Parallel Python Using Jupyter & IPyParallel

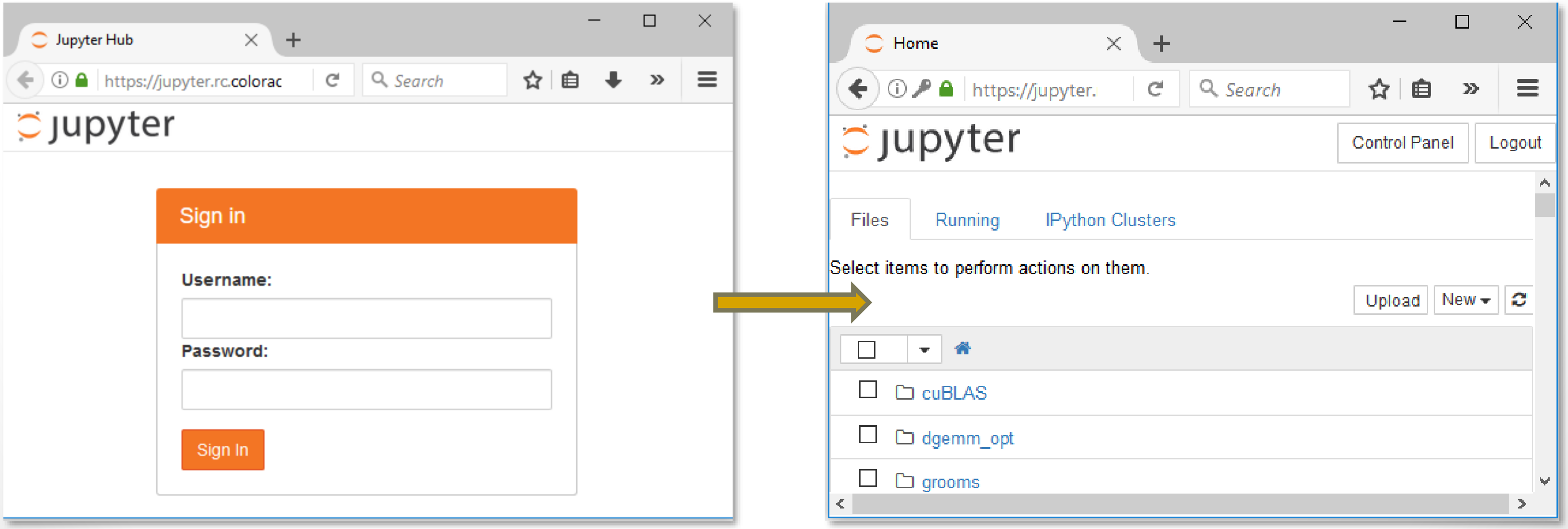
Nick Featherstone
CU Research Computing

Web Link to These Slides

Getting started…

- Login to the RC Jupyter Hub:

https :// jupyter . rc . colorado . edu



Jupyter Hub

https://jupyter.rc.colorac

Search

jupyter

Sign in

Username:

Password:

Sign In

Home

https://jupyter.

Search

jupyter

Control Panel | Logout

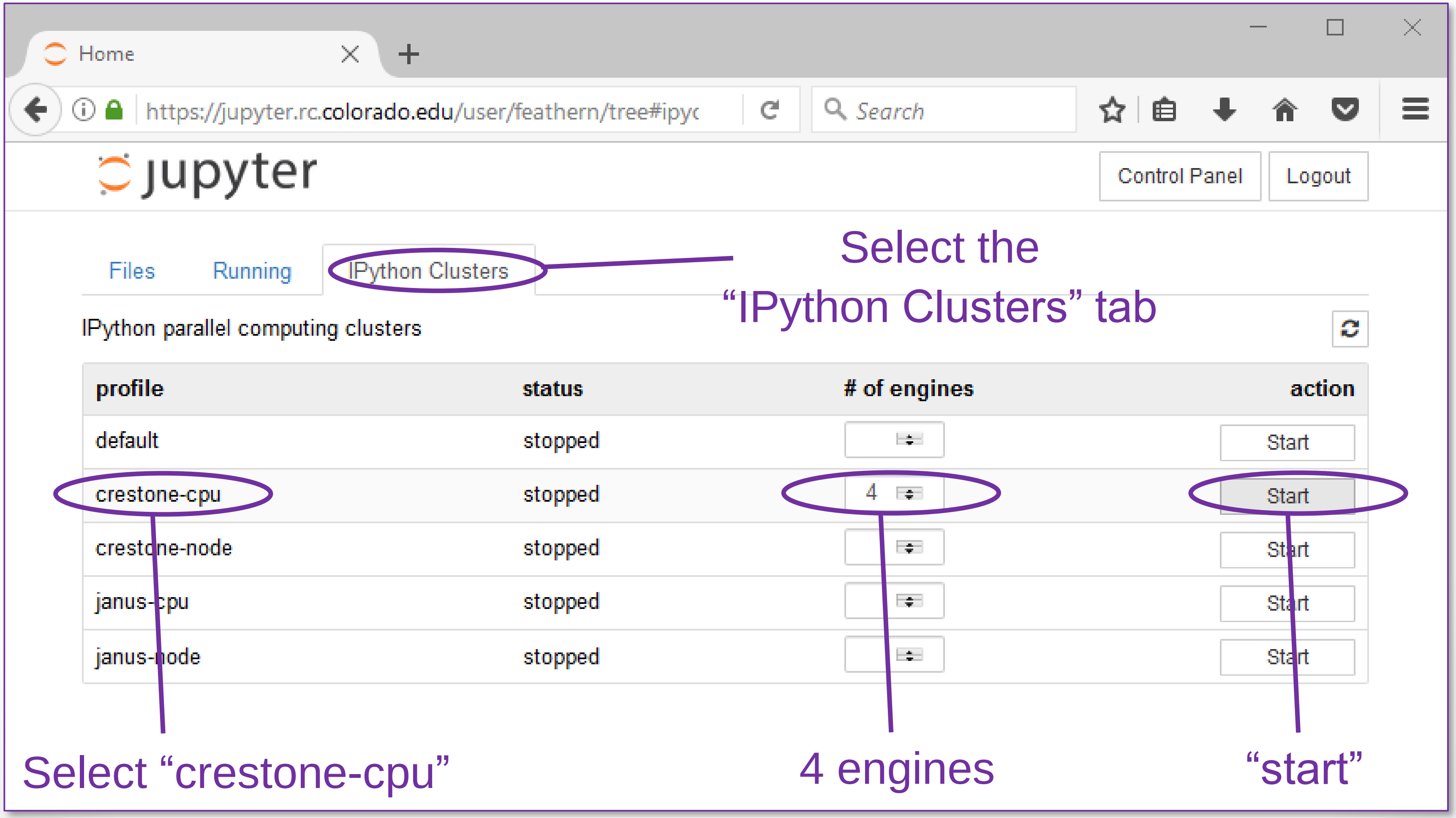
Files | Running | IPython Clusters

Select items to perform actions on them.

Upload | New ▾

- □ cuBLAS
- □ dgemm_opt
- □ grooms

Getting started…



IPython parallel computing clusters

profile	status	# of engines	action
default	stopped		Start
crestone-cpu	stopped	4	Start
crestone-node	stopped		Start
janus-cpu	stopped		Start
janus-node	stopped		Start

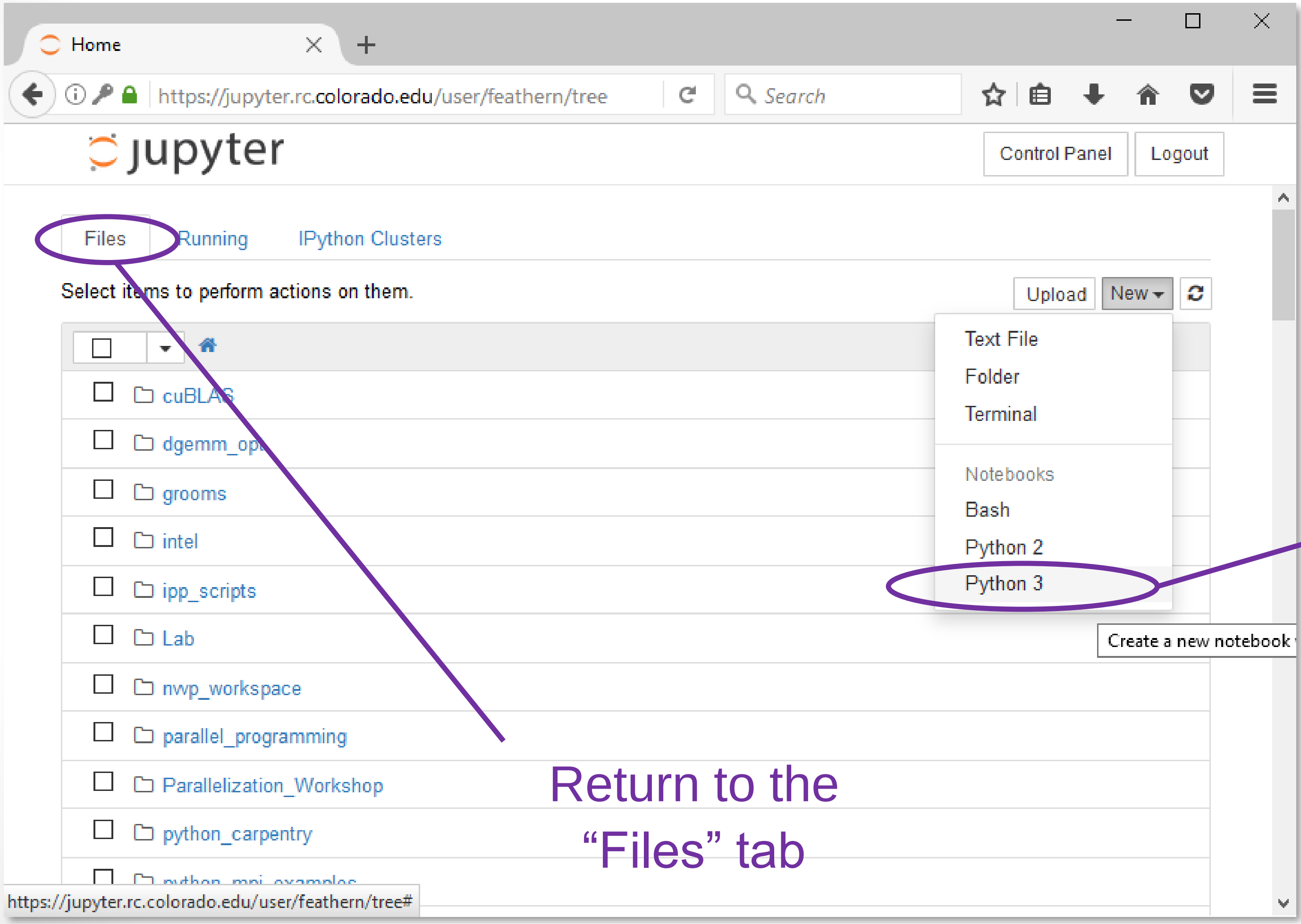
Select the “IPython Clusters” tab

Select “crestone-cpu”

4 engines

“start”

Getting started…



Select items to perform actions on them.

- □ cuBLAS
- □ dgemm_opt
- □ grooms
- □ intel
- □ ipp_scripts
- □ Lab
- □ nwp_workspace
- □ parallel_programming
- □ Parallelization_Workshop
- □ python_carpentry
- □ python_mpi_examples

https://jupyter.rc.colorado.edu/user/feathern/tree#

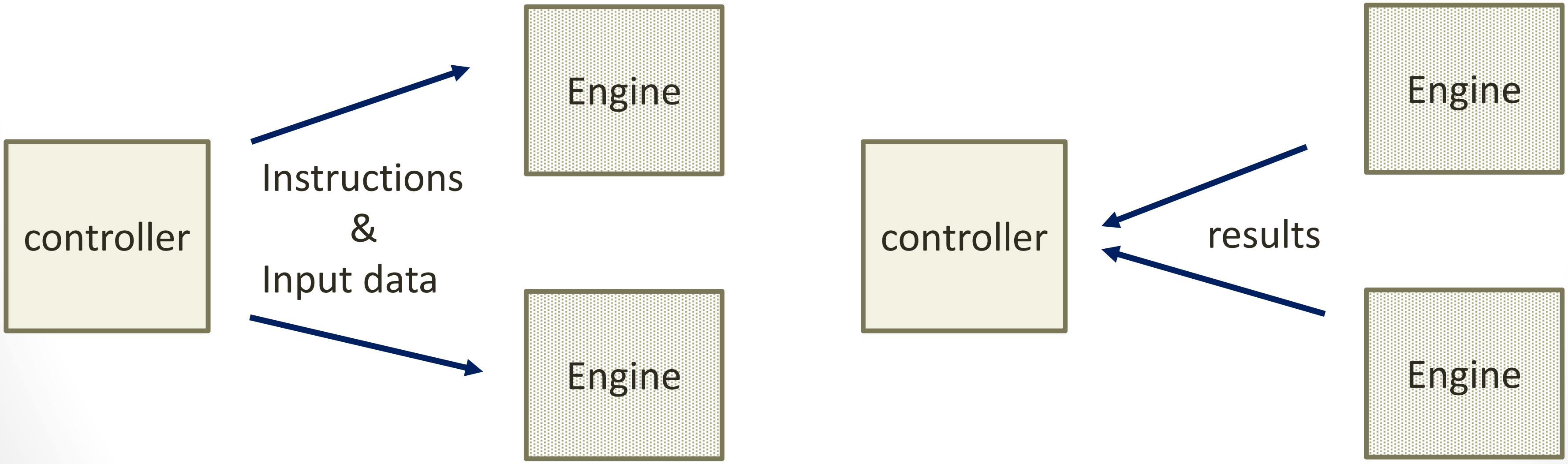
Start a Python 3 Notebook

Return to the "Files" tab

Engine/Controller Paradigm

- What did we just do?

  initiated a controller python session and 4 python engines
- **Most** of your code runs on the **controller (Jupyter notebook)**
- **Some** of your code runs on the **engines (the crestone cpu's)**



Documentation: http://ipyparallel.readthedocs.io/en/latest/multiengine.html

Hello World (1)

Most ipyparallel sessions will begin with something like:

- import ipyparallel

  … enables access to ipyparallel features
- rc=ipyparallel.Client(profile='crestone-cpu')

  … connects controller to the engines
- print(rc.ids)

  … list of engine IDs, starting with 0

Open this file:

Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / examples /

hello.py

Hello World (2)

Some other interesting pieces that we will be working with:

```
all_proc  = rc[:]
```

… “direct views” into each engine

… allows us to manipulate individual engines

```
hostnames = all_proc.apply_sync(socket.gethostname)
```

… tells all engines to call the gethostname function

… results stored in hostnames

Engine Views

The client object ***rc*** is essentially a list of engines.

We can explicitly reference individual engines or subsets of the engine pool by sampling or slicing ***rc*** as we would a list.

engines = rc[:]	view into all engines
proc0 = rc[0]	view into engine zero
even = rc[range(0,4,2)]	view into all even numbered engines

Variable Assignment

Create a variable ‘a’ on all engines:

```
engines = rc[:]
engines[‘a’] = 2
```

Create a variable ‘b’ on a engine ID 1:

```
one = rc[1]
one[‘b’] = 3
```

Initialize variable ‘c’ on ID 2 using ‘b’ from 1:

```
two = rc[2]
two[‘c’]=one[‘b’]
```

Open this file:

Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / examples /

assignment.py

Exercise 1

Modify this program so that var1 receives the value n, where n is the remainder of the **Engine ID** (0 – 3) divided by 3.
i.e., n = Engine ID % 3

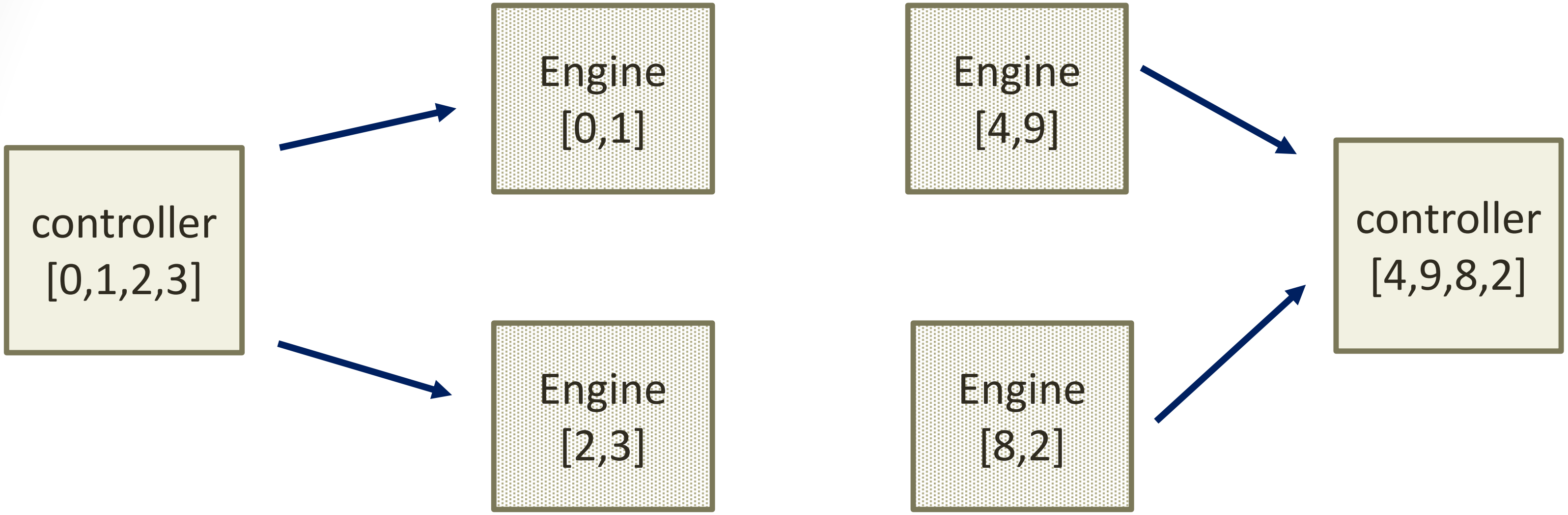
Open this file:

Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / exercises /

assignment_ex.py

Scattering & Gathering



Scatter: distribute data from one process to the group

Gather: collate data from group onto one process

Both are common operations in parallel applications.

Scattering & Gathering

Open this file:

Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / examples /

scatter_gather.py

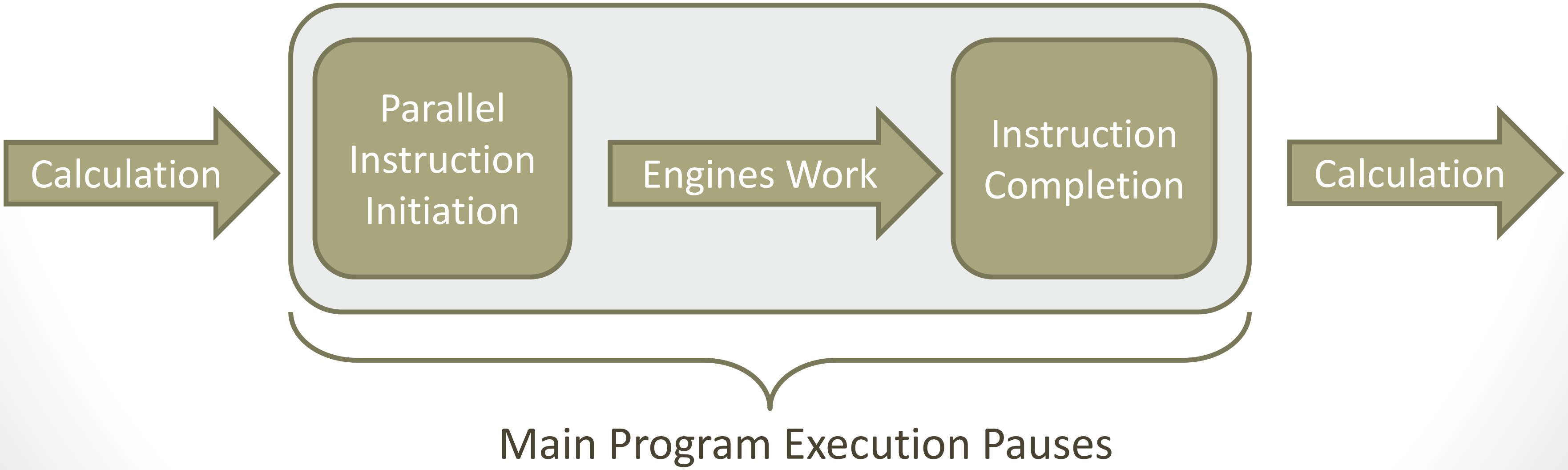
- To scatter list ‘a’ from the controller to the engines’ variable ‘mylist’:
  all_proc.scatter('mylist‘ , a)

- To view each engine’s copy of ‘mylist’:
  sub_lists = all_proc['mylist‘]

- To gather back from the engines to the controller:
  gathered = all_proc.gather('mylist‘)

Quick Note: Blocking

- You may have noticed: all_proc.block=True
- This tells the engines and the controller to wait until parallel instructions have completed before resuming the code execution


Calculation
Parallel Instruction Initiation
Engines Work
Instruction Completion
Calculation
Main Program Execution Pauses

Exercise 2

- Scatter list ‘a’ to all even processors, assigning its distributed values to the variable ‘mylist’ on each engine.
- Similarly, scatter list ‘b’ to all odd processors’ ‘mylist’ variable
- Gather from all processors to create the list [0,1,0,1,2,3,4,9,....]

Open this file:

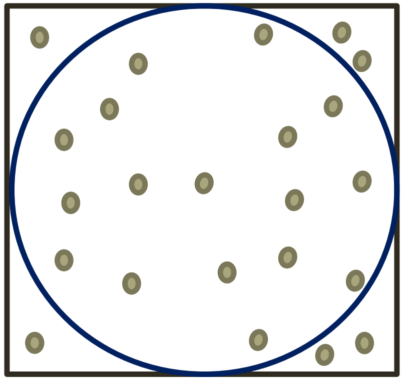
Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / exercises /

scatter_gather_ex.py

Who wants some Pi?

- Estimating pi is straightforward
- Compute ***n*** random coordinates in the domain: -1 < (x,y) < 1
- Count the number of coordinates ***m*** that fall within a unit circle centered on the origin
- Estimate = m/n



Function Evaluation via Map_Sync

- IPyParallel has an analog to map: map_sync

```
results= all_proc.map_sync(function_name, list_of_arguments)
```

- “results” contains a list of results from each process
- argument list is distributed (scattered) among all processes

Open this file:

Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / examples /

compute_pi.py

Exercises 3 & 4

- Try these exercises (instructions provided in each file):

Open this file:

Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / exercises /

parallel_functions.py
collatz.py

Classes in IPyParallel

- Class definitions exist soley on the controller/hub at first
- Definitions must be “pushed” to the engines before use:

```
all_proc.push(“engine class name” : controller class name)
```

Open this file:

Parallelization Workshop / Day3-Parallel_Python /

session2_ipyparallel / examples /

push.py