Hewlett Packard Enterprise

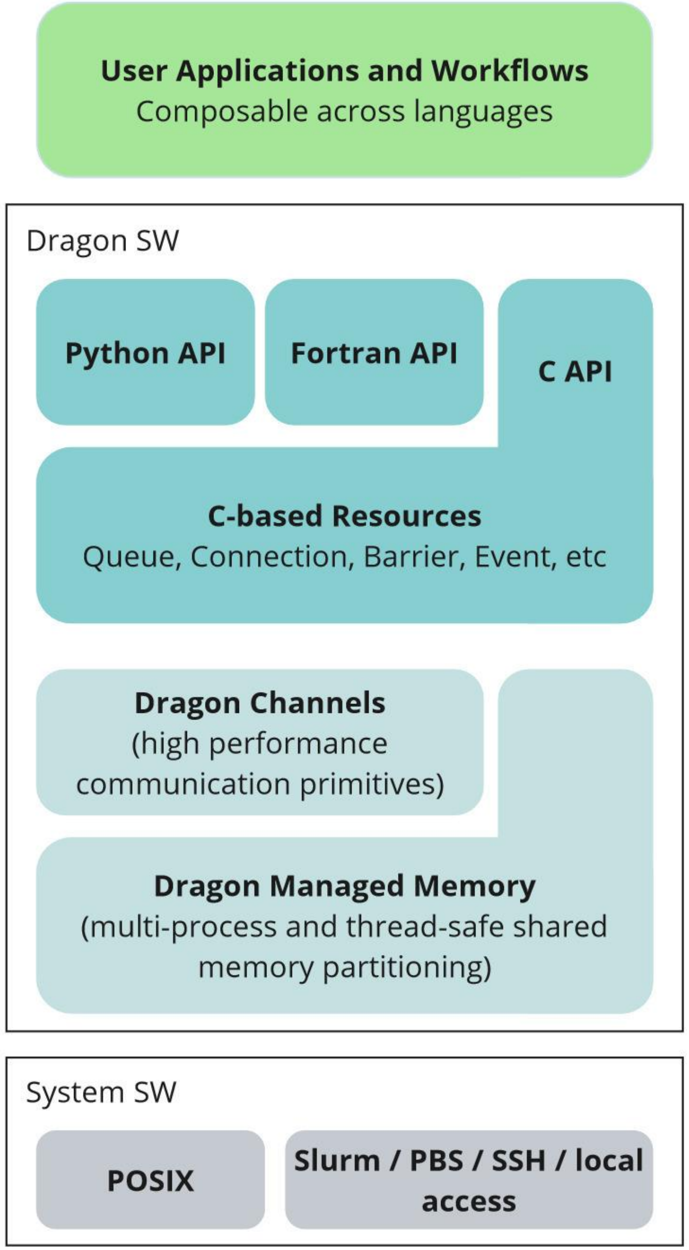
A High-Performance Parsl Executor Based on Dragon

Pete Mendygral, HPC&AI Cloud Services

October 20, 2023

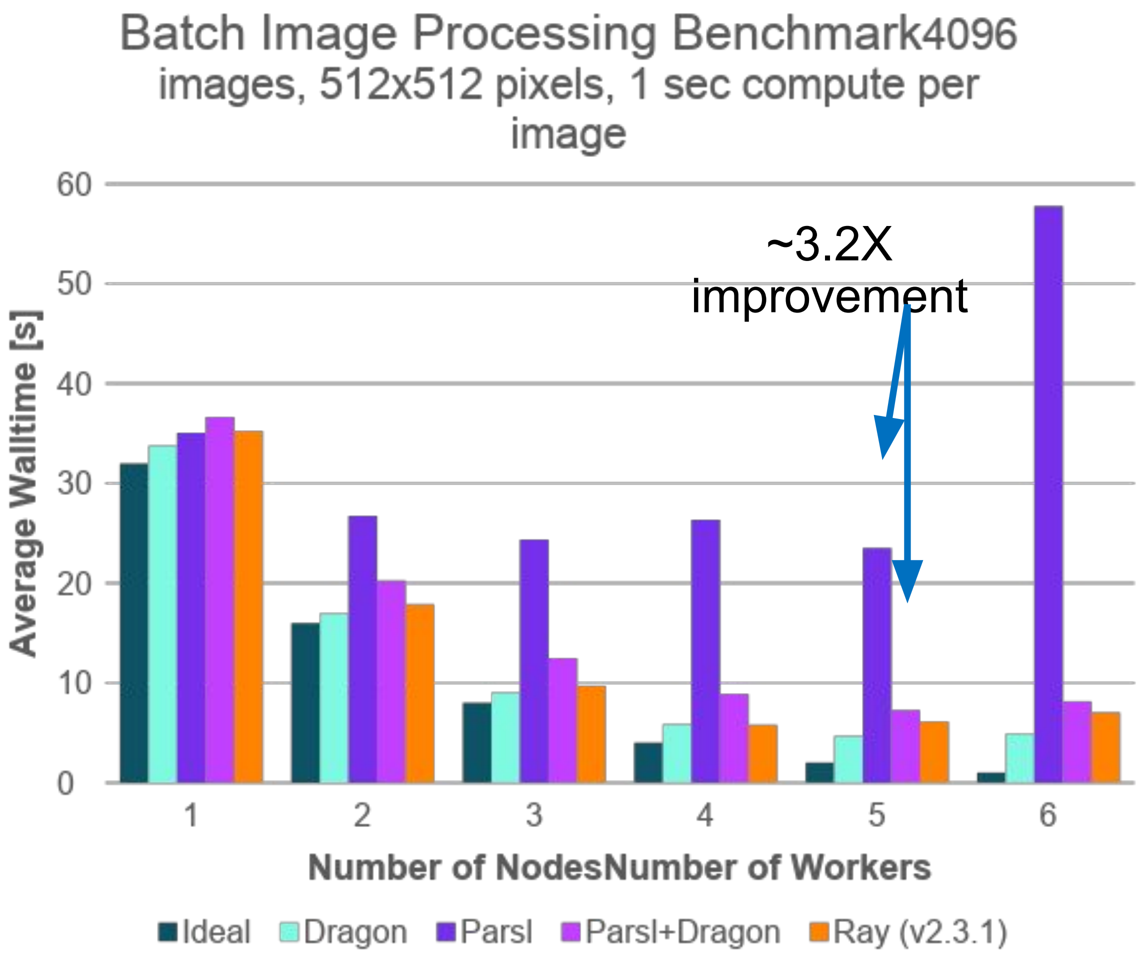
What did we do?

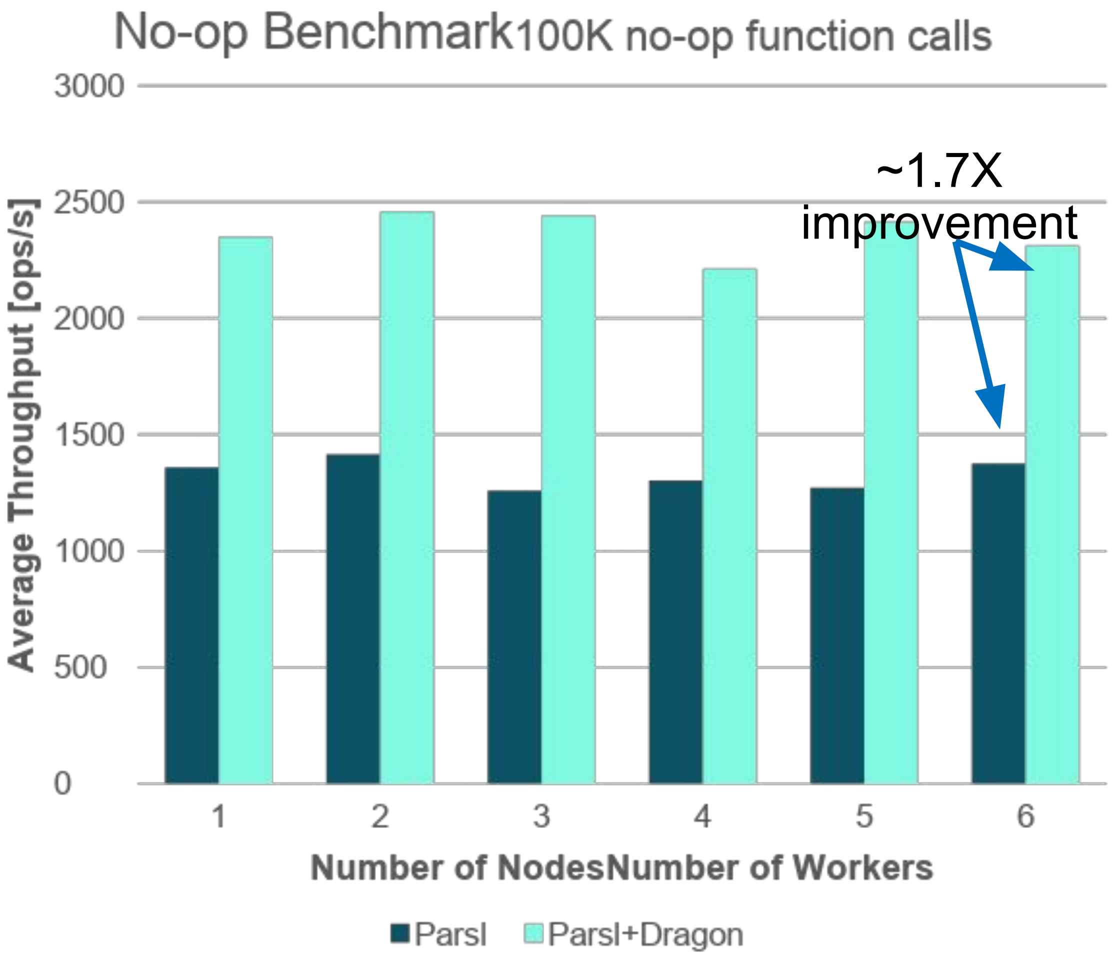
- Dragon
  - Composable distributed run-time for managing dynamic processes, memory, and data at scale through high-performance communication objects
  - Core interfaces for Python/C/C++/Fortran*
  - Higher level interfaces for targeted use-cases
    - Standard Python `multiprocessing` API
      - Transparently scales efficiently across many nodes
      - Validated against CPython unit tests
    - Other interfaces in roadmap, **like Parsl!**
  - Self-contained with minimal external dependencies
  - Open source: https://github.com/DragonHPC/dragon
    - Developed and maintained by HPE and community
- Dragon executor for Parsl
  - Implemented with `multiprocessing` and Dragon-native APIs
  - First target `@python_app`
  - https://github.com/DragonHPC/dragon/blob/main/src/dragon/workflows/parsl_batch_executor.py



*Core interfaces not yet in all languages listed

Parsl+Dragon Benchmarking





- Cray EX with AMD processors
- Benchmark run inside existing allocation

- Dragon data gathered with RDMA-enabled transport (HSTA)
- https://github.com/DragonHPC/dragon/blob/main/examples/multiprocessing/numpy-mpi4py-examples/parsl_batched_scipy_scale_work.py

Parsl @mpi_app with Dragon

- Manage MPI applications within an allocation
- Proof of concept with plans to adapt to official Parsl API

```python
# input args to function is arbitrary.
# Return tuple is most important
@mpi_app
def mpi_factorial_app(num_ranks: int, bias: float,
                      policy: Policy = None):
    import os

    # executable located in run_dir that we want to launch
    exe = "factorial"
    run_dir = os.getcwd()
    # list of the mpi args we want to pass to the app
    mpi_args = [str(bias)]
    # format that is expected by the DragonMPIExecutor
    return exe, run_dir, policy, num_ranks, mpi_args
```

```python
def main():
    mp.set_start_method("dragon")

    config = Config(
        executors=[
            DragonMPIExecutor(),
        ],
        strategy=None,
    )

    parsl.load(config)

    bias = 10
    num_mpi_ranks = 10
    scale_factor = 1 / 10000
    connections = mpi_factorial_app(num_mpi_ranks, bias)
    send_scale_factor(connections.result()["in"], scale_factor)
    output_string = get_results(connections.result()["out"])
    print(
        f"mpi computation: {output_string}, exact = {(
            scale_factor * math.factorial(num_mpi_ranks-1) + bias)} ",
        flush=True,
    )
```

```
> dragon parsl_mpi_app_demo.py
mpi computation: 0.000100 * 362880.000000 + 10.000000 = 46.288000, exact = 46.288000000000004
```

Dragon Info and Next Steps

- **Dragon Info:**
  - Github repo with latest build: https://github.com/DragonHPC
  - Documentation: https://dragonhpc.github.io/dragon
  - Email HPE dev team: dragonhpc@hpe.com

- **Next Steps for Improving Dragon Integration with Parsl:**
  - Prioritize additional Parsl API integration targets for Dragon
  - Explore opportunities for integrating Dragon based communication / sync objects
    – connection, queue, barrier, dictionary objects
  - Enable use of the Dragon Executor from outside an existing allocation
  - Opportunities for using Dragon Telemetry for realtime Parsl workflow insights
  - Opportunities for using Dragon Proxy for multi-site Parsl workflows

Please stop by the High Performance Python for Science at Scale (HPPSS) workshop at SC23!

https://hppss.github.io/SC23/

Thank you

Pete Mendygral (pete.mendygral@hpe.com)

HPE Dragon Team: Michael Burke, Eric Cozzi, Julius Donnert, Veena Ghorakavi, Faisal Hadi, Nick Hill, Maria Kalantzi, Kent Lee, Pete Mendygral, Davin Potts, Nick Radcliffe, Rajesh Ratnakaram