Scaling Up Scientific Workflows with Makeflow

Li Yu

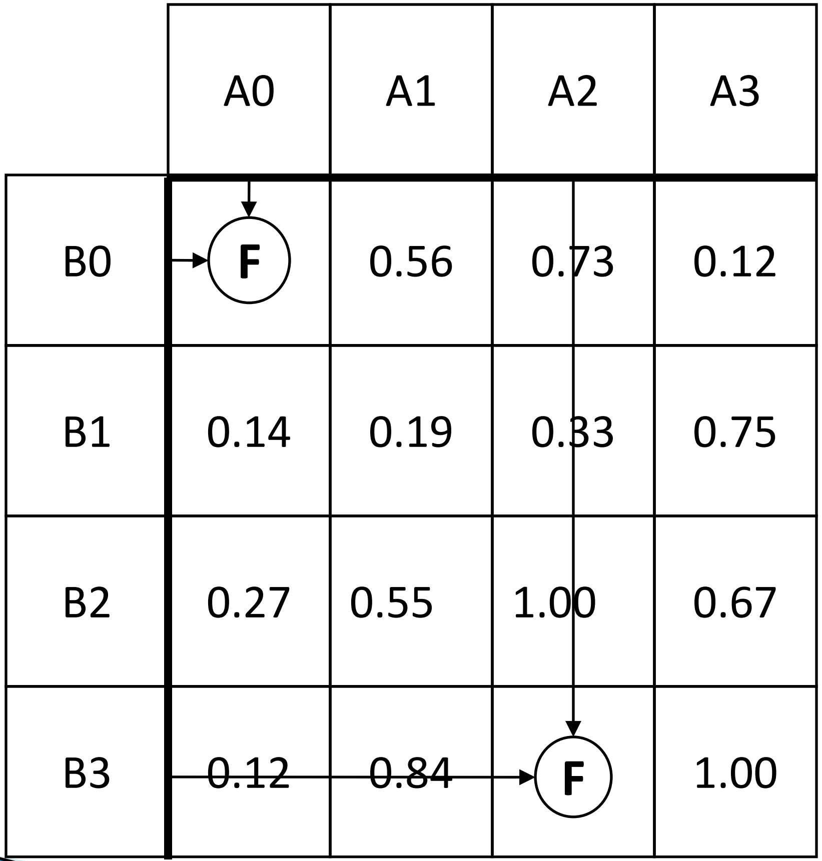
University of Notre Dame

Overview

- Distributed systems are hard to use!
- An abstraction is a regular structure that can be efficiently scaled up to large problem sizes.
- We have implemented abstractions such as AllPairs and Wavefront.
- Today – Makeflow and Work Queue:
  - Makeflow is a **workflow engine** for executing large complex workflows on clusters, grids and clouds.
  - Work Queue is Master/Worker framework.
  - Together they are **compact, portable, data oriented, good at lots of small jobs and familiar syntax.**

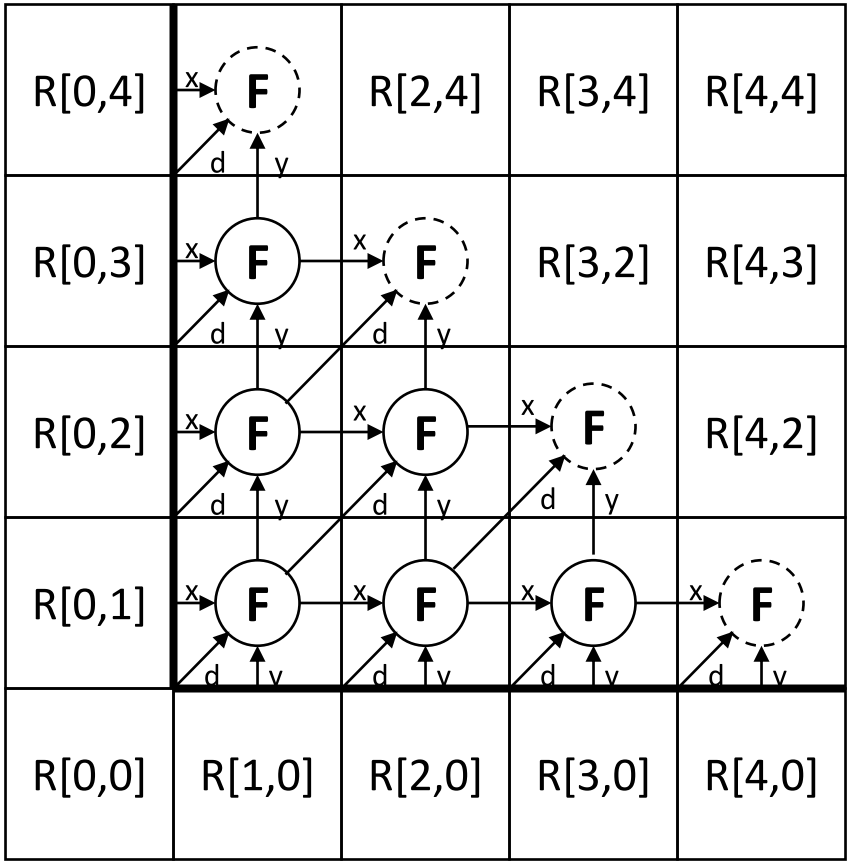
Specific Abstractions: AllPairs & Wavefront

AllPairs:



	A0	A1	A2	A3
B0	F	0.56	0.73	0.12
B1	0.14	0.19	0.33	0.75
B2	0.27	0.55	1.00	0.67
B3	0.12	0.84	F	1.00

Wavefront:



R[0,4]	F	R[2,4]	R[3,4]	R[4,4]
R[0,3]	F	F	R[3,2]	R[4,3]
R[0,2]	F	F	F	R[4,2]
R[0,1]	F	F	F	F
R[0,0]	R[1,0]	R[2,0]	R[3,0]	R[4,0]

Makeflow

- Makeflow is a **workflow engine** for executing large complex workflows on clusters, grids and clouds.
- Can express any arbitrary **D**irected **A**cyclic **G**raph (DAG).
- Good at lots of small jobs.
- Data is treated as a first class citizen.
- Has a syntax similar to traditional UNIX Make
- It is fault-tolerant.

Don’t We Already Have DAGMan?

- DAGMan is great!

- But Makeflow…
  - Workflow specification in just ONE file.
  - Uses Master/Worker model.
  - Treats data as a first class citizen

- Experiment: Create 1M Job DAG
  - DAGMan: 6197 s *just to write the files*
  - Makeflow: 69 s to write the Makeflow.

Makeflow

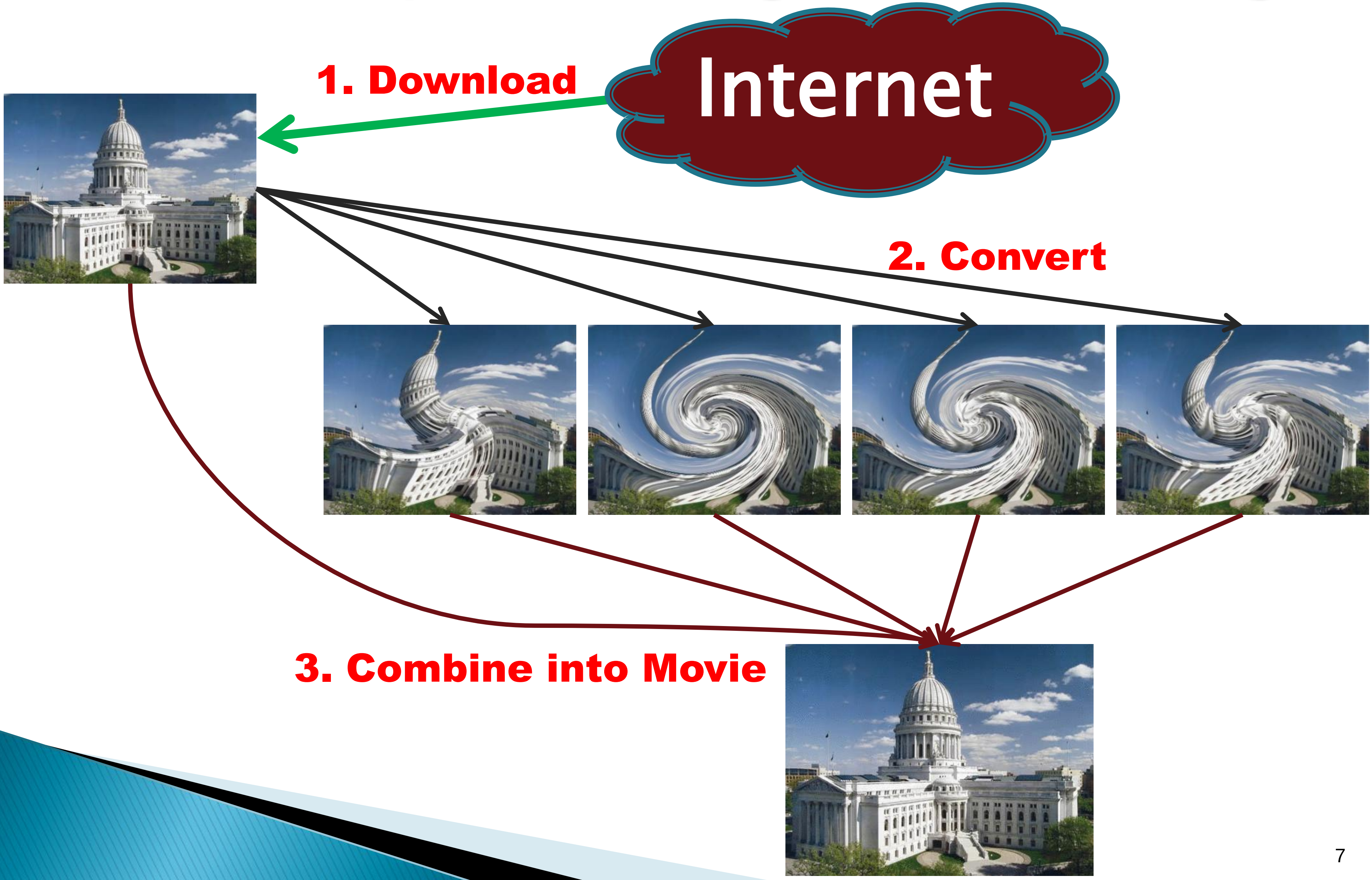
≈

DAGMan

+

Master/Worker

An Example – Image Processing



Internet

1. Download

2. Convert

3. Combine into Movie

An Example – Makeflow Script

```
# This is an example of Makeflow.
CURL=/usr/bin/curl
CONVERT=/usr/bin/convert
URL=http://www.cse.nd.edu/~ccl/images/a.jpg
a.montage.gif: a.jpg a.90.jpg a.180.jpg a.270.jpg a.360.jpg
  LOCAL $CONVERT -delay 10 -loop 0 a.jpg a.90.jpg a.180.jpg
  a.270.jpg a.360.jpg a.270.jpg a.180.jpg a.90.jpg a.montage.gif
a.90.jpg: a.jpg
  $CONVERT -swirl 90 a.jpg a.90.jpg
a.180.jpg: a.jpg
  $CONVERT -swirl 180 a.jpg a.180.jpg
a.270.jpg: a.jpg
  $CONVERT -swirl 270 a.jpg a.270.jpg
a.360.jpg: a.jpg
  $CONVERT -swirl 360 a.jpg a.360.jpg
a.jpg: LOCAL
  $CURL -o a.jpg $URL
```

An Example – Makeflow Script

```
# This is an example of Makeflow.
CURL=/usr/bin/curl
CONVERT=/usr/bin/convert
URL=http://www.cse.nd.edu/~ccl/images/a.jpg
a.montage.gif: a.jpg a.90.jpg a.180.jpg a.270.jpg a.360.jpg
  LOCAL $CONVERT -delay 10 -loop 0 a.jpg a.90.jpg a.180.jpg
  a.270.jpg a.360.jpg a.270.jpg a.180.jpg a.90.jpg a.montage.gif
a.90.jpg: a.jpg
  $CONVERT -swirl 90 a.jpg a.90.jpg
a.180.jpg: a.jpg
  $CONVERT -swirl 180 a.jpg a.180.jpg
a.270.jpg: a.jpg
  $CONVERT -swirl 270 a.jpg a.270.jpg
a.360.jpg: a.jpg
  $CONVERT -swirl 360 a.jpg a.360.jpg
a.jpg:
  LOCAL $CURL -o a.jpg $URL
```

Running the Makeflow

- Just use the local machine:

  ```
  % makeflow sample.makeflow
  ```

- Use a distributed system with ‘-T’ option:
  - ‘-T condor’: uses the Condor batch system

    ```
    % makeflow -T condor sample.makeflow
    ```

  - Take advantage of Condor MatchMaker

    ```
    BATCH_OPTIONS=Requirements =(Memory>1024)\n Arch = x86_64
    ```

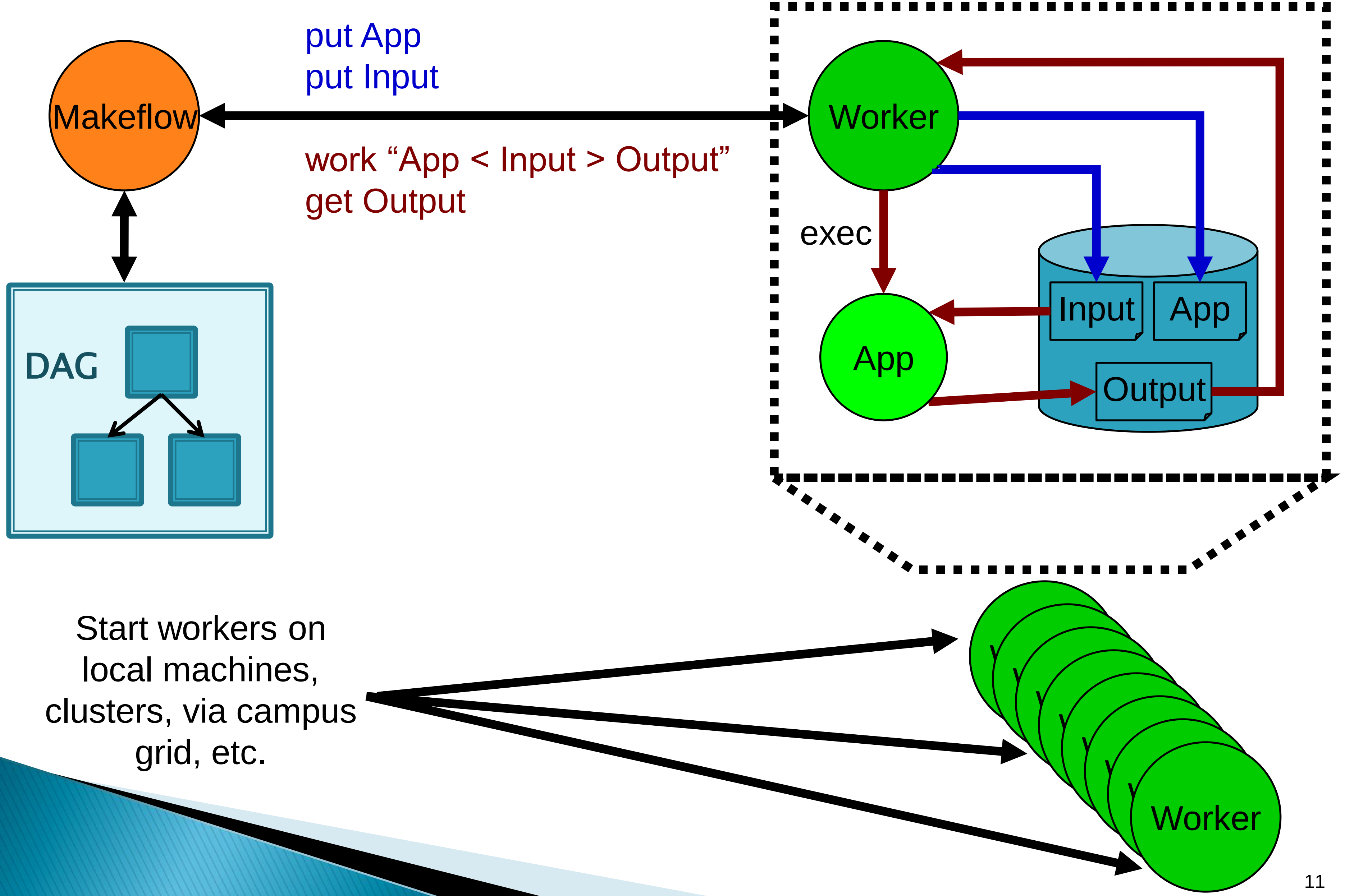
  - ‘-T sge’: uses the Sun Grid Engine

    ```
    % makeflow -T sge sample.makeflow
    ```

  - ‘-T wq’: uses the Work Queue framework

    ```
    % makeflow -T wq sample.makeflow
    ```

Makeflow with Work Queue



Makeflow

put App
put Input

work “App < Input > Output”
get Output

DAG

Worker

exec

App

Input

App

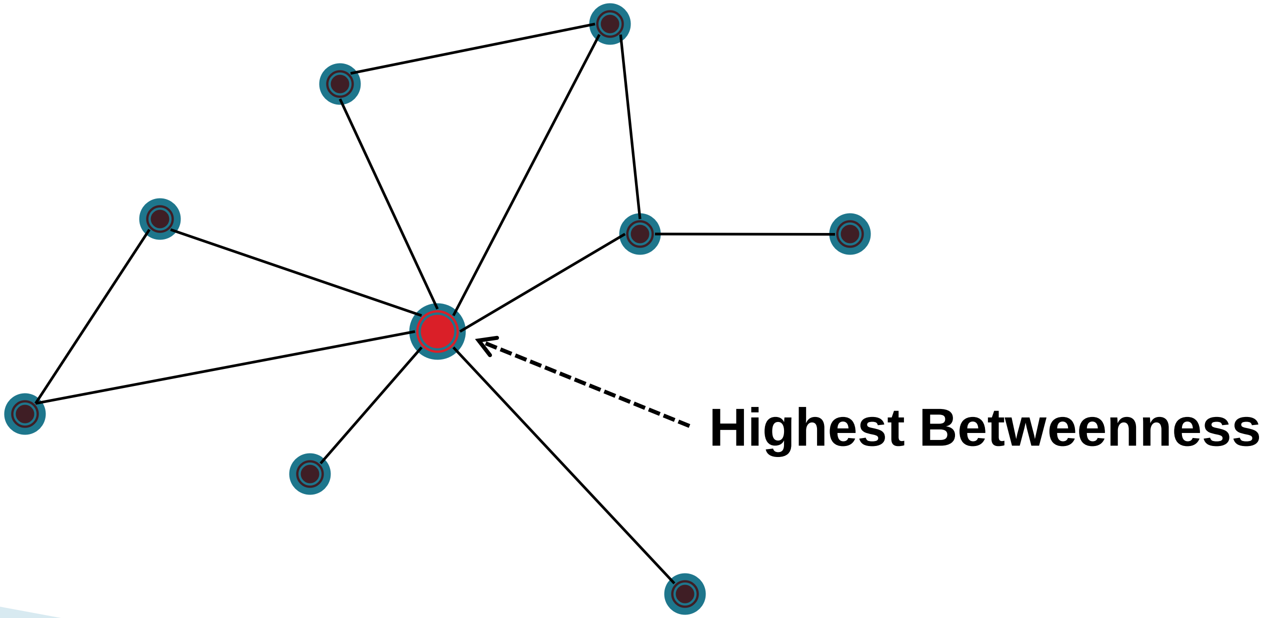
Output

Start workers on
local machines,
clusters, via campus
grid, etc.

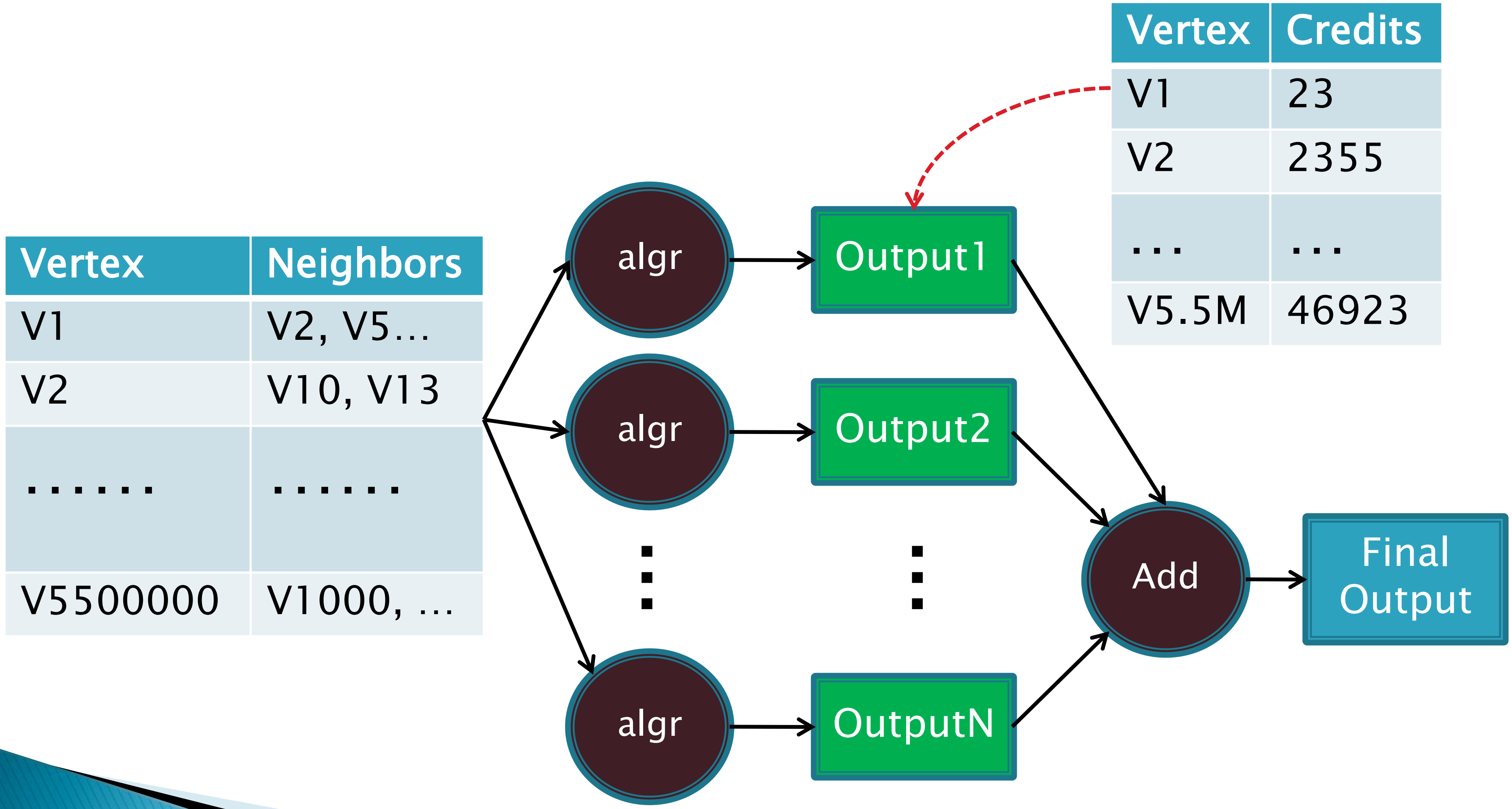
Worker

Application – Data Mining

- Betweenness Centrality
  - Vertices that occur on many shortest paths between other vertices have higher betweenness than those that do not.
  - Application: social network analysis.
  - Complexity: O(n^3) where ‘n’ is the number of vertices.



The Workflow



Vertex	Neighbors
V1	V2, V5…
V2	V10, V13
…….	…….
V5500000	V1000, …

Vertex	Credits
V1	23
V2	2355
…	…
V5.5M	46923

Size of the Problem

- About 5.5 million vertices
- About 20 million edges
- Each job computes 50 vertices (110K jobs)

Input Data Format

Vertex	Neighbors
V1	V2, V5…
V2	V10, V13
…….	…….
V5500000	V1000, …

Raw : 250MB
Gzipped: 93MB

Output Data Format

Vertex	Credits
V1	23
V2	2355
…	…
V5.5M	46923

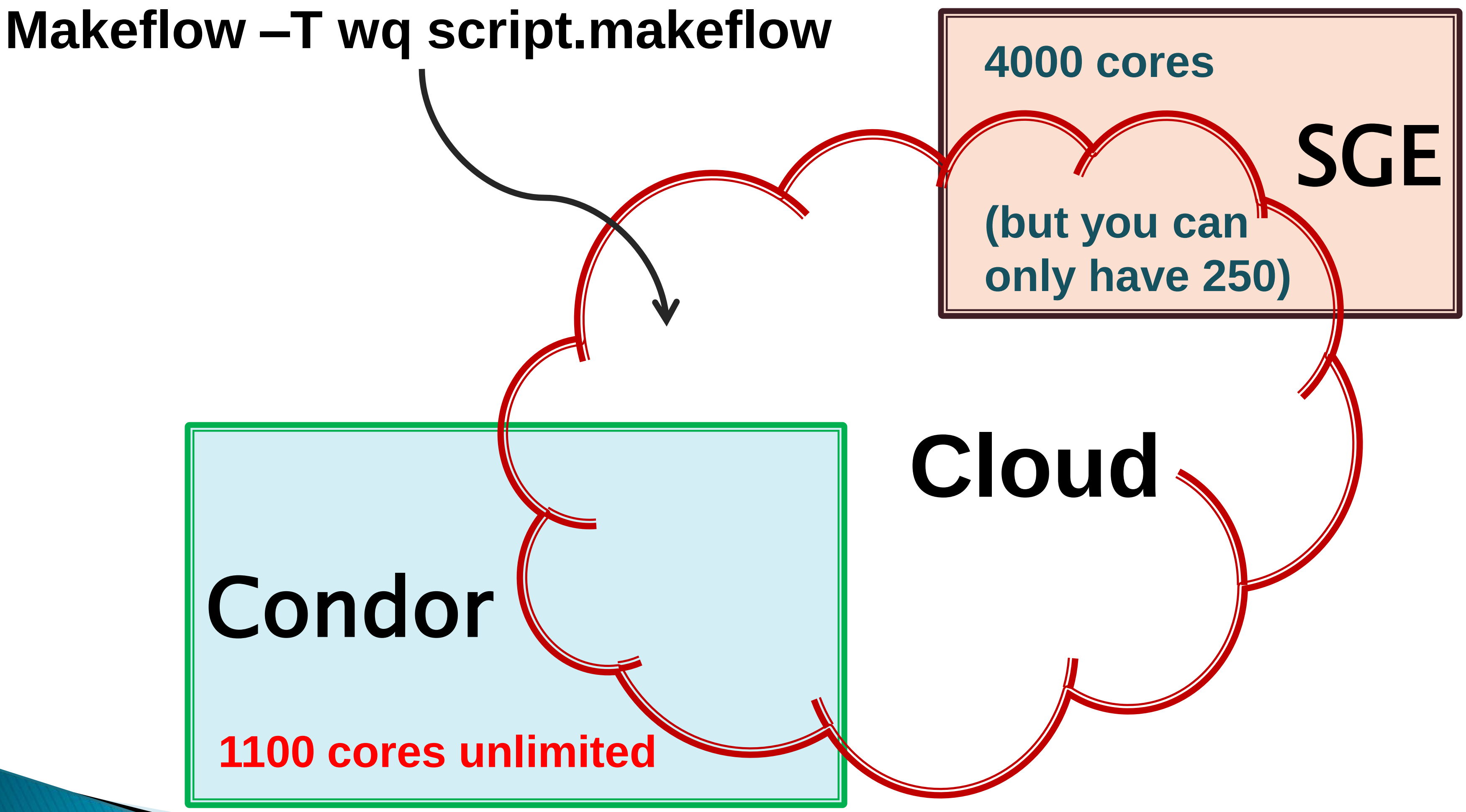
Raw : 30MB
Gzipped: 13MB

The Result

- Resource used:
  - 300 Condor CPU cores
  - 250 SGE CPU cores

- Runtime:
  - 2000 CPU Days -> 4 Days
  - 500X speedup!

Make Your Own Cloud



Makeflow –T wq script.makeflow

4000 cores

SGE

(but you can only have 250)

Cloud

Condor

1100 cores unlimited

Making a Cloud is as simple as:

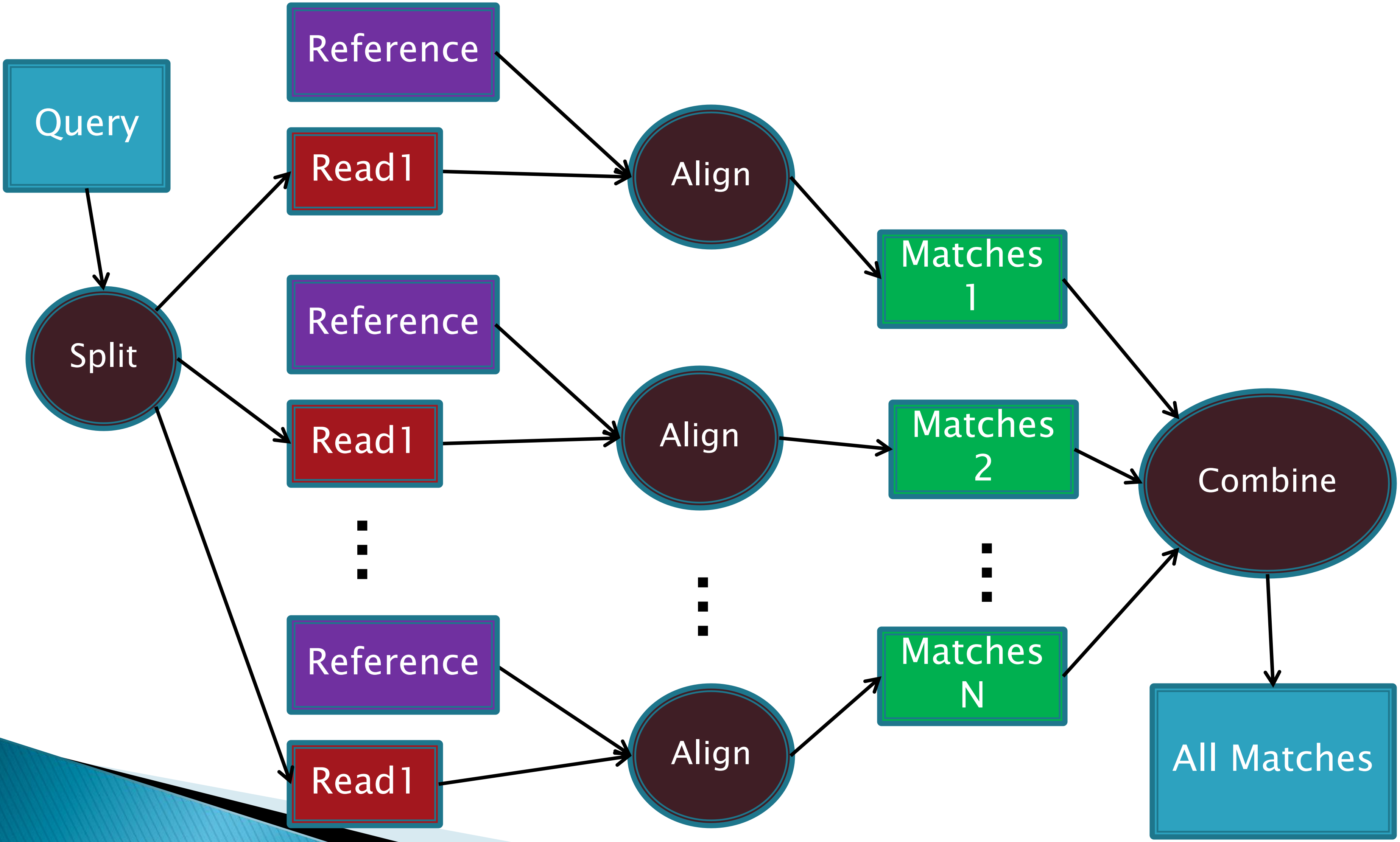
```
$>condor_submit_workers master.nd.edu 9012 300

$>sge_submit_workers master.nd.edu 9012 250

$>makeflow –T wq script.makeflow
```

Application – Biocompute

- Sequence Search and Alignment by Hashing Algorithm (SSAHA)
- Short Read Mapping Package (SHRiMP)
- Genome Alignment:

  -
    ```
    CGGAAATAATTATTAAGCAA
       |||||||||
    GTCAAATAATTACTGGATCG
    ```
- Single nucleotide polymorphism (SNP) discovery

The Workflow



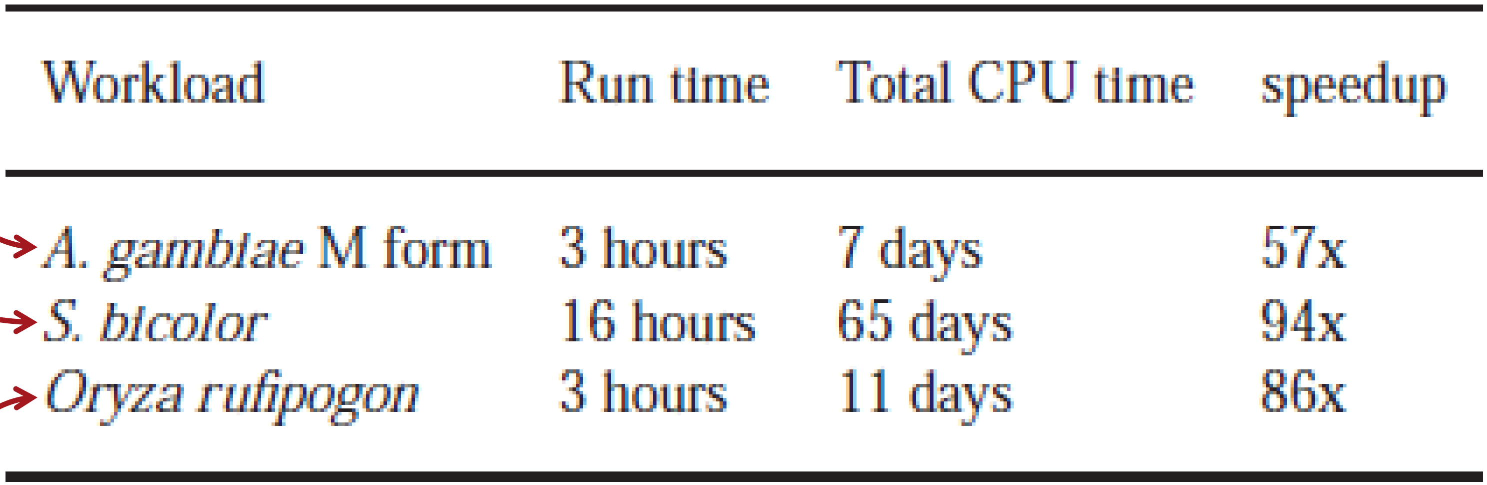
Sizes of some real workloads

- *Anopheles gambiae: 273 million* bases
  - 2.5 million reads consisting of 1.5 billion bases were aligned using SSAHA

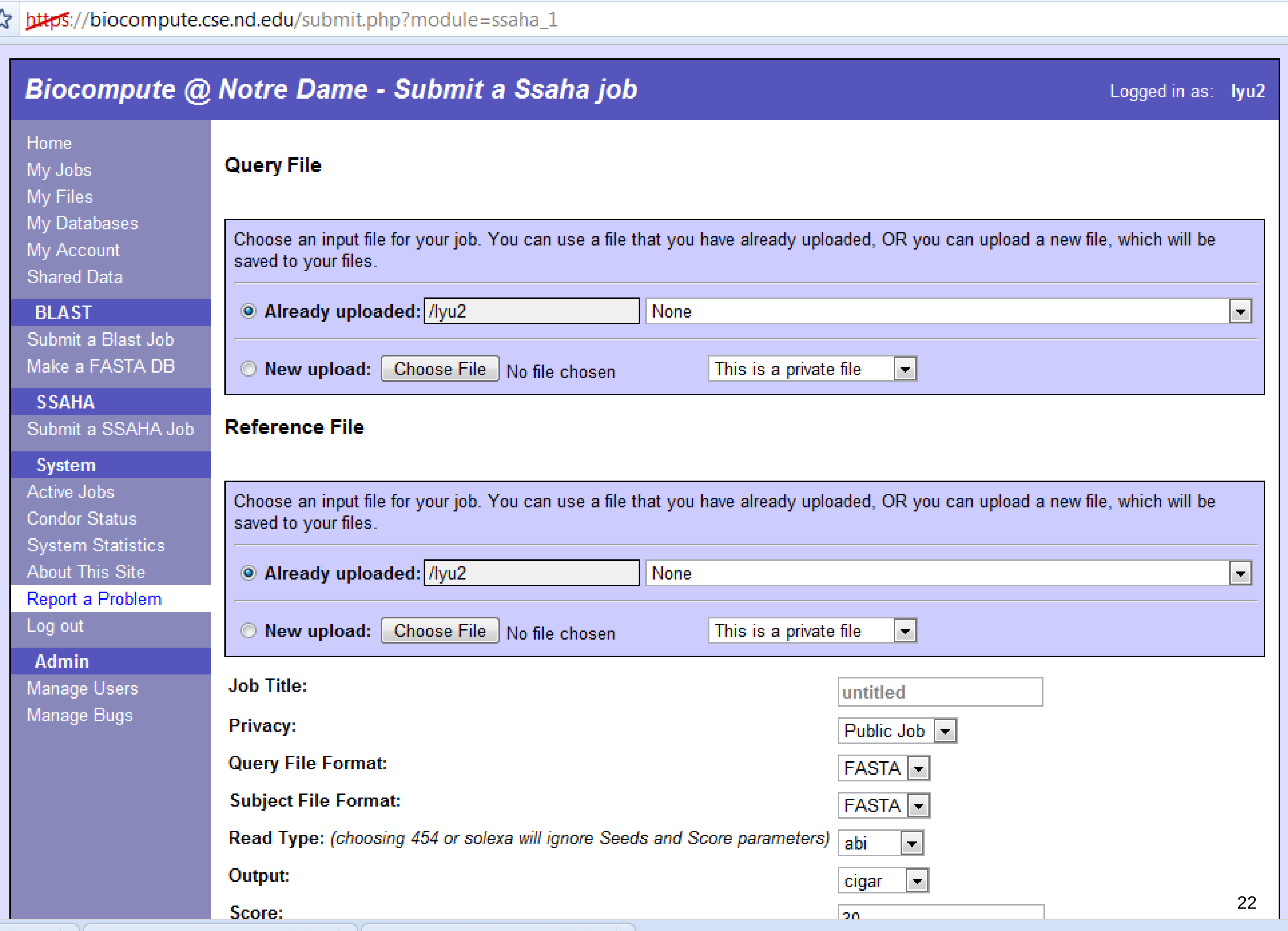
- *Sorghum bicolor:* 738.5 million bases
  - 11.5 million sequences consisting of 11 billion bases were aligned using SSAHA

- 7 million query reads of *Oryza rufipogon to the genome Oryza sativa using SHRiMP*

Performance



Workload	Run time	Total CPU time	speedup
A. gambiae M form	3 hours	7 days	57x
S. bicolor	16 hours	65 days	94x
Oryza rufipogon	3 hours	11 days	86x



~~https~~://biocompute.cse.nd.edu/submit.php?module=ssaha_1

Biocompute @ Notre Dame - Submit a Ssaha job

Logged in as: lyu2

Query File

Choose an input file for your job. You can use a file that you have already uploaded, OR you can upload a new file, which will be saved to your files.

● **Already uploaded:** /lyu2 None

○ **New upload:** Choose File No file chosen This is a private file

Reference File

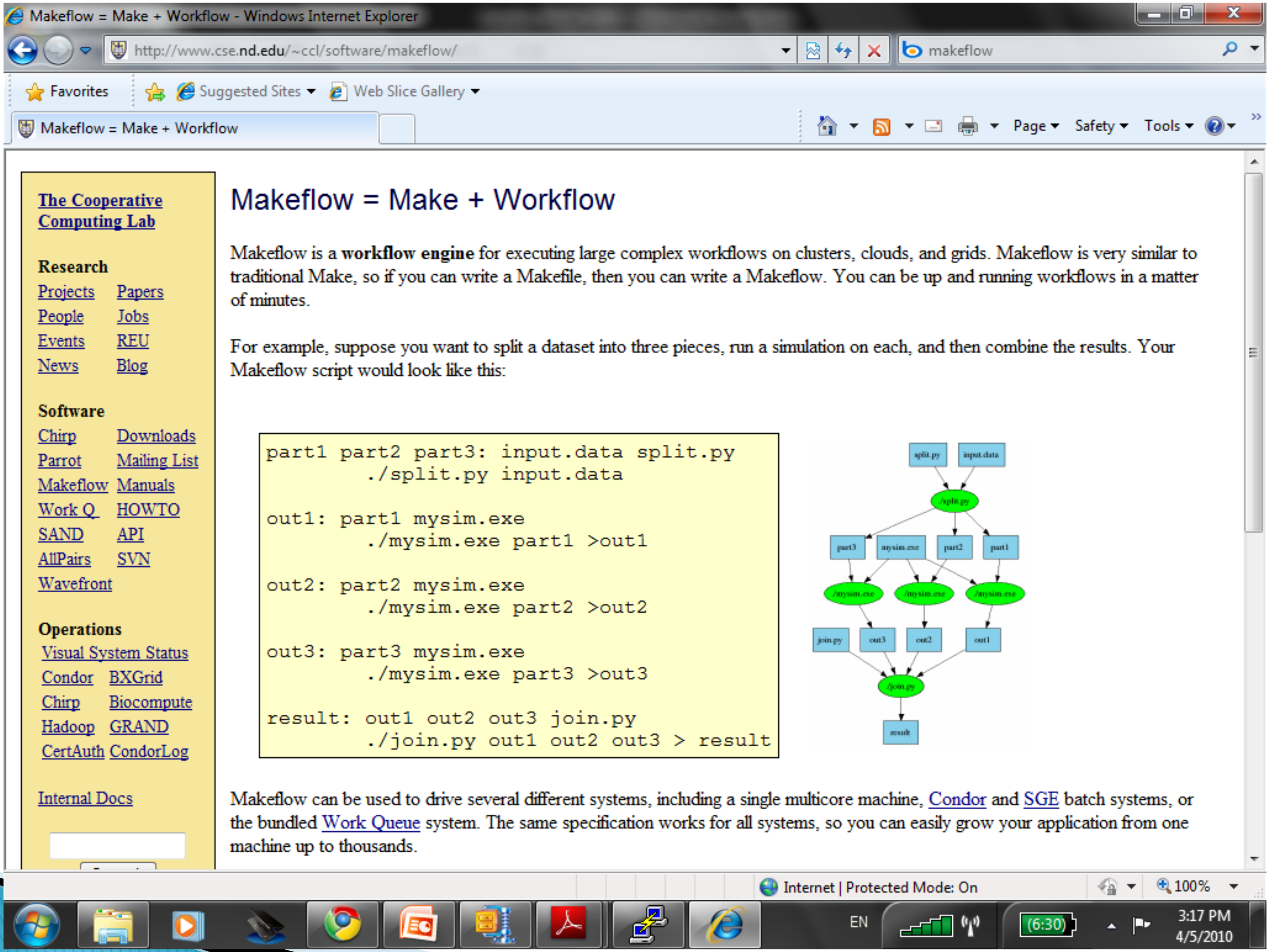
Choose an input file for your job. You can use a file that you have already uploaded, OR you can upload a new file, which will be saved to your files.

● **Already uploaded:** /lyu2 None

○ **New upload:** Choose File No file chosen This is a private file

Job Title:	untitled
Privacy:	Public Job
Query File Format:	FASTA
Subject File Format:	FASTA
Read Type: *(choosing 454 or solexa will ignore Seeds and Score parameters)*	abi
Output:	cigar
Score:	30

Google “Makeflow”



The Cooperative Computing Lab

Research

Projects Papers
People Jobs
Events REU
News Blog

Software

Chirp Downloads
Parrot Mailing List
Makeflow Manuals
Work Q HOWTO
SAND API
AllPairs SVN
Wavefront

Operations

Visual System Status
Condor BXGrid
Chirp Biocompute
Hadoop GRAND
CertAuth CondorLog

Internal Docs

Makeflow = Make + Workflow

Makeflow is a **workflow engine** for executing large complex workflows on clusters, clouds, and grids. Makeflow is very similar to traditional Make, so if you can write a Makefile, then you can write a Makeflow. You can be up and running workflows in a matter of minutes.

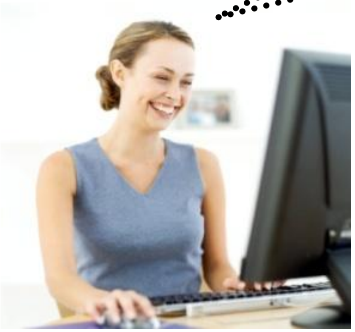
For example, suppose you want to split a dataset into three pieces, run a simulation on each, and then combine the results. Your Makeflow script would look like this:

```
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
```

Makeflow can be used to drive several different systems, including a single multicore machine, Condor and SGE batch systems, or the bundled Work Queue system. The same specification works for all systems, so you can easily grow your application from one machine up to thousands.

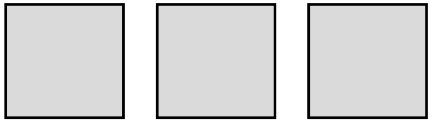
Extra Slides

A Quick Review of Abstractions

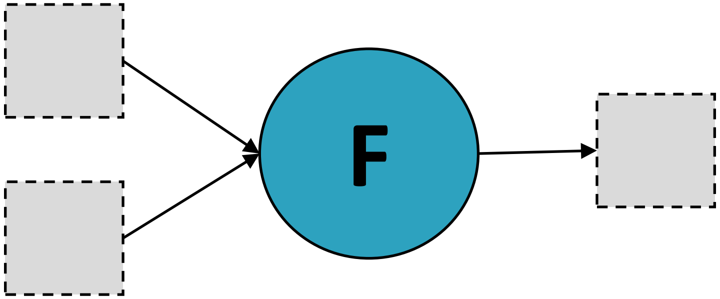


Here is my function: F(x,y)
Here is a folder of files: set S

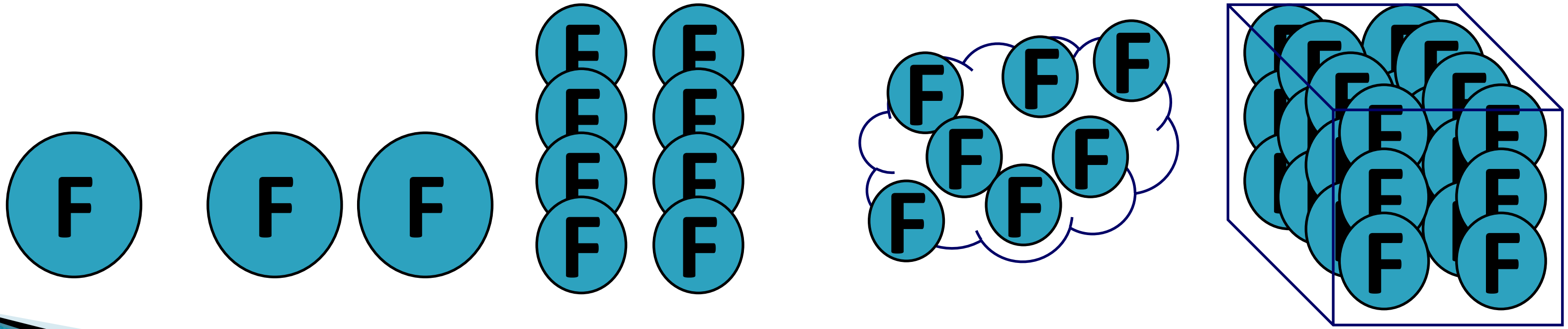
set S of files



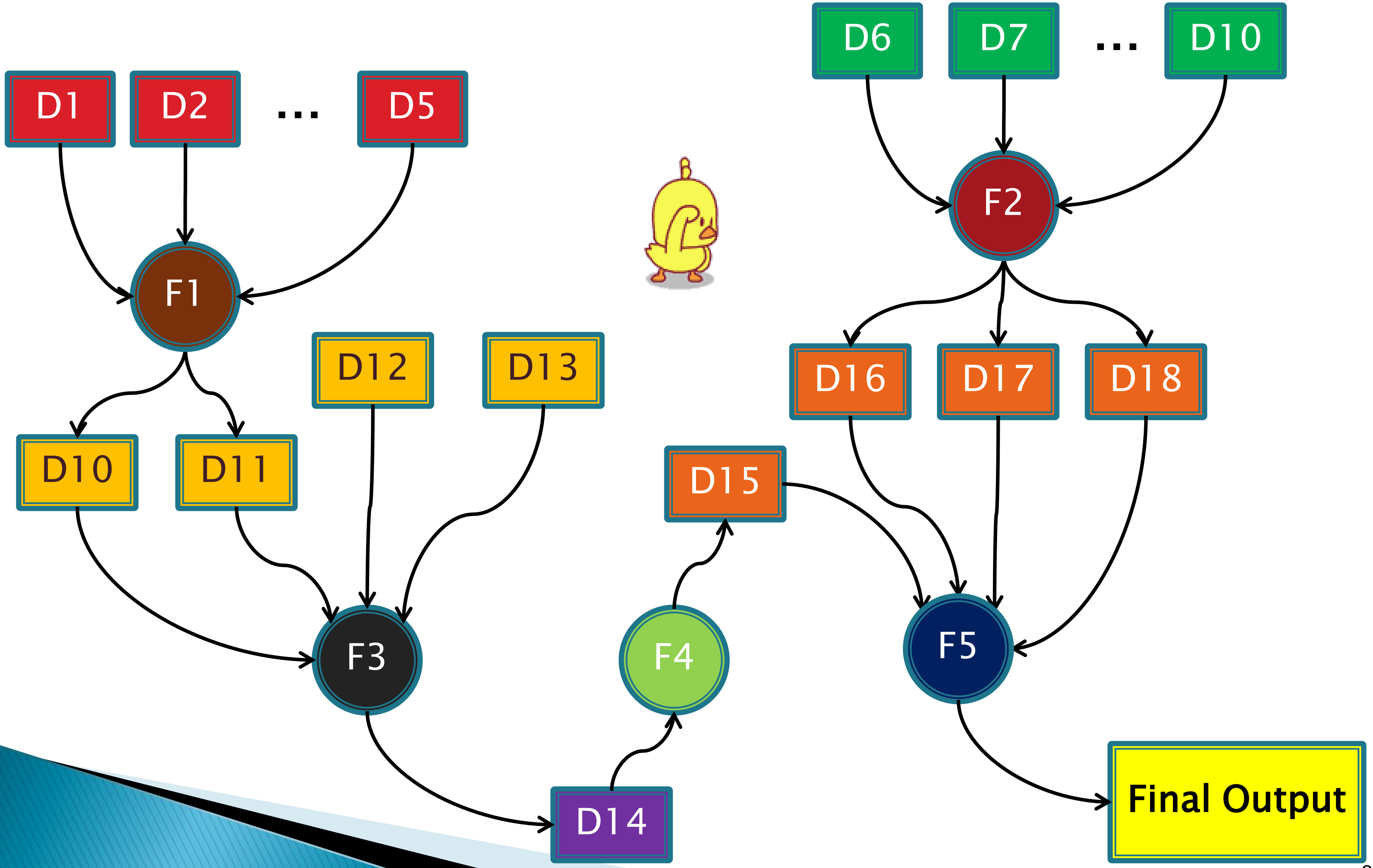
binary function F



1CPU Multicore Cluster Grids Supercomputer



Sometimes …



D1

D2

…

D5

F1

D10

D11

D12

D13

F3

D14

F4

D15

D6

D7

…

D10

F2

D16

D17

D18

F5

Final Output