The architecture for running Parsl multi-site workflows on the Parallel Works platform

Stefan Gary, Alvaro Vidal Torreira, Matthew Shaxted, Michael McQuade, Quan Nguyen, Louis Le, Matt Long, Michael Wilde

Parallel Works

SBIR · STTR
America's Seed Fund

ParslFest
October 19, 2023

sfgary@parallelworks.com

Motivation

What is a multi-site workflow?

```
python_app(my_func, executors=[label])(...)
```

Applies to:

different partitions on the same cluster or

different clusters at different sites/clouds

Why is a multi-site workflow useful?

Collaboration/portability with other teams

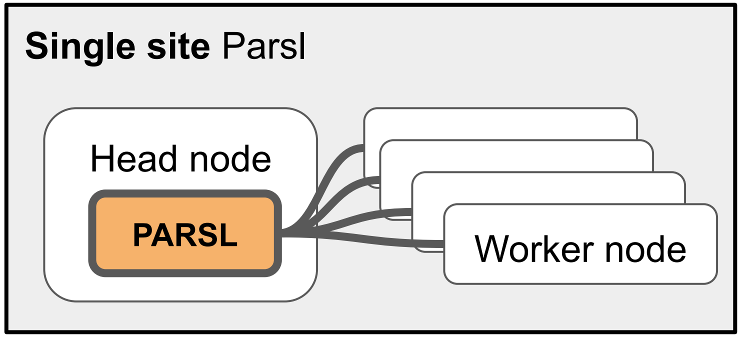
Changes cloud hardware or performance:cost

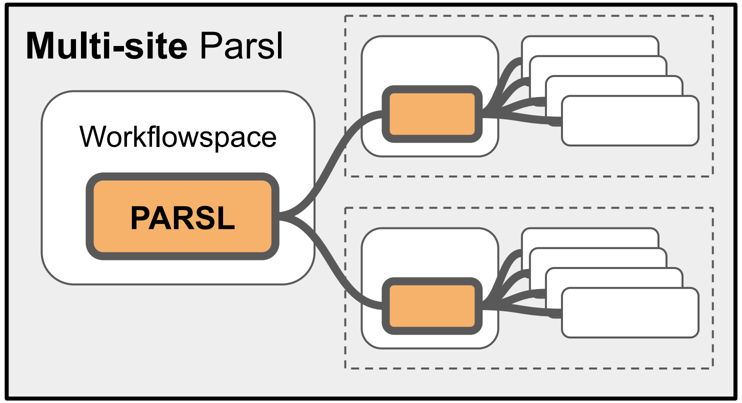
Modern HPC is **not** monolithic - some jobs run best on different resources, e.g.

data-compute proximity

licensed software

CPU/GPU





Context

What does Parallel Works do?

Configure, start cloud clusters

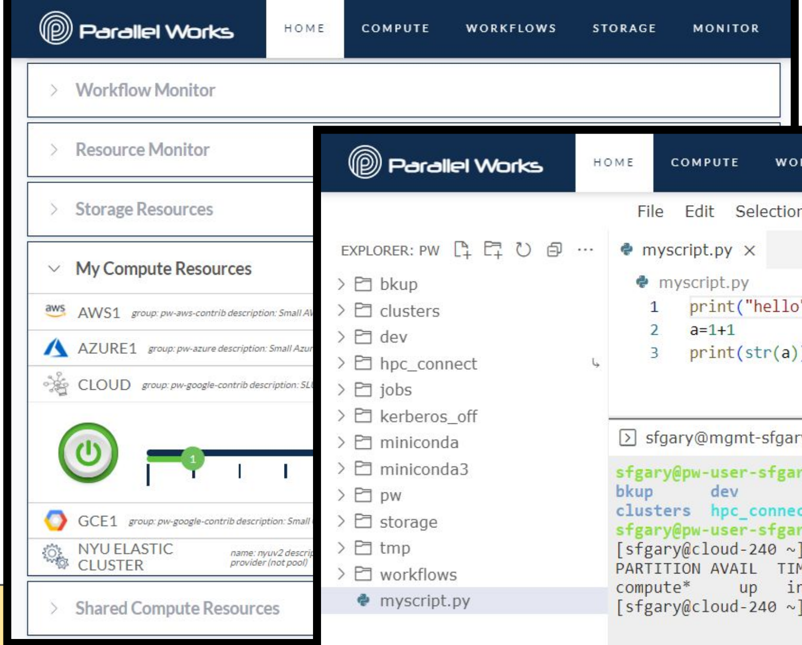
Same feel on the major CSPs

IDE (cloud terminal, workspace)

GUI, CLI, API launch Parsl workflows

Collaborates for demonstration stages of R&D SBIR grants; partner with academic teams for real-world, funder-relevant testbeds.

Graphics from cluster



Parallel Works — HOME COMPUTE WORKFLOWS STORAGE MONITOR

Workflow Monitor

Resource Monitor

Storage Resources

My Compute Resources

AWS1 group: pw-aws-contrib description: Small A

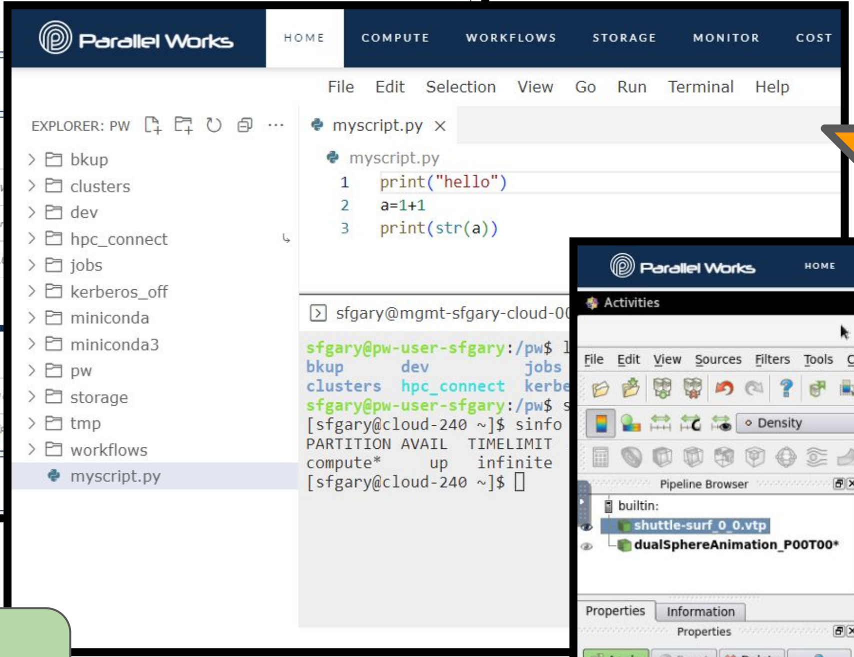
AZURE1 group: pw-azure description: Small Azur

CLOUD group: pw-google-contrib description: SL

GCE1 group: pw-google-contrib description: Small

NYU ELASTIC CLUSTER name: nyuv2 descri provider (not pool)

Shared Compute Resources



Parallel Works — HOME COMPUTE WORKFLOWS STORAGE MONITOR COST

File Edit Selection View Go Run Terminal Help

EXPLORER: PW

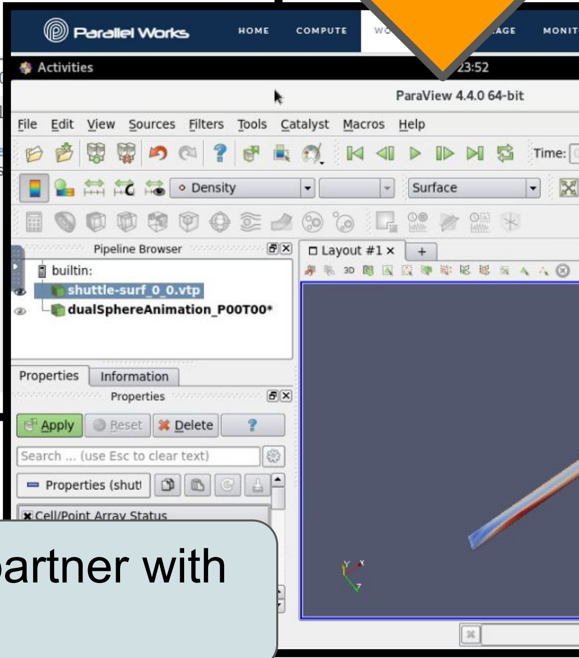
bkup, clusters, dev, hpc_connect, jobs, kerberos_off, miniconda, miniconda3, pw, storage, tmp, workflows, myscript.py

myscript.py

```
print("hello")
a=1+1
print(str(a))
```

sfgary@mgmt-sfgary-cloud-0

```
sfgary@pw-user-sfgary:/pw$ l
bkup        dev          jobs
clusters    hpc_connect  kerbe
sfgary@pw-user-sfgary:/pw$ s
[sfgary@cloud-240 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT
compute*     up   infinite
[sfgary@cloud-240 ~]$
```



Parallel Works — HOME COMPUTE

Activities 23:52

ParaView 4.4.0 64-bit

File Edit View Sources Filters Tools Catalyst Macros Help

Density — Surface

Pipeline Browser

builtin:
shuttle-surf_0_0.vtp
dualSphereAnimation_P00T00*

Properties Information

Apply Reset Delete

Search ... (use Esc to clear text)

Properties (shutt

Cell/Point Array Status

Layout #1

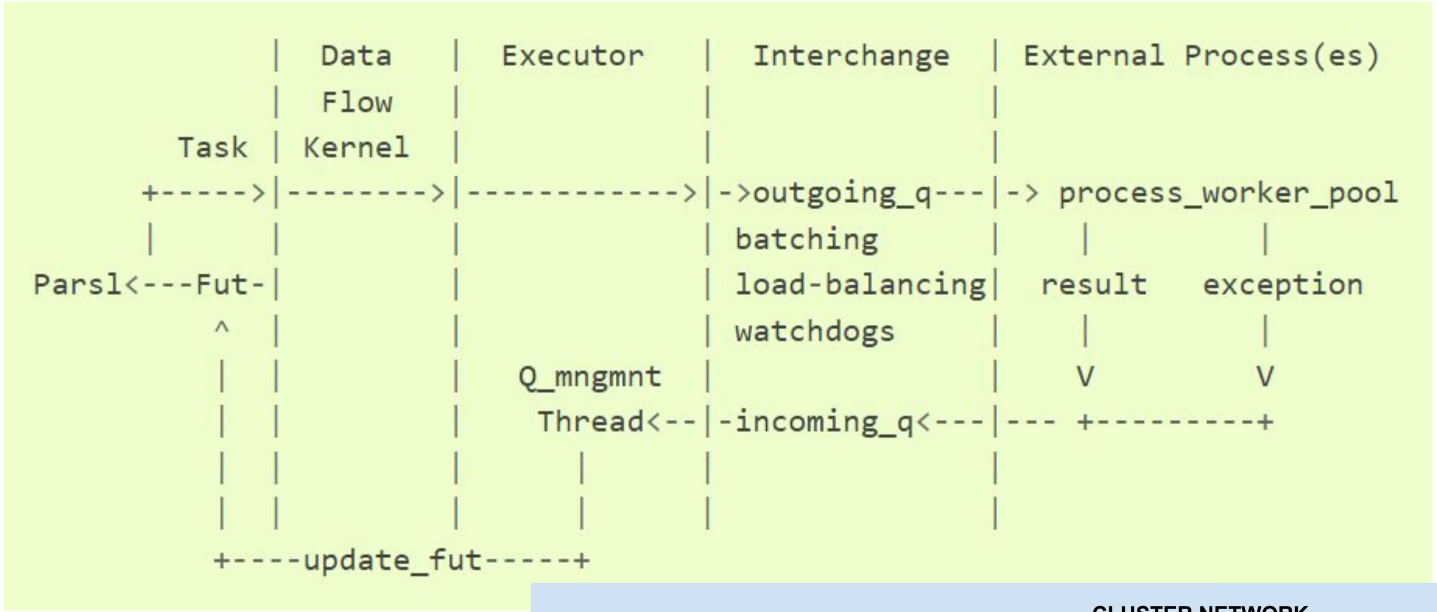
Typical multi-site workflow stumbling blocks

- I'm *guessing* that most users run Parsl scripts on the head node of an on-premise cluster; as such, for most users, the resource is:
  - 1) **persistent** (user or sysadmin installs Parsl and it's always accessible by all nodes) and
  - 2) on the same **local network** - ports on head node are accessible to worker nodes

---

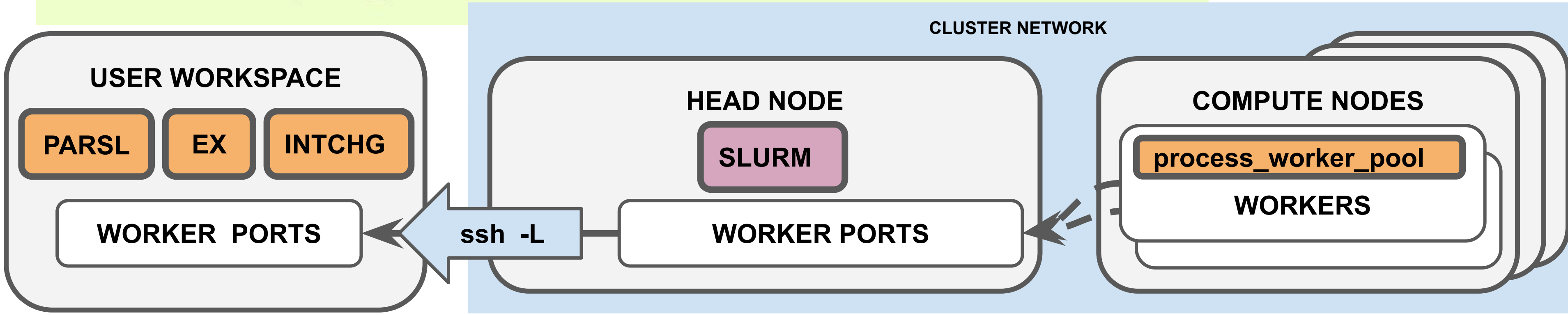
- **SSH tunnels and ports:** clusters aren't all on the same local network.
- **Parsl needs to be installed** everywhere with exactly the same versions - need automation for installation or attaching persistent storage when using *ephemeral* cloud resources.
- If using the FluxExecutor, Flux needs to be on the clusters; managing the installation of Flux *and* Parsl via Spack can be challenging. (But recently, Flux can be installed directly with Conda.)

Architecture for multi-site with HighThroughputExecutor



```
                   |  Data   |  Executor   |  Interchange  | External Process(es)
                   |  Flow   |             |               |
              Task | Kernel  |             |               |
            +----->|-------->|------------>|->outgoing_q---|-> process_worker_pool
            |      |         |             | batching      |    |         |
      Parsl<---Fut-|         |             | load-balancing|  result   exception
                ^  |         |             | watchdogs     |    |         |
                |  |         |   Q_mngmnt  |               |    V         V
                |  |         |    Thread<--|-incoming_q<---|--- +---------+
                |  |         |      |      |               |
                |  |         |      |      |               |
                +----update_fut-----+
```

Credit:
[Parsl docs]



How does it *actually* work?

1. PW connects to cluster head node with **ssh -R tunnel** and modifies ~/.ssh/config for easy connections back to PW (i.e. forwarding head node SSH port to PW).
2. PW **launches workflow with a bash script** in the user's workspace on PW platform.
3. The launch script clones and then starts parsl_utils which:
   a. Gathers **resource information** (i.e. IP address of cluster) via PW API
   b. Establishes **SSH -L tunnels** port forwarding the Parsl worker ports from the cluster to PW
   c. Checks/**installs Conda & Parsl**
   d. Builds the **Parsl configuration** based on a, b, c & template
4. **Parsl workflow is launched**

https://github.com/parallelworks/parsl_utils

```
Config(
    executors=[HighThroughputExecutor(
        address='*',
        label='host1',
        cores_per_worker=1.0,
        launch_cmd='process_worker_pool.py -a 10.128.0.17…'
        provider=SlurmProvider(
            channel=SSHChannel(
                '34.16.72.220',
                key_filename='/home/sfgary/.ssh/pw_id_rsa',
                port=22,
                script_dir='/home/sfgary/pw/jobs/…',
                username='sfgary'
            ),
            init_blocks=0,
            launcher=SingleNodeLauncher(debug=True, fail_on_any=False),
            max_blocks=1,
            nodes_per_block=1,
            parallelism=1,
            partition='compute',
            regex_job_id='Submitted batch job (?P<id>\\S*)',
            scheduler_options='\n#SBATCH --exclusive\n',
            walltime='01:00:00',
            worker_init="export PYTHONPATH=/home/sfgary/miniconda…"
        ),
        storage_access=[PWRSyncStaging(), PWGsutil(), PWS3()],
        worker_debug=True,
        worker_logdir_root='/home/sfgary/pw/jobs/…',
        worker_port_range=(50000, 55500),
        worker_ports=(53404, 54568),
        working_dir='/home/sfgary/pw/jobs/…'
    ), HighThroughputExecutor(...)]
```

Parsl multi-site **future work**

- **Parsl ≥2023.7.24** - need HTEX address="*" and specify the local IP address of the head node when starting process_worker_pool.py in HTEX launch_cmd.
- **A custom data provider** in parsl_utils wraps file, GCP, and AWS bucket access. We plan to automate the workflow integration of storage information in a similar way as we do with compute resources.

Timeout/failover **future work:** The timeout starts counting when the app starts running, not when it is submitted (and queued).

HTEX limitations for MPI jobs (and workaround), **future work**

Parsl HTEX hardcodes SLURM --ntasks-per-node to 1; prevents running multiple MPI tasks on the same node in parallel.

Chris Harrop's talk tomorrow

It is possible to bypass this issue by changing SLURM_TASKS_PER_NODE *in the bash_app itself* and forcing a static number of Parsl blocks (i.e. init_blocks = min_blocks = max_blocks = repeats or parallelism > 1). The FluxExecutor is an alternative for launching MPI tasks in parallel.

Thank you Ben and Yadu!