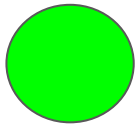
TaskVine/Parsl

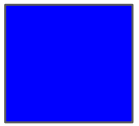
Interactions between DAG Manager and Workflow Executor

Colin Thomas
University of Notre Dame

TaskVine Temporary Files

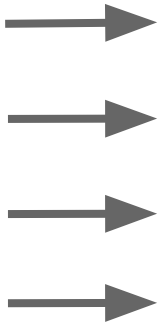


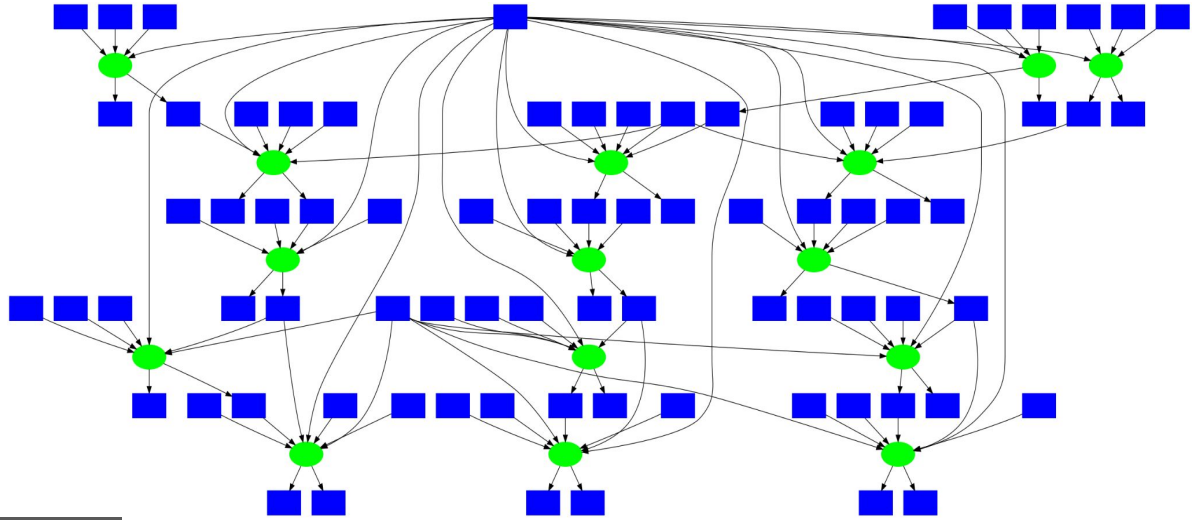
Task



Data

Intermediate Data



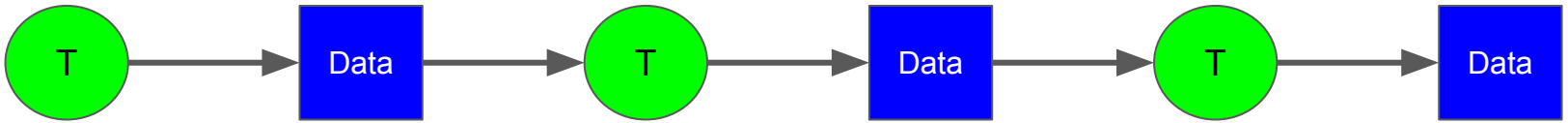


Workflow DAG

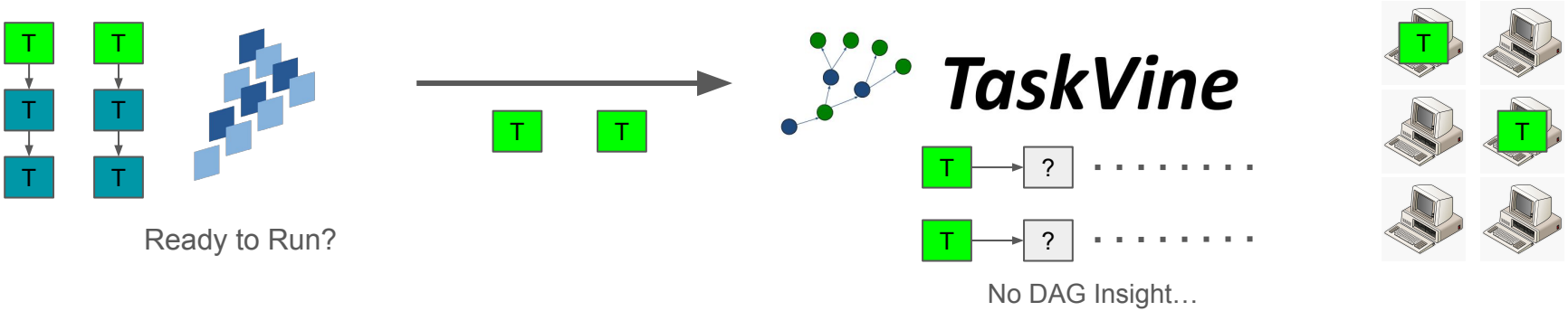
```
from parsl.data_provider.files import File

Tmp_data = File("taskvinetemp://data")
```

Optimal Scheduling of Sequential Tasks

- Ideally we would schedule sequentially dependent tasks to run on a single node
- Scheduling on a per-task basis, it is costly to match tasks to worker with the correct local data
- We must look ahead in the application and make associations of sequential tasks.



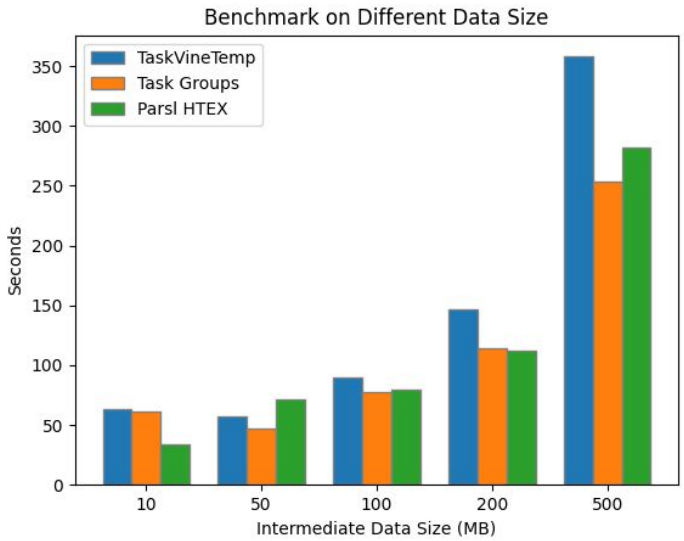
Implementation - Necessary Interactions



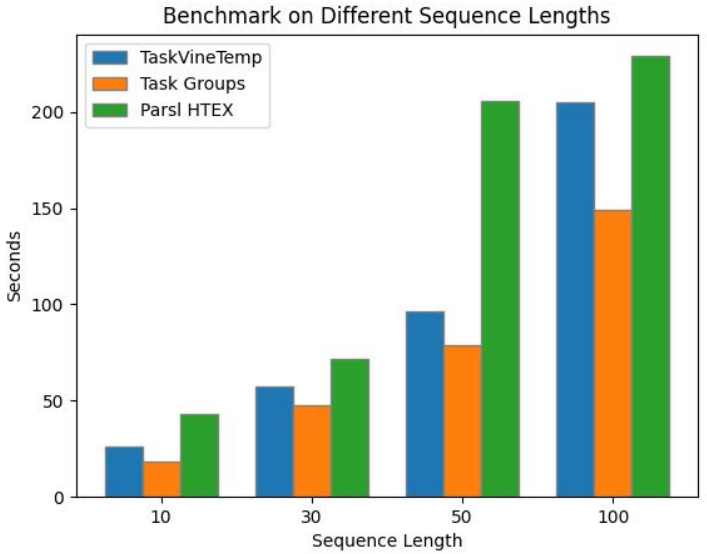
Benchmark Evaluation

- Each run consists of 20 starting tasks, each followed with a sequence of dependent tasks.
- Each intermediate task produces and consumes a file

20 Sequences of 30 Tasks



20 Sequences, 50MB Intermediate Data

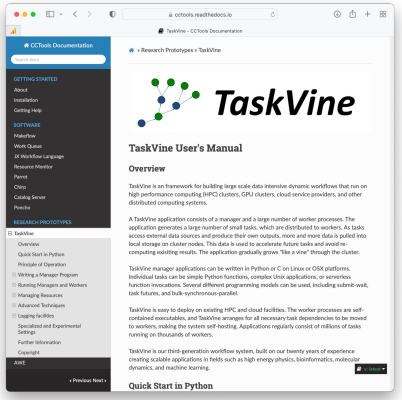


Further Thoughts

- Grouping tasks is best suited toward applications with sequential tasks with non-trivial intermediate dependencies.
- Committing long sequences of tasks to a single worker in a diverse cluster may be disadvantageous if the worker is less performant than average.
- How should we take into account a complex DAG where there may be multiple interpretations of group assignment, varied data sizes and consequences for fault recovery?

Colin Thomas
Ph.D. Student

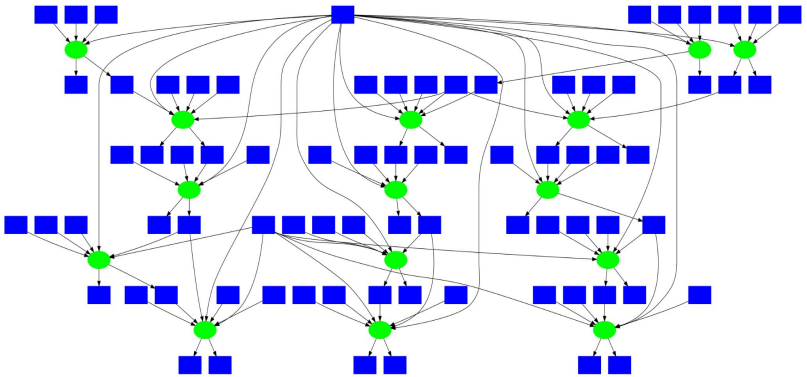
cthoma26@nd.edu

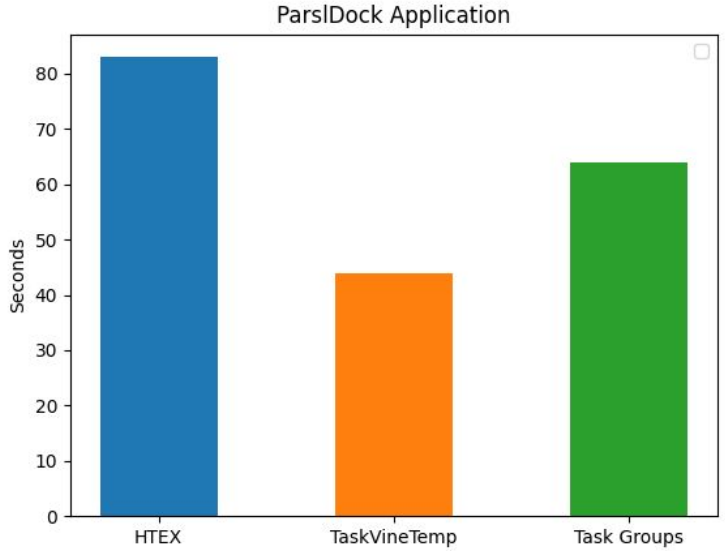


Supplemental Slides

ParslDock

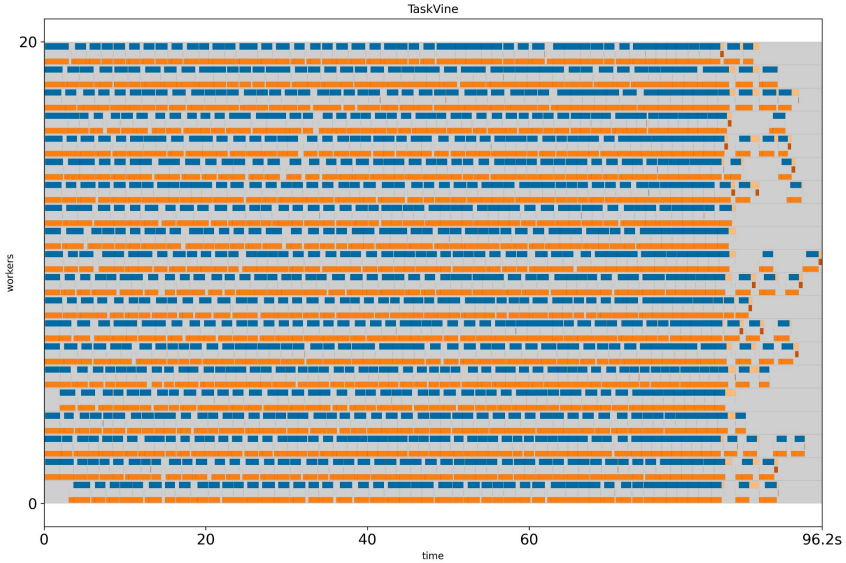
- ParslDock creates several batches of sequentially dependent tasks.
- A sequence is 5 tasks with multiple intermediate dependencies.

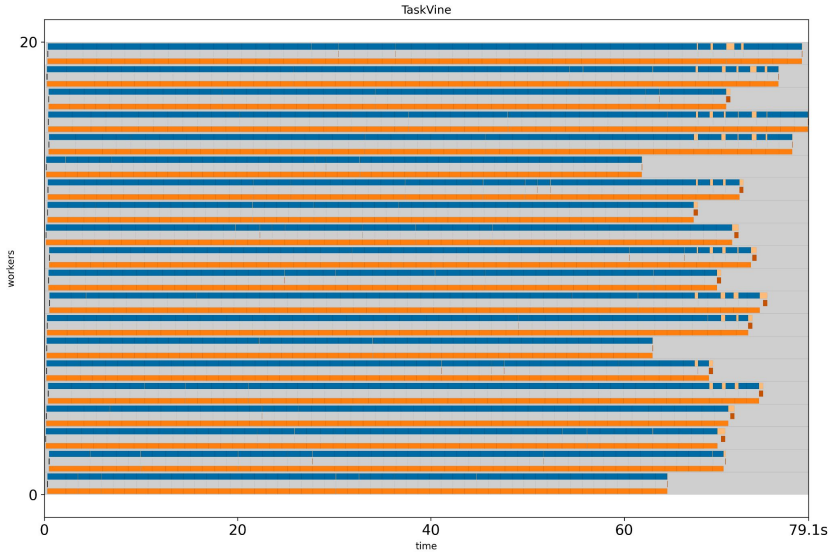




Task Groups

- Scheduling sequential tasks as one “group” eliminates latency between one task’s completion and the next beginning.
- Data transfer between shared FS and nodes is reduced.
- Failures do not require restarting the whole sequence. Granularity of tasks is not changed.





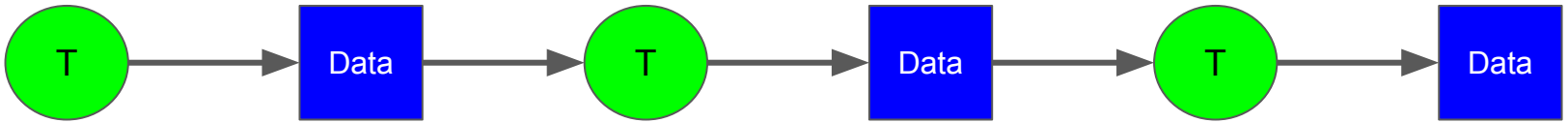
Expressing Temporary Files in Parsl

- Temporary files are declared using the *taskvinetemp://* protocol.
- They do not exist upon declaration, but are created by a task.


```python
import parsl
from parsl import bash_app
from parsl.config import Config
from parsl.executors.taskvine import TaskVineExecutor
from parsl.executors.taskvine import TaskVineFactoryConfig
from parsl.executors.taskvine import TaskVineManagerConfig
from parsl.executors.taskvine.taskvine_staging_provider import TaskVineStaging
from parsl.data_provider.files import File

config = Config(
        executors=[
            TaskVineExecutor(
                manager_config=TaskVineManagerConfig(project_name="temp_file_example"),
                factory_config=TaskVineFactoryConfig(batch_type="local"),
                storage_access=[TaskVineStaging()],
            ),]
        )

# intermediate producer
@bash_app
def f1(outputs = []):
    return "echo intermediate_data > intermediate.txt"

# intermediate consumer
@bash_app
def f2(inputs = [], stdout="output.txt"):
    return f"cat {inputs[0]}"

with parsl.load(config) as conf:
        intermediate_file = File("taskvinetemp://intermediate.txt")
        intermediate_producer = f1(outputs = [intermediate_file])
        intermediate_consumer = f2(inputs = [intermediate_producer.outputs[0]])
        intermediate_consumer.result()
```

Optimal Scheduling of Sequential Tasks

- Ideally we would run each sequence of tasks on a single node, eliminating unnecessary data movement.
- Considering an individual task out of a queue and attempting to find the host where data dependencies are met has a large overhead.
- How does a versatile, data-centric scheduler take advantage of massive locality benefits in large-scale workflows without complex searches?
- Automatically identify sequential data patterns in declared tasks and label them into groups.



Implementation - Necessary Interactions

- Parsl only gives the executor tasks that are ready to run.
- The executor has useful information about the state of workers and data locality, but it cannot be fully exploited without a bigger picture of the DAG.
- In order for TaskVine to be informed about tasks which are related, but not ready to run, we must “fool” the Parsl data staging into believing that intermediate data has already been created.