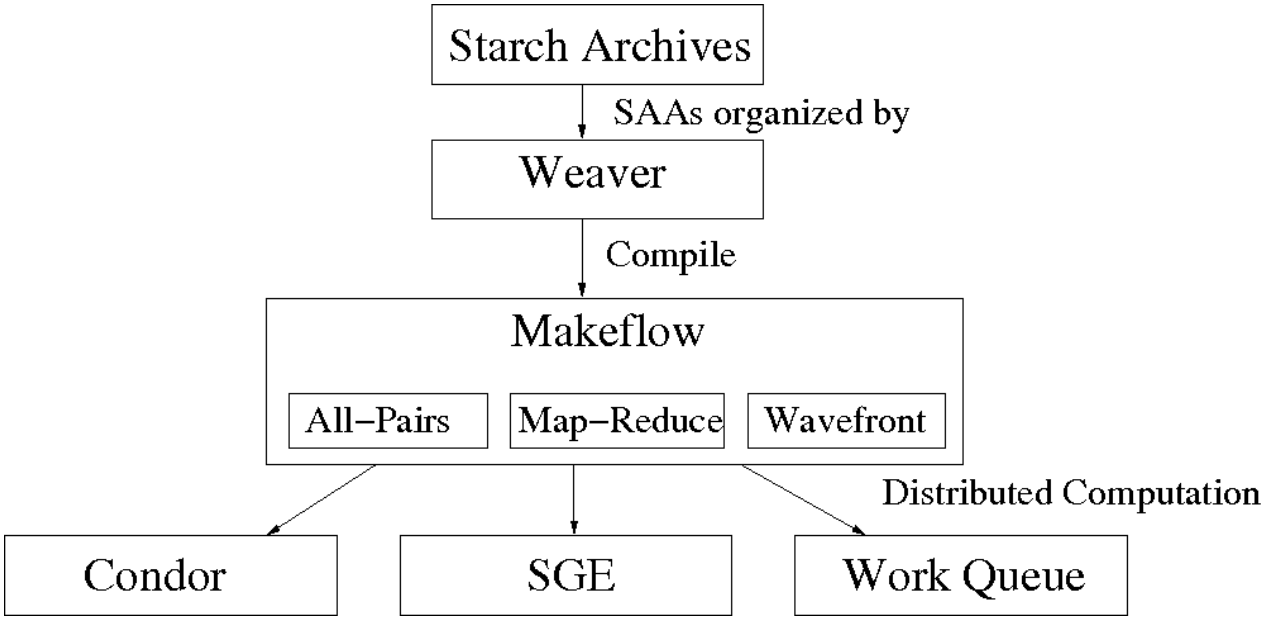


Solution Strategy

- Encapsulation of environment
- High level workflow specification language
- Portable low level workflow engine

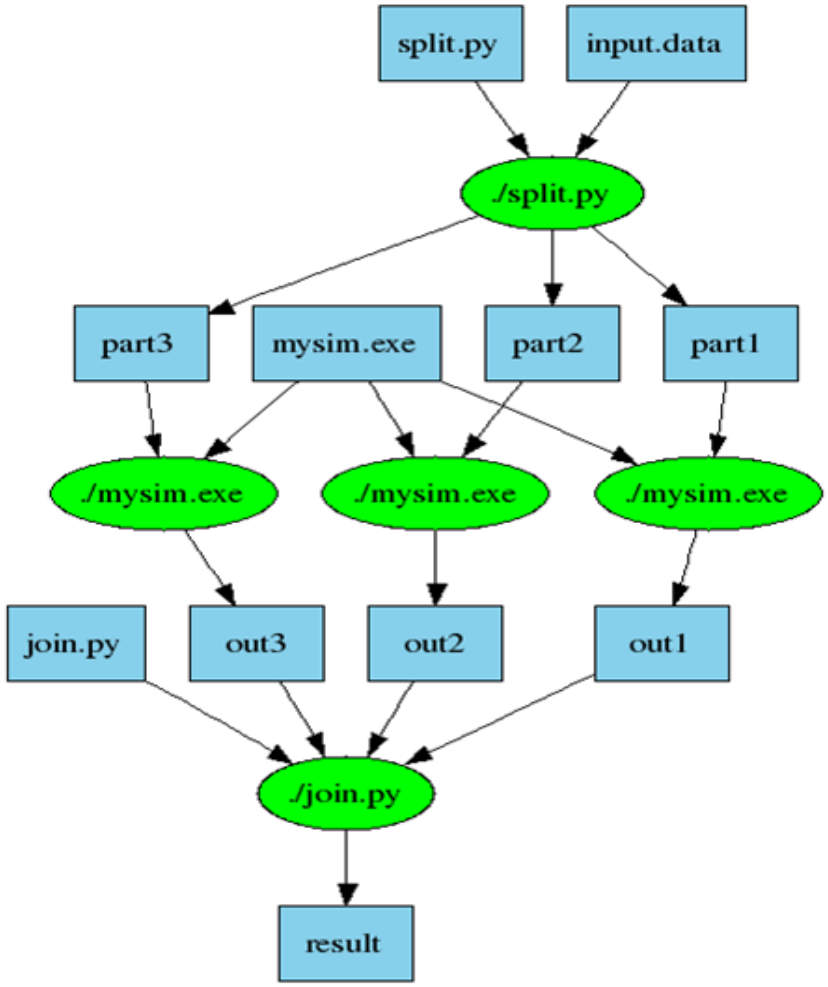
Solution Tools

- Starch
- Weaver
- Makeflow

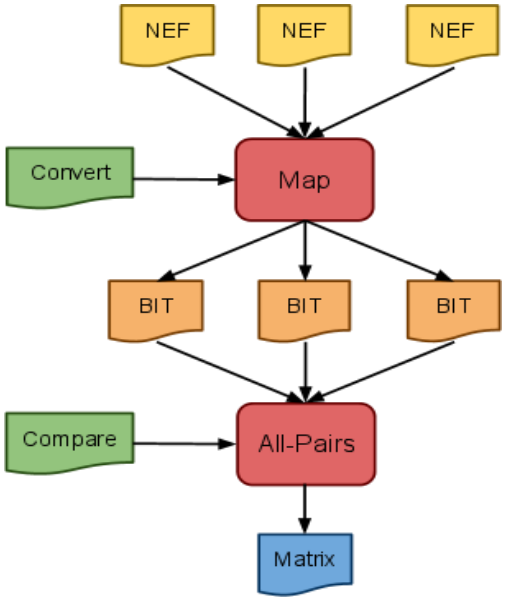


Solution Tools - Makeflow

```
part1 part2 part3: input.data split.py
        ./split.py input.data

out1: part1 mysim.exe
        ./mysim.exe part1 > out1

out2: part2 mysim.exe
        ./mysim.exe part2 > out2

out3: part3 mysim.exe
        ./mysim.exe part3 > out3

result: out1 out2 out3 join.py
        ./join.py out1 out2 out3 > result
```

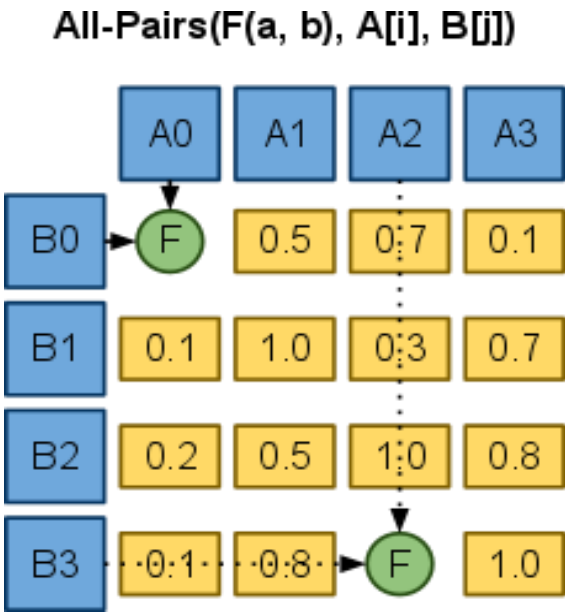


Solution Tools - Weaver



General DAG

+



Specific Abstraction

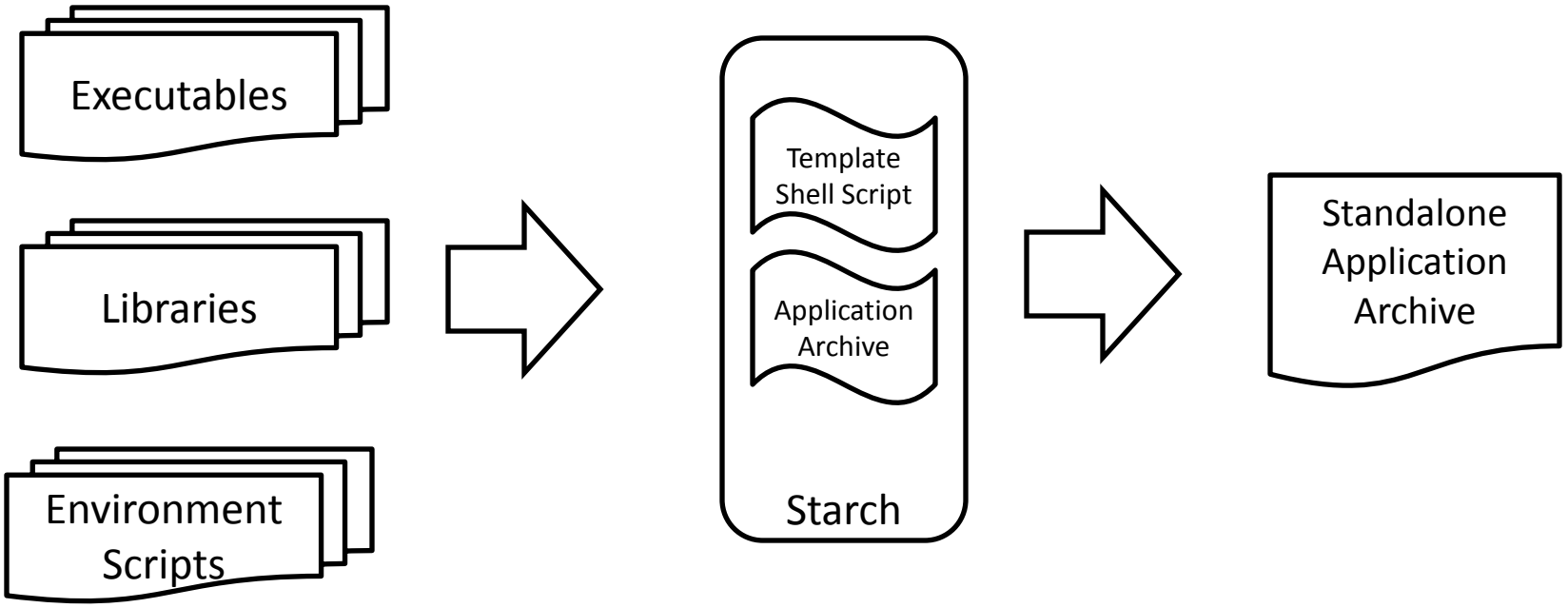
+

interface

=

Simplified Distributed Programming

Solution Tools - Starch



Final EST pipeline

Perl

```
print "$amosbnk.betacalc betacalc.err: betaCalc GeneticCode.rb $amosbnk.tmp2.stf ruby wrapper.pl
    $amosbnk.tgz bio.tgz bio-1.3.0.tgz ruby_gems-1.9.1.tgz bio.rb\n";
print "\t./wrapper.pl $amosbnk.tgz '(./ruby betaCalc $amosbnk.tmp2.stf ugAveCov seq maj_ugp_strict
    min_ugp_strict blastx_hsp MainStrict > $amosbnk.betacalc) >& betacalc.err'\n\n";
print "$amosbnk.stf stf.err: mergeStf $amosbnk.tmp1.stf $amosbnk.betacalc ruby wrapper.pl $amosbnk.tgz
    bio.tgz bio-1.3.0.tgz ruby_gems-1.9.1.tgz bio.rb\n";
print "\t./wrapper.pl $amosbnk.tgz '(./ruby mergeStf $amosbnk.tmp1.stf $amosbnk.betacalc > $amosbnk.stf) >&
    stf.err'\n";
```

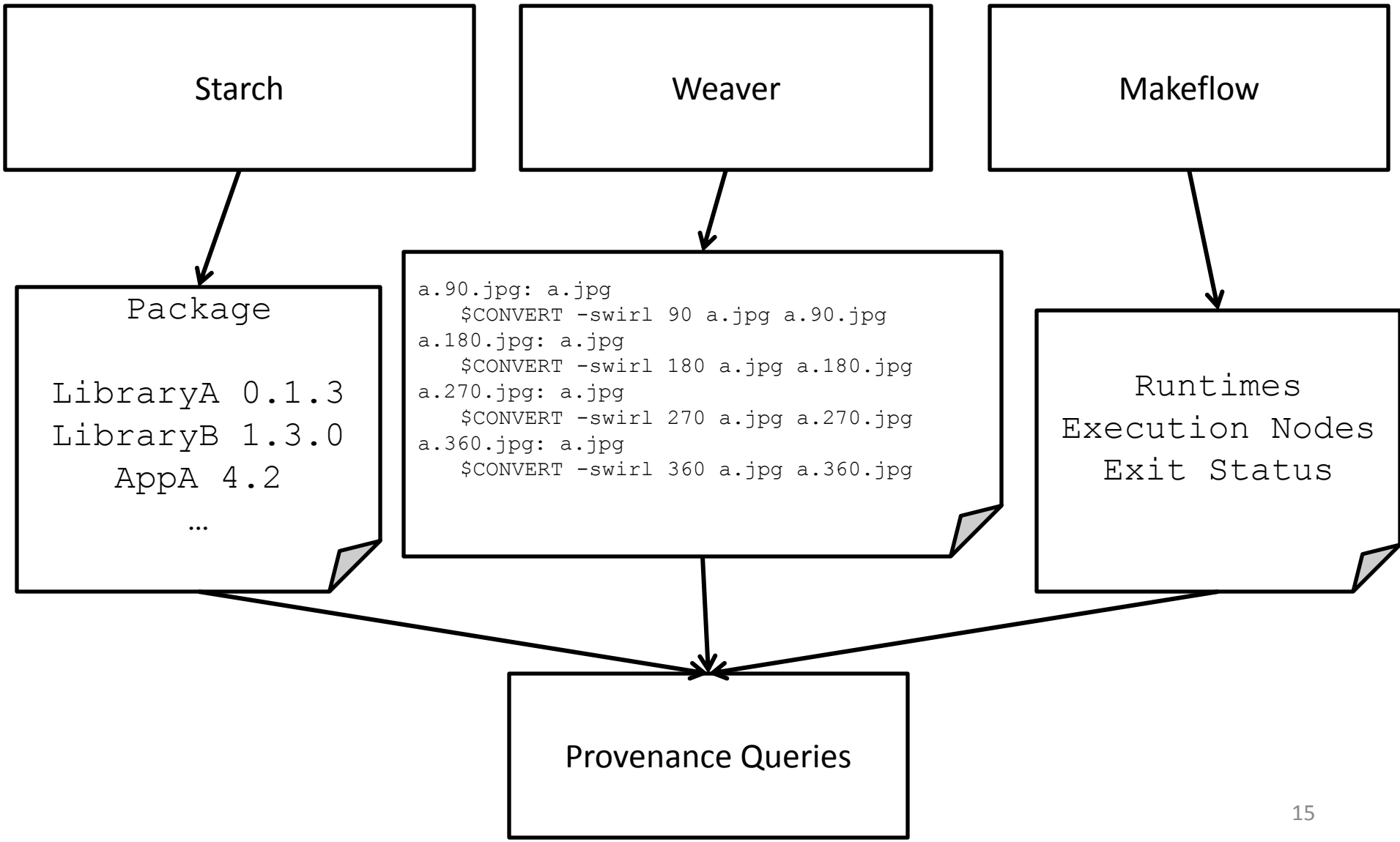
Weaver

```
f = betaCalc(str(bank)+'.tmp2.stf ugAveCov seq maj_ugp_strict min_ugp_strict blastx_hsp MainStrict')
t = Run(f, '', output = f.output_string(str(bank)+'.betacalc'))

f = mergeStf(str(bank)+'.tmp1.stf '+str(bank)+'.betacalc')
t = Run(f, '', output = f.output_string(str(bank)+'.stf'))
```

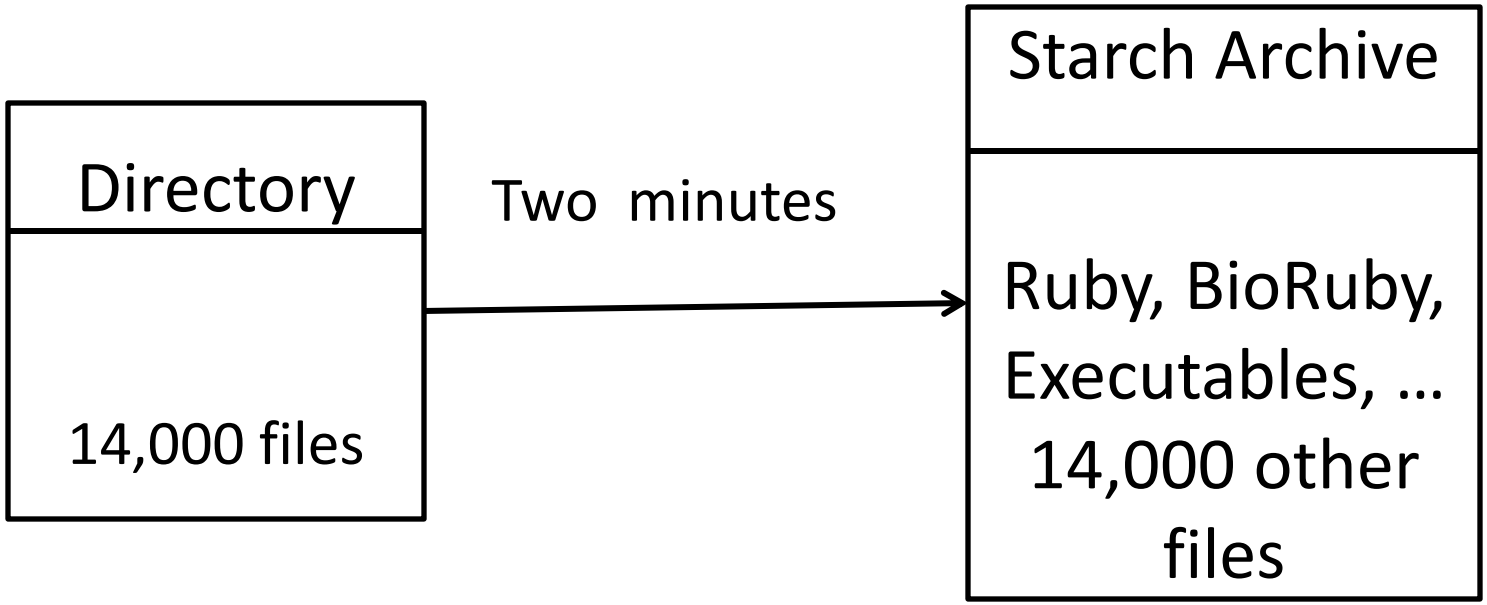
Pipeline Characteristics

- Provenance
- Encapsulation
- Performance
- Portability
- Fault Tolerance
- Code Clarity

Pipeline Characteristics - Provenance



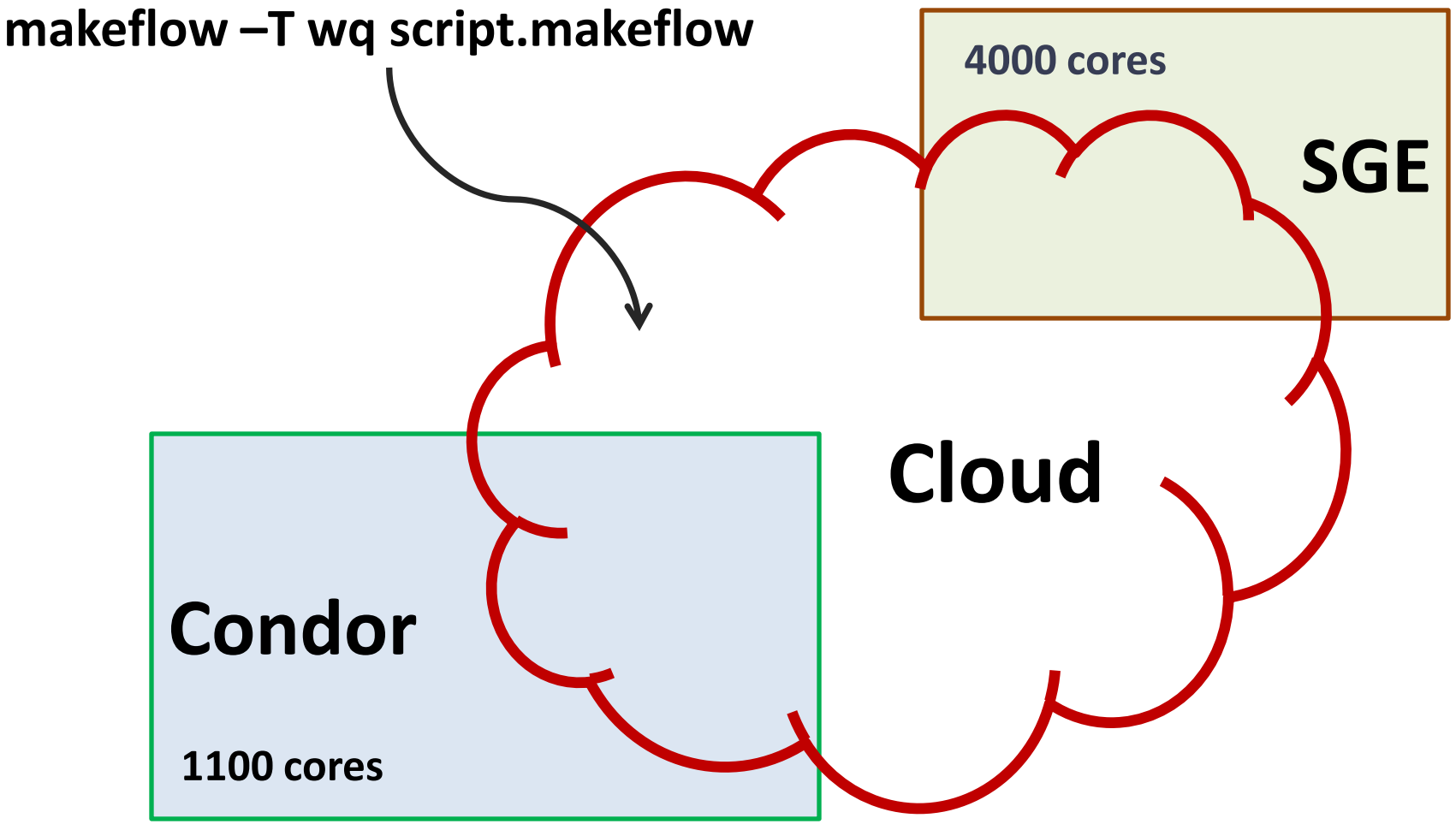
Pipeline Characteristics - Encapsulation



Pipeline Characteristics - Performance

- Manual (sequential execution) – 1 week
- Makeflow (automated execution) – 1 hour
- No difference in Perl and Weaver

Pipeline Characteristics - Portability



Pipeline Characteristics – Fault Tolerance

- Makeflow tracks job progress and supports automated rerunning failed tasks

- Makeflow can resume execution from stoppage

Pipeline Characteristics – Code Clarity

- Weaver generating a Makeflow

```
f = split(query, database)
t = Run(f, '', output = f.output_string(split_out), local = True)
Map(Blast, files, output = 'blast.output', merge_func = 'cat' )
```

- Perl generating a Makeflow

```
for (my $i = 0; $i < $num_splits; $i++) {
		$inputlist .= "$inputfile.v.$dbfile.input.$i ";
		$outputlist .= "$inputfile.v.$dbfile.output.$i ";
		$errorlist .= "$inputfile.v.$dbfile.error.$i ";}
print OUTFILE $inputlist . ": $inputfile split_inputs.pl\n";
print OUTFILE "\tLOCAL ./split_inputs.pl $qry_granularity $char_granularity $inputfile
$dbfile\n\n";

for (my $i=0; $i < $num_splits; $i++) {a
		print OUTFILE "$inputfile.v.$dbfile.output.$i $inputfile.v.$dbfile.error.$i:
$inputfile.v.$dbfile.input.$i $dbfile.tgz blastall tee blastwrapper.pl\n";
		print OUTFILE "\t./blastwrapper.pl $dbfile $inputfile $i\n\n";
```

Conclusion – Lessons Learned

- Code modularity
- Code clarity
- Abstractions
- Portability

Acknowledgments

- My advisor – Dr. Scott Emrich
- Notre Dame Bioinformatics Lab
  - Allison Regier
  - Shawn O'Neil
  - Rory Carmichael
  - Andrew Rider
  - Irena Lanc
  - Lauren Assour
- Cooperative Computing Lab – Dr. Douglas Thain
  - Peter Bui
  - Li Yu

Software

- Makeflow
  - http://www.nd.edu/~ccl/software/makeflow
  - or Google “Makeflow”
- Weaver and Starch
  - http://bitbucket.org/pbui/

Biocompute

http://biocompute.cse.nd.edu/