Fine Grained Resource Management for Functions in Parsl and Work Queue

Douglas Thain
University of Notre Dame
3 October 2019

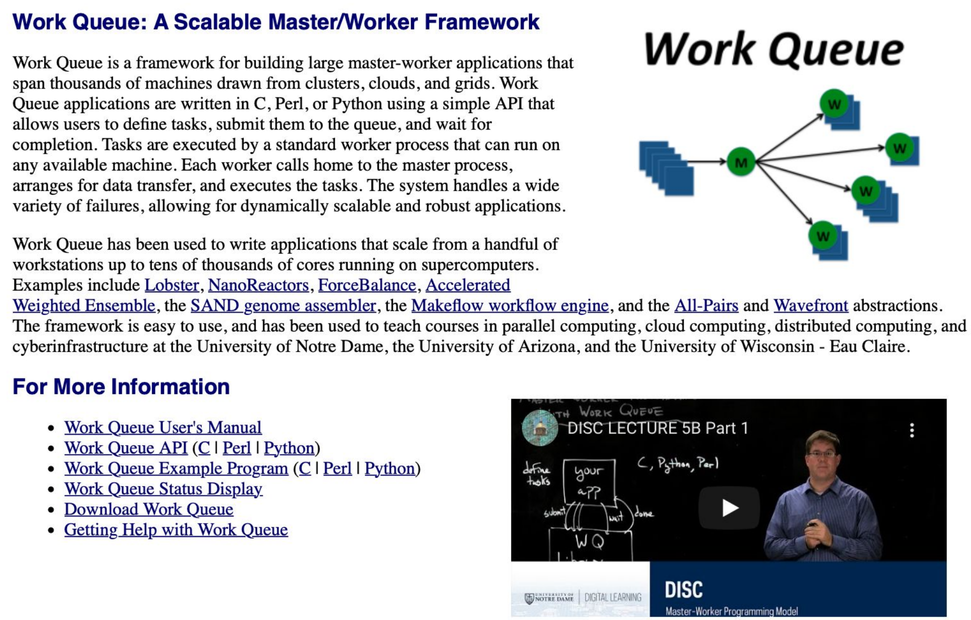
CCTools

Parsl + Work Queue for Scalable Apps

http://parsl-project.org

Powerful Pythonic Workflow Programming Model

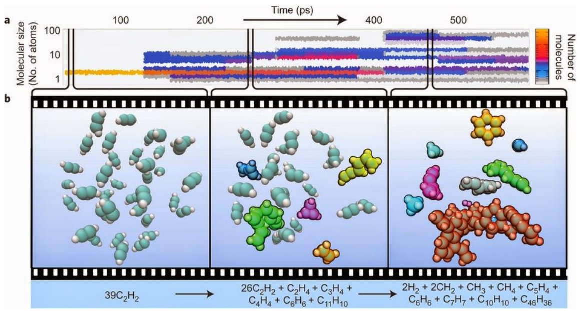
http://ccl.cse.nd.edu



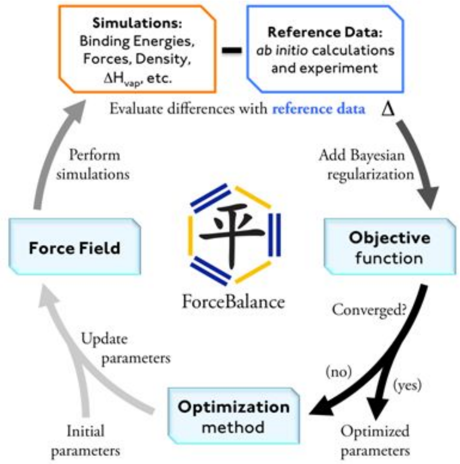
Scalable, Portable, Robust Distributed Execution

Some Work Queue Applications

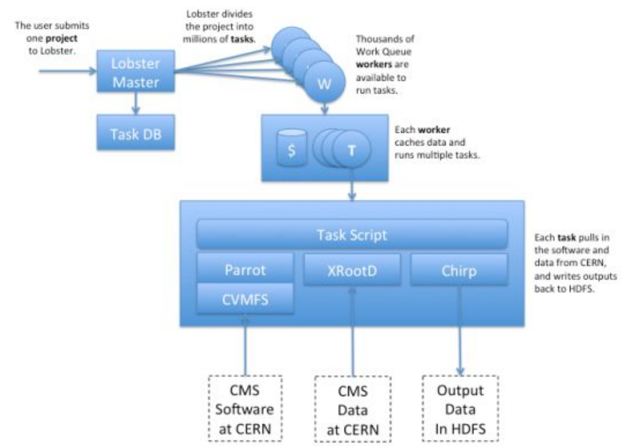
Nanoreactors
ab-initio Chemistry



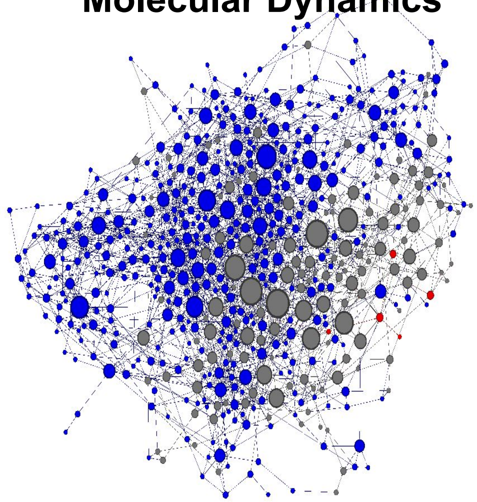
ForceBalance
FF Optimization



Lobster
CMS Data Analysis



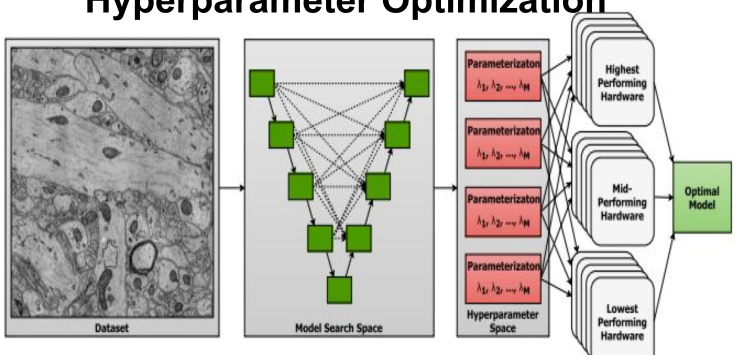
Adaptive Weighted Ensemble
Molecular Dynamics



Low-Level API:

task = create(details);
submit(task);
task = wait(timeout);

SHADHO
Hyperparameter Optimization



Peter Bui, Dinesh Rajan, Badi Abdul-Wahid, Jesus Izaguirre, Douglas Thain,
Work Queue + Python: A Framework For Scalable Scientific Ensemble Applications,
Workshop on Python for High Performance and Scientific Computing (PyHPC) at Supercomputing 2011.

Work Queue Capabilities

Elastic. Workers can be added and removed during runtime, and the manager automatically uses the workers available.

Robust. Tasks running on workers that fail are automatically detected and handled elsewhere.

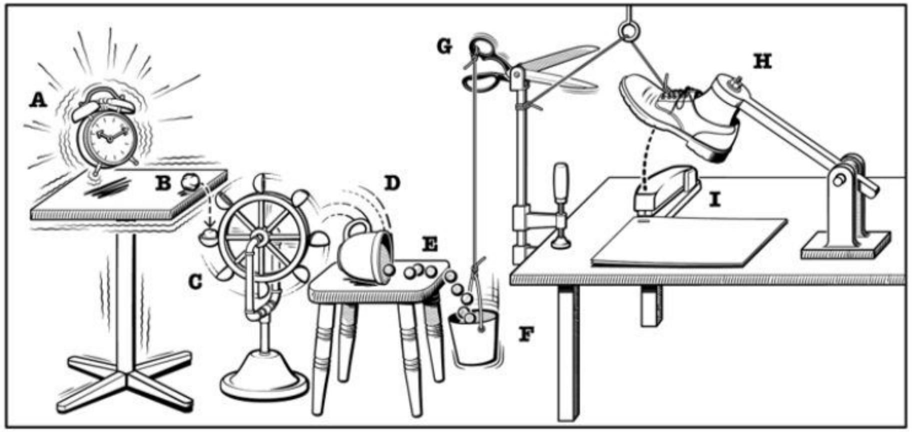
Data Management. Files may be cached at the workers, which reduces transfer times and network utilization. (No shared FS)

Resource Management. Resources such as core, memory, and disk are tracked and limited, so both tasks and workers can be heterogeneous.

Language Agnostic. Workers may run in campus cluster, national labs, or commercial cloud facilities. Managers can be written in Python, Perl, or C. (SWIG/JSON bindings for more)

So What's New?

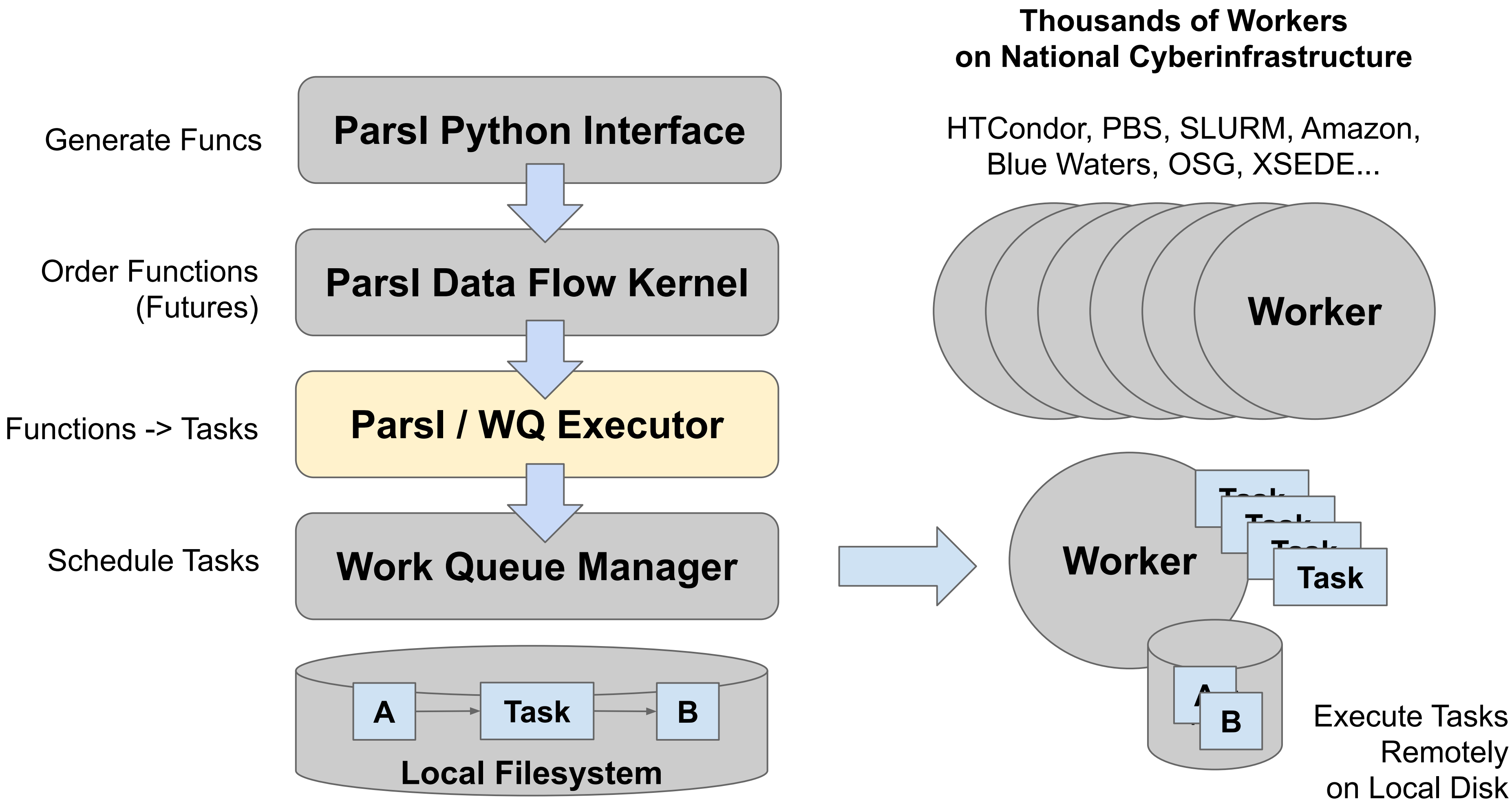
Last Year: 2019



This Year: 2020

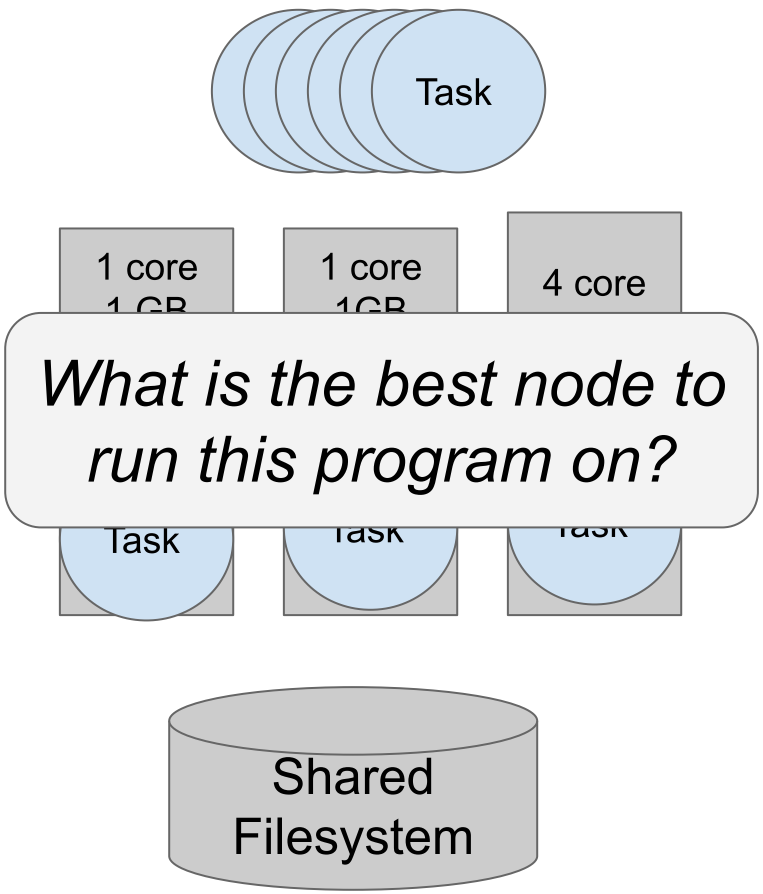
```
$ conda install -y -c conda-forge cctools parsl
```

System Architecture

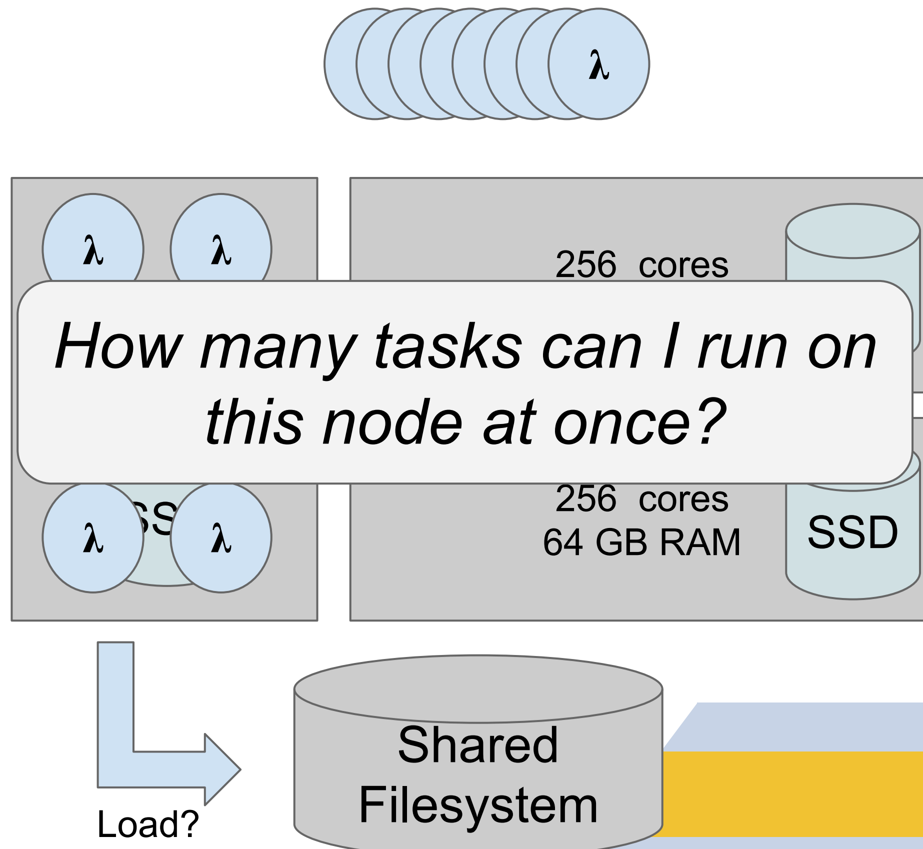


Evolution of Batch Computing

"Classic" Batch Computing:
One Process per Node



Manycore Cluster Computing:
Multiple (Small) Functions per Node



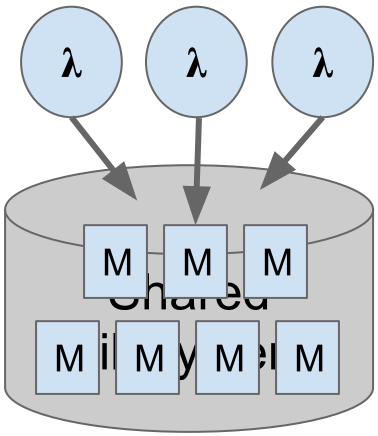
Pop Quiz!

What are the two most terrifying words in the Python language?

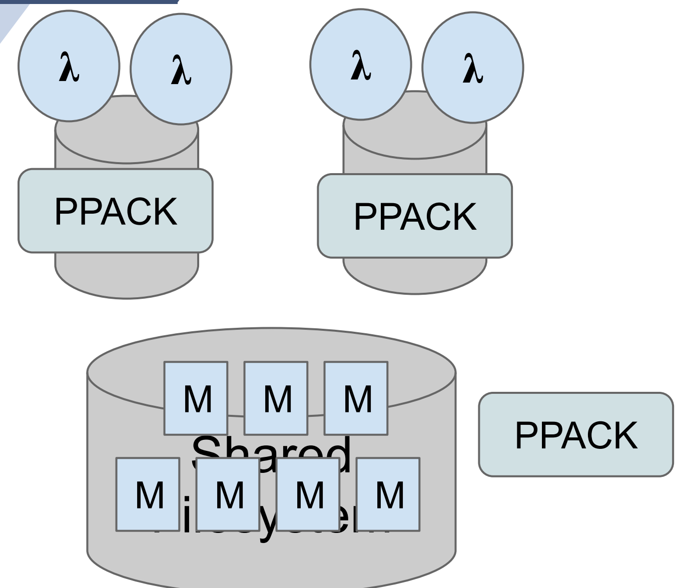
import tensorflow

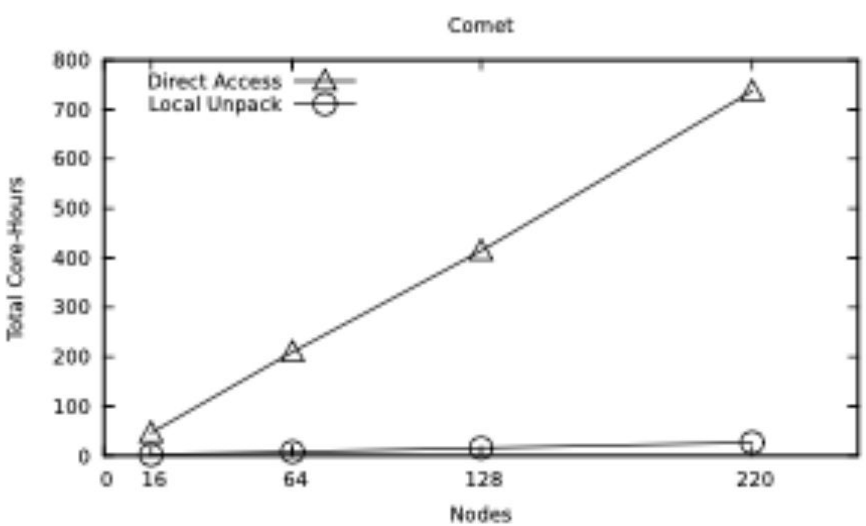
Challenge 1: Transporting Python Environments

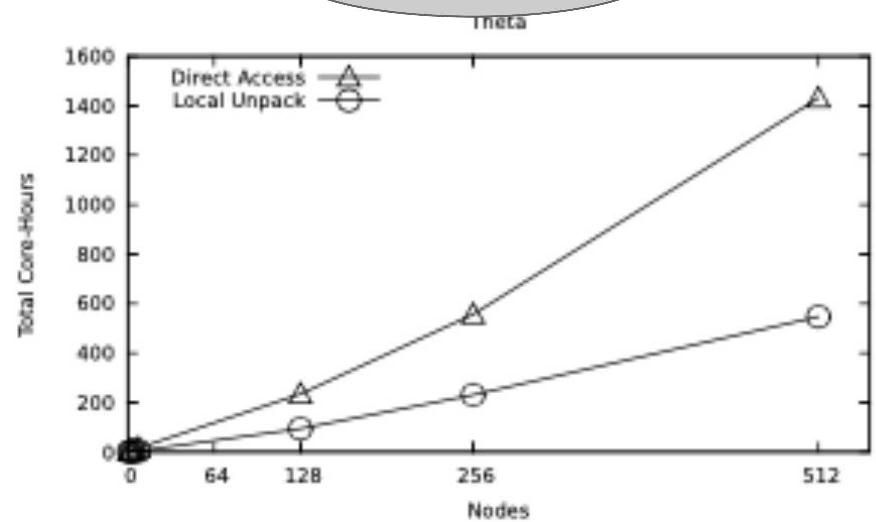
Direct Access



Local Unpack

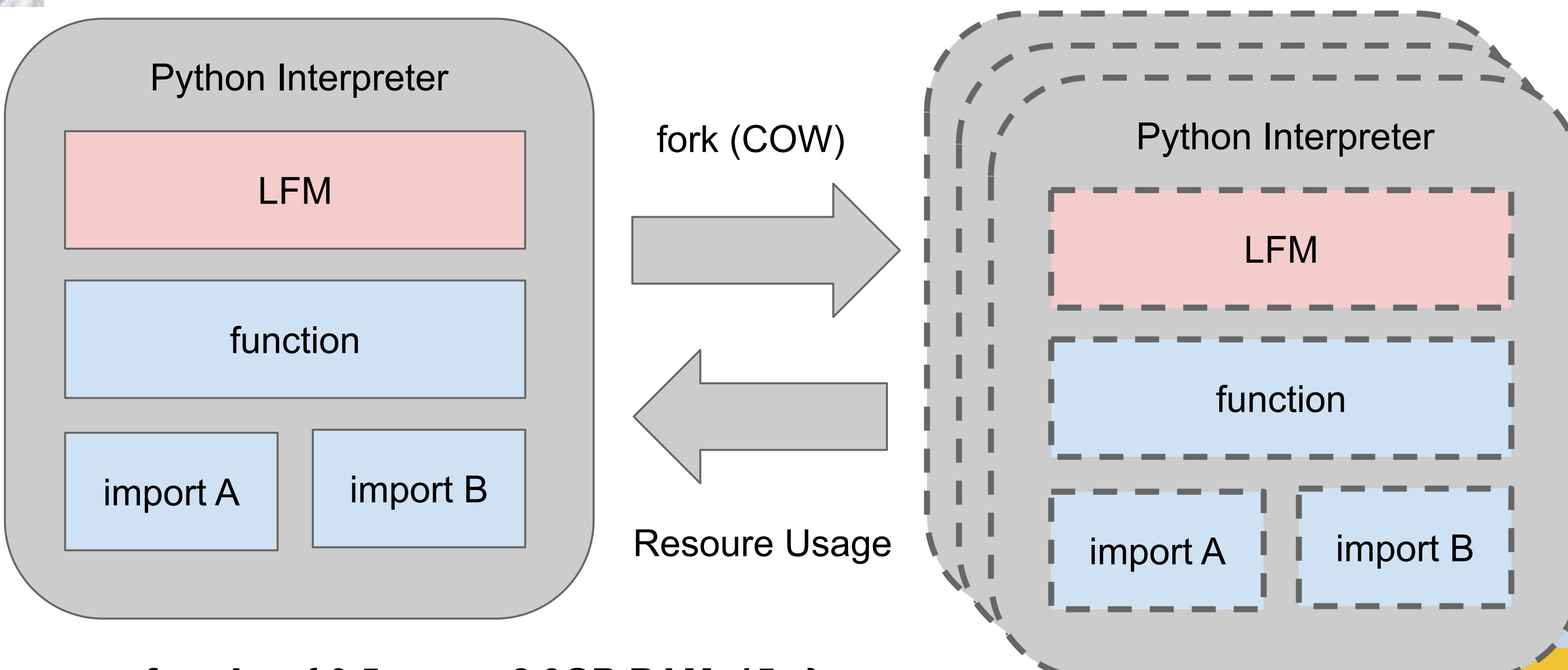






Challenge 2: How many functions per node?

We must be able to measure a single function call!
LFM - Lightweight Function Monitor



funcA -> { 0.5 cores, 2.3GB RAM, 15s }
funcB -> { 2.1 cores, 0.9GB RAM, 9s }

Lightweight Function Monitors (LFMs)

Activate LFMs with an import and the @monitored keyword

In [7]:
```
from resource_monitor import monitored
from time import sleep
```

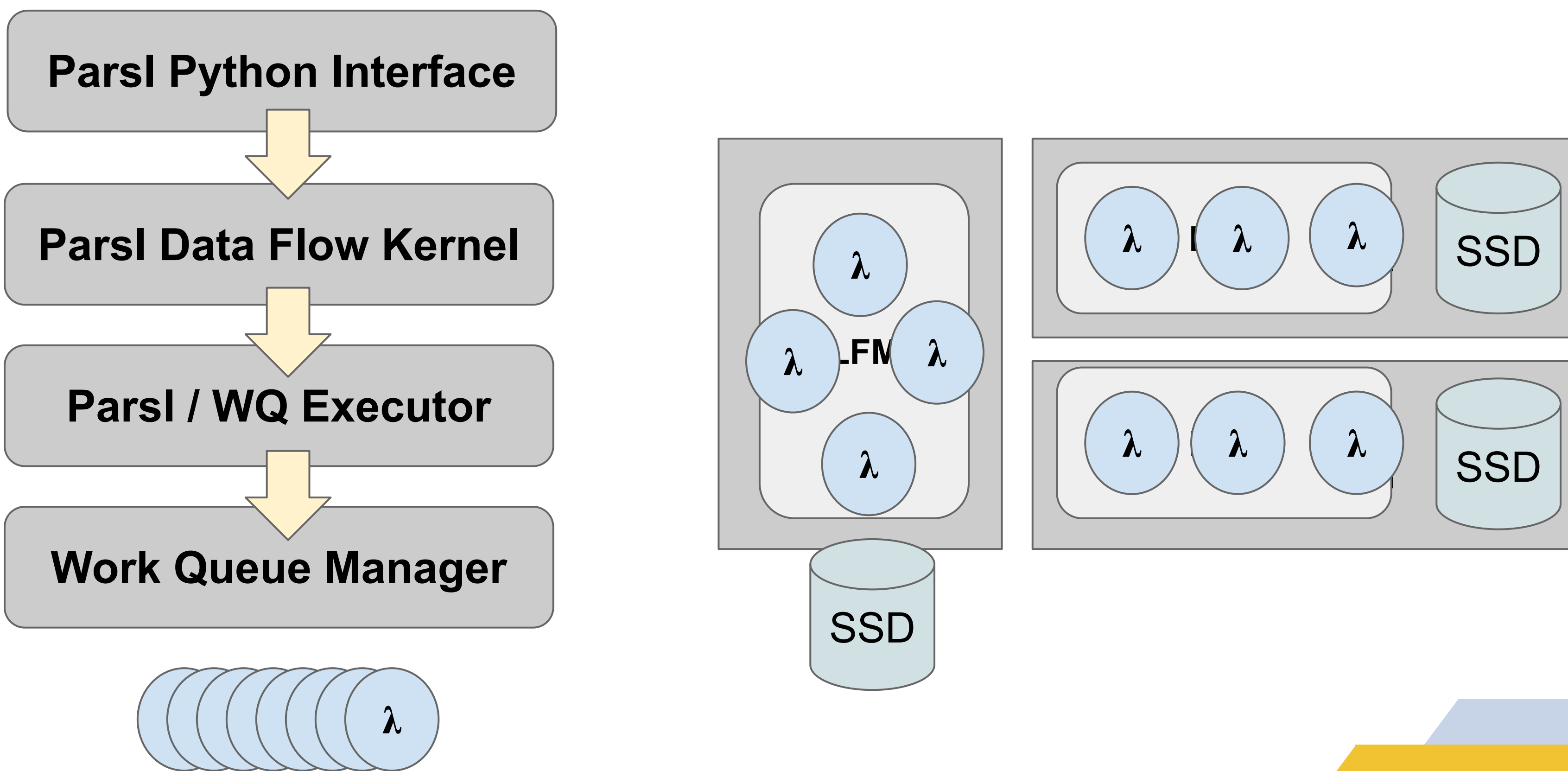
In [12]:
```
# declare a function to be monitored with the @monitored() decorator

@monitored()
def my_function_1(wait_for):
    sleep(wait_for)
    return 'waitied for {} seconds'.format(wait_for)

(result, resources) = my_function_1(.1)
print(result, '{}'.format({'memory': resources['memory'], 'wall_time': resources['wall_t
```

```
waitied for 0.1 seconds {'memory': 49, 'wall_time': 101689}
```

Putting it All Together



Tutorial Later This Afternoon

Fine-grained Management of Resources with WorkQueue

Zhuozhao Li
U. Chicago

Tim Shaffer
U. Notre Dame

To utilize Work Queue with Parsl, please install the full CCTools software package within an appropriate Anaconda or Miniconda environment (instructions for installing Miniconda can be found here):

```
$ conda create -y --name <environment> python=<version> conda-pack
$ conda activate <environment>
$ conda install -y -c conda-forge cctools parsl
```

This creates a Conda environment on your machine with all the necessary tools and setup needed to utilize Work Queue with the Parsl library.

The following snippet shows an example configuration for using the Work Queue distributed framework to run applications on remote machines at large. This examples uses the `WorkQueueExecutor` to schedule tasks locally, and assumes that Work Queue workers have been externally connected to the master using the work_queue_factory or condor_submit_workers command line utilities from CCTools. For more information on using Work Queue or to get help with running applications using CCTools, visit the CCTools documentation online.

```
from parsl.config import Config
from parsl.executors import WorkQueueExecutor

config = Config(
    executors=[
        WorkQueueExecutor(
            label="wqex_local",
            port=50055,
            project_name="WorkQueue Example",
            shared_fs=True,
            see_worker_output=True
        )
    ]
)
```

Contributors

U. of Notre Dame: Tim Shaffer, TJ Dasso, Andrew Litteken,
Tanner Juedeman, Ben Tovar

U. Chicago: Zhuozhao Li, Yadu Babuji , Ben Clifford,
Anna Woodard, Kyle Chard, Ian Foster

NSF Grant #ACI-1642409 (SI2-CCTools)
NSF Grant #OAC-1931387 (CSSI Dataswarm)
DOE SGCSR Fellowship (Shaffer)

http://ccl.cse.nd.edu

http://parsl-project.org

Work Queue