Many-task Computing with Python

Monte Lunacek
Jazek Braden
Thomas Hauser

Research Computing, University of Colorado **Boulder**

What is many-task computing?

MTC, Raicu *et al.*

- set of tasks
  - may have dependencies
  - not neccessarily single-core
- use a large resources over a short period

High-throughput Computing (HTC)

- executed over a period of months
- operations per month

Why Python?

- flexible, powerful programming language
- easy to learn
- accessible to *non-programmers*
- glue
- free and open
- large community

University of Colorado

- Some tightly-coupled MPI codes
- Many **independent tasks** (MTC)
- Diverse computing backgrounds
  - Geography
  - Ecology and Evolutionary Biology
  - Microbial Ecology
  - Astronomy
  - Geology
- Range of computational experience

Supercomputing Without the Pain

- Accessible to anyone with:
  - Simulation or analysis to run
  - Desire to do it faster
- Remove barriers to entry

Outline

MTC

Solutions, **mpi4py** example

Message queues

- **IPython**
- **Celery**
- Fault-tolerance, elasticity, memory

Scaling Results

Conclusions

Many-task Computing

```
def work(x):
    import time, os
    start_time = time.time()
    time.sleep(x)
    end_time = time.time()
    return {'pid': os.getpid(),
            'start_time': start_time,
            'end_time': end_time}
```

Monte Carlo simulations

Parameter scan

Uncertainty Quantification

Parameter Optimization

Approach

Condor/Moab/SLURM

Bash/pbsdsh

MPI **mpi4py**

Message broker: **IPython, Celery**

SAGA BigJob

and many, many more ...

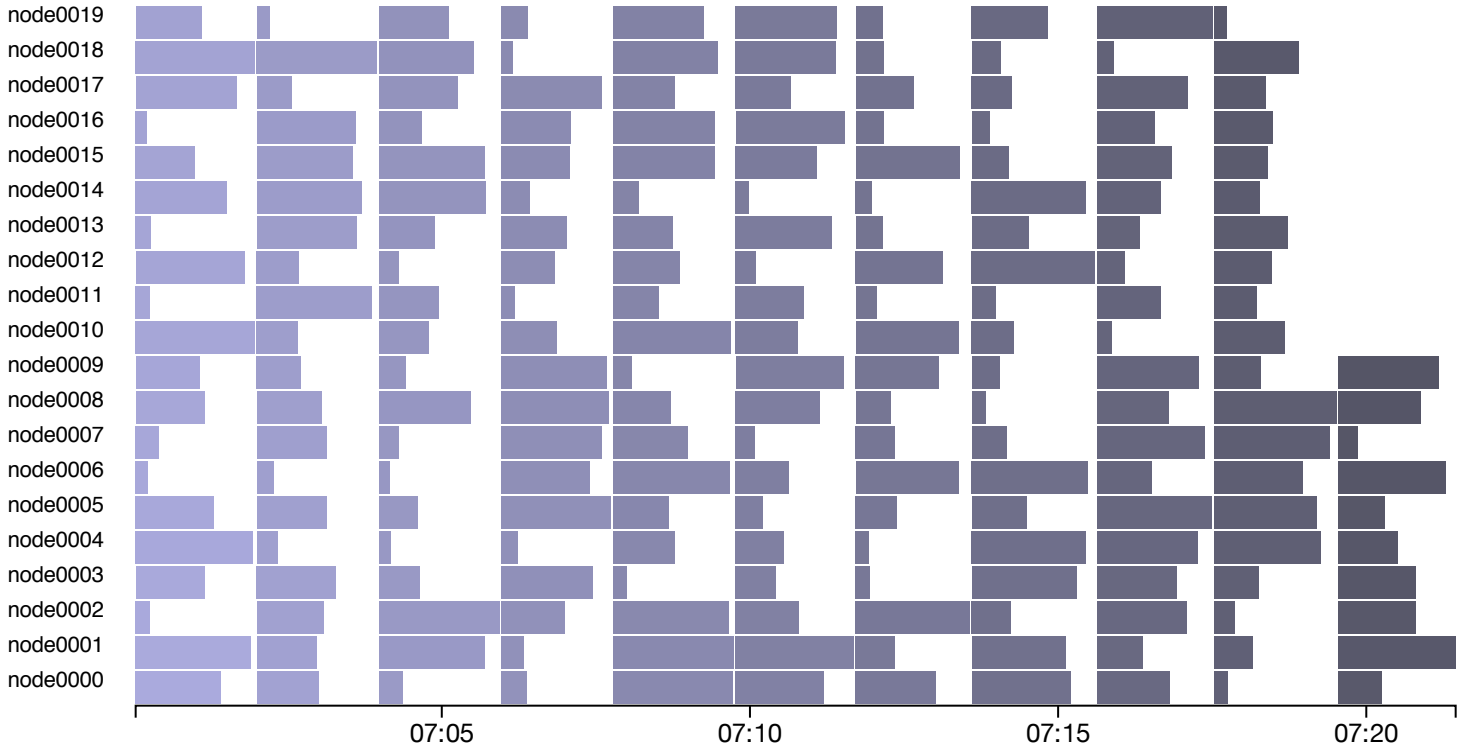
Bash/pbsdsh

```
#!/bin/bash
PATH=$PBS_O_WORKDIR:$PBS_O_PATH
TRIAL=$(($PBS_VNODENUM + $1))
python work.py 5
```

```
for i in {1..N}
do
  pbsdsh wrapper.sh $count
  count=$(( $count + 12))
done
```

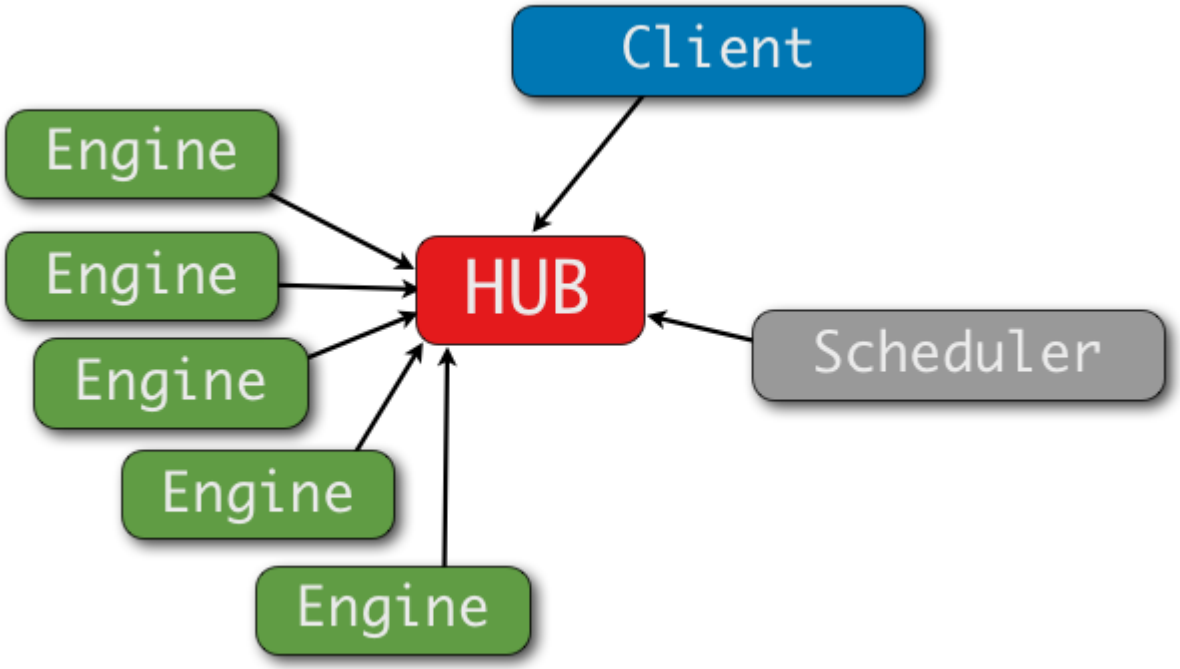
A little **painful**

A little inefficient



MPI example

Message queues



fault-tolerance, elasticity, memory

Weak Scaling

Compare: mpi4py, IPython Parallel, Celery

What recommendations can we offer?

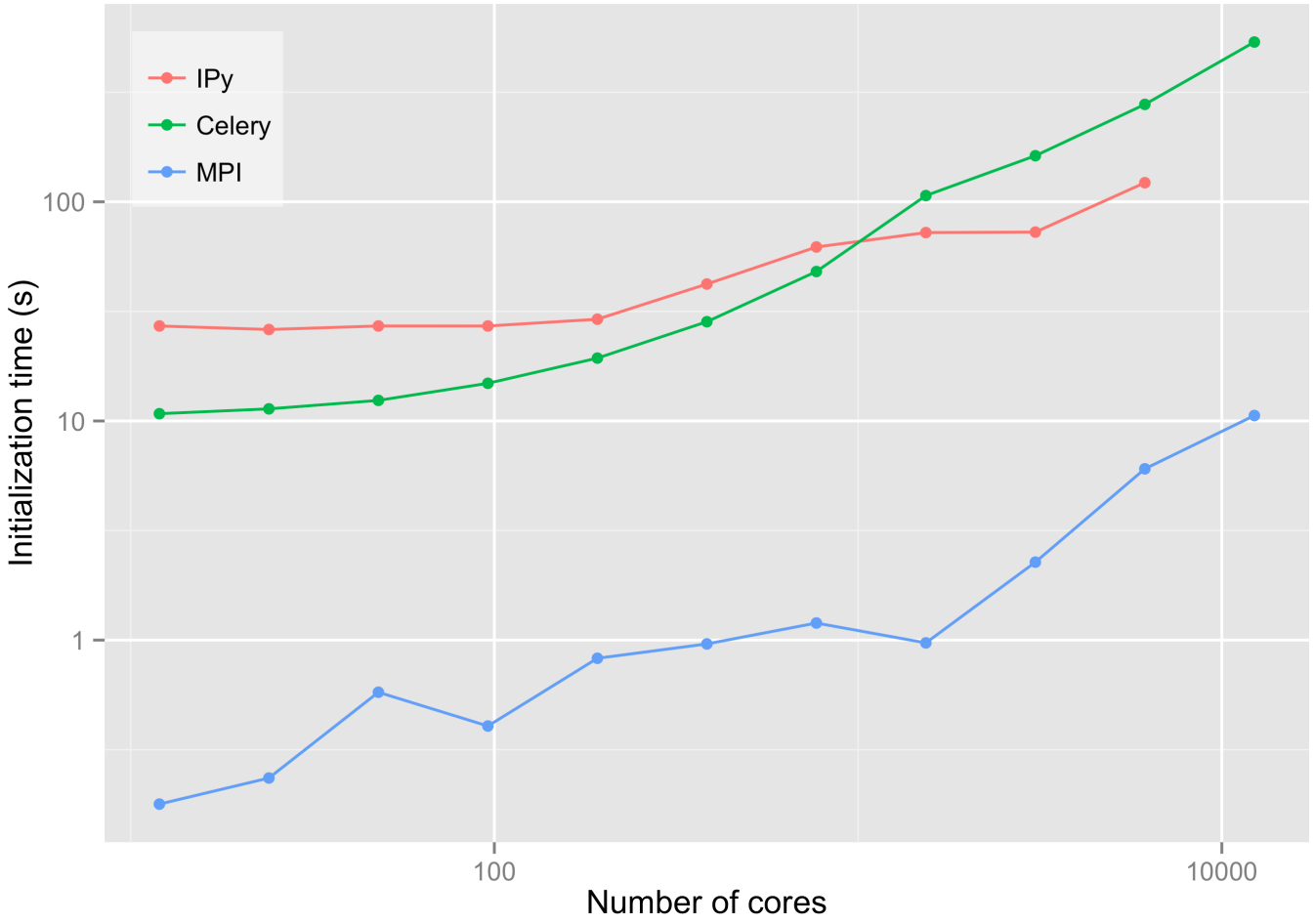
Best case scenario

- small messages
- limited file IO
- sleep

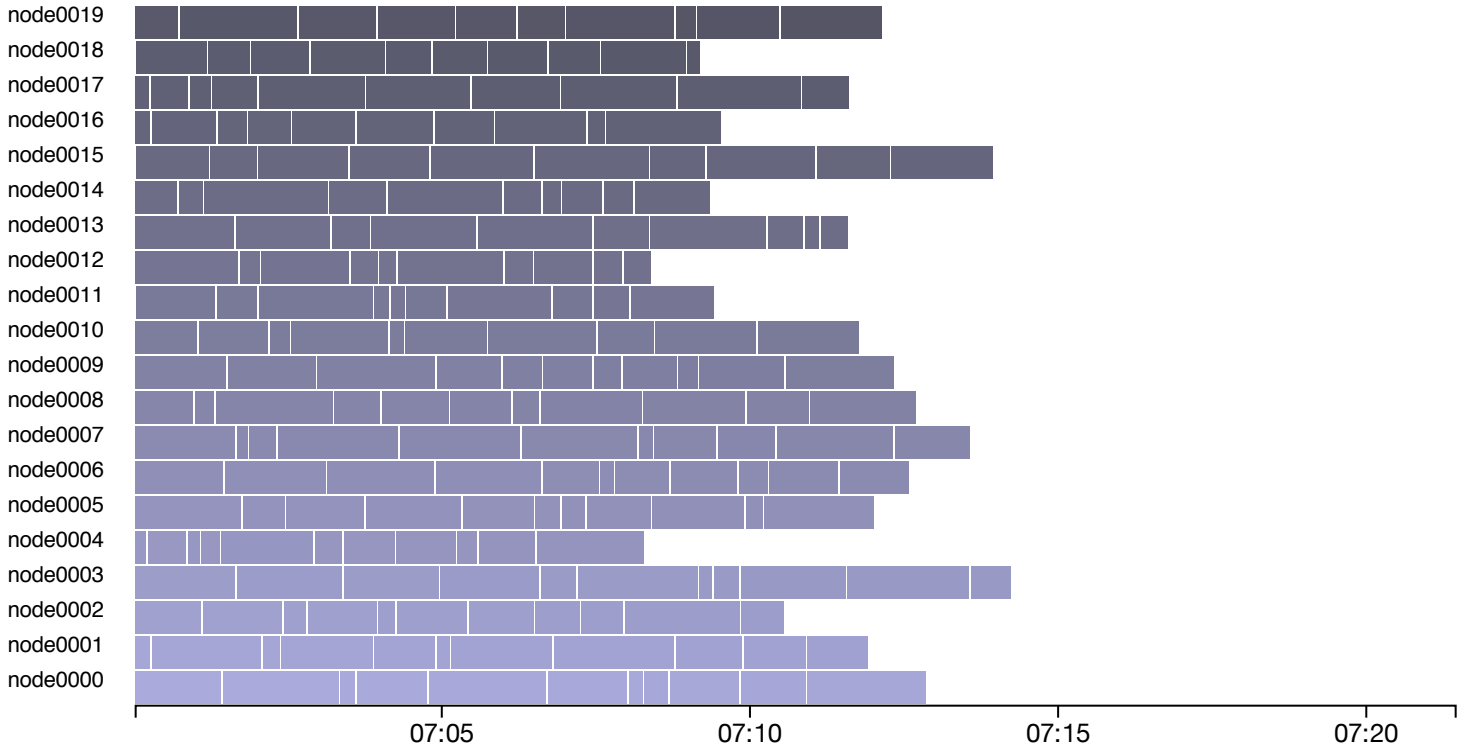
Weak scaling

- task = 10 * number of cores
- up to 12,288 cores
- time uniform(27,33) seconds

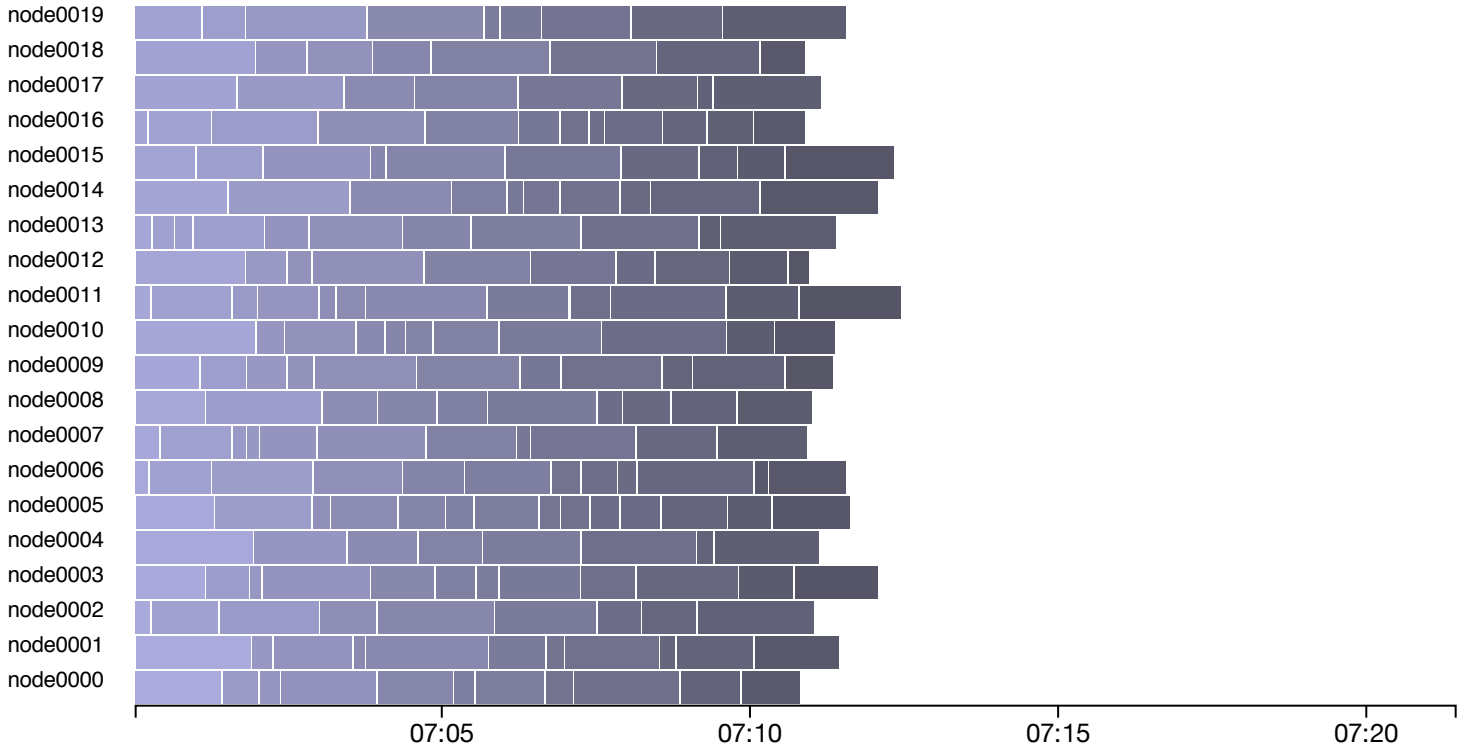
Initialization



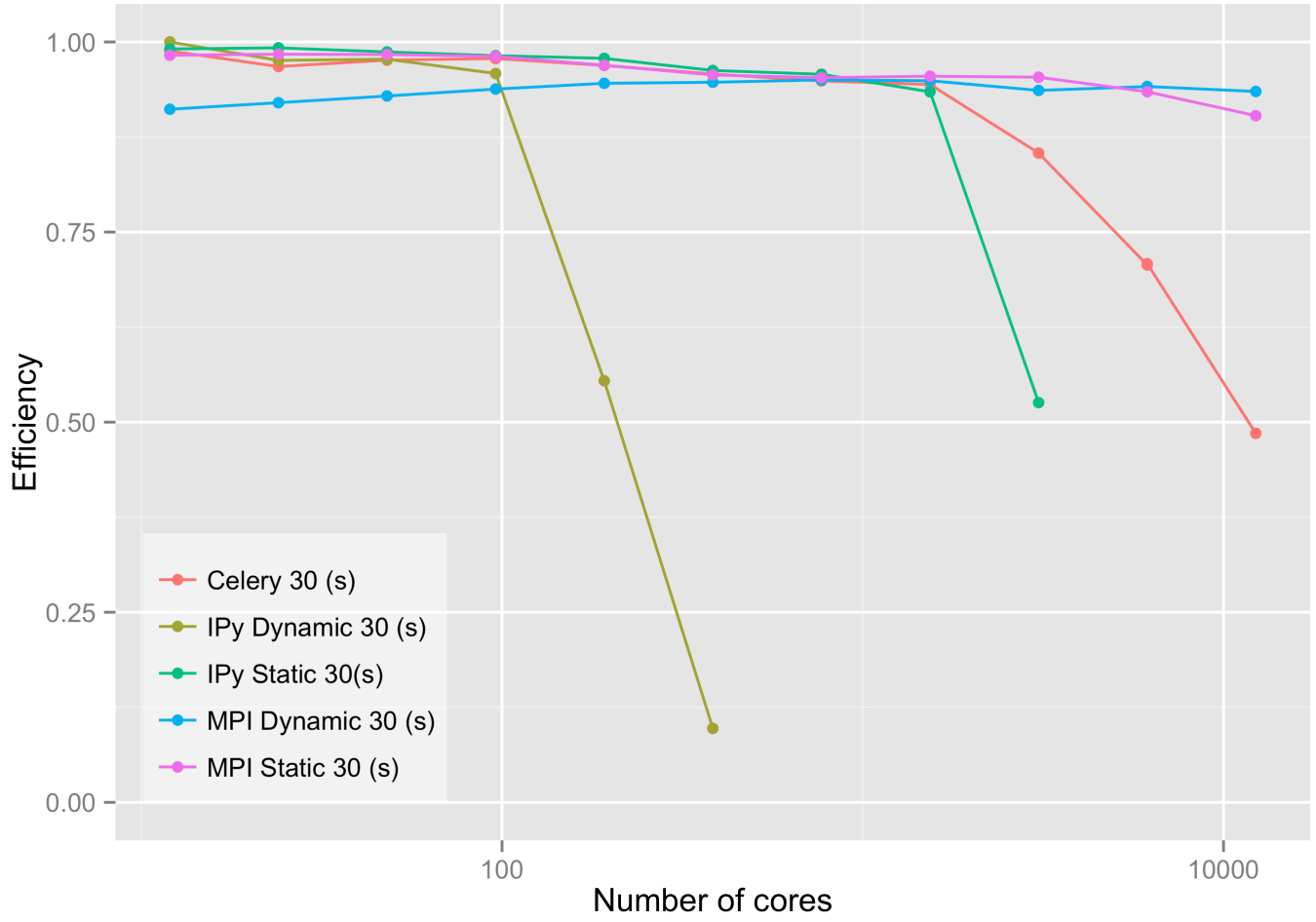
Scheduling: Static



Scheduling: Dynamic



Weak Scaling



Compare

Recommend

mpi4py	many cores, robust simulation
IPython	100 cores, not robust
IPython	many cores, not robust, consistent time
Celery	many cores, not robust, variable time
Multiprocessing	single-node

Issues

IPython, Celery	launching
all	user interface

https://github.com/mlunacek/load_balance_ipython

(https://github.com/mlunacek/load_balance_ipython)

Conclusions

Python is an excellent way to manage MTC jobs

IPython and Celery

- elastic
- memory
- **fault-tolerant**

Moving forward

- Simple interface for MTC
- Profiling tools
- Realistic tasks

References

Paper

Scaling of Many-Task Computing Approaches in Python on Cluster Supercomputers

Monte Lunacek *et al.* IEEE Cluster 2013

Tutorial

tomorrow: 8:30 - 10:15, room 9

slides

https://github.com/mlunacek/python_frcrc_2013

links

Distributions: Enthought (http://www.enthought.com/products/epd.php), Anaconda (https://store.continuum.io/cshop/anaconda)

Packages: IPythoParallel (http://ipython.org/ipython-