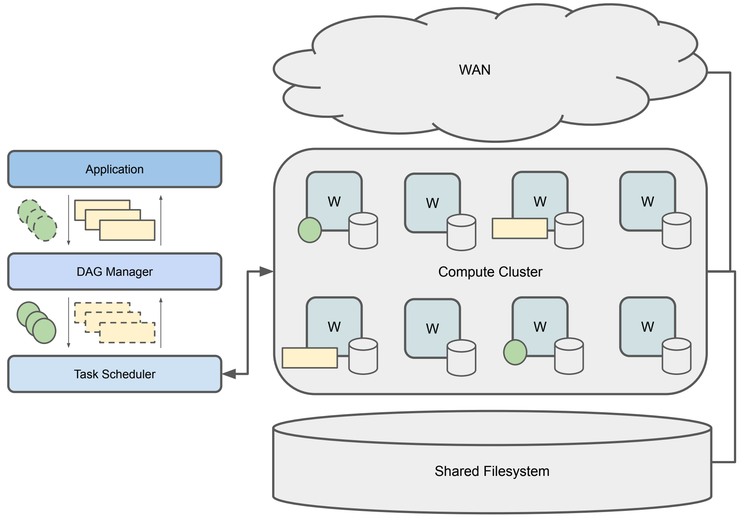
Task Graph Restructuring via Function Based Annotation For Large-Scale Scientific Applications

Barry Sly-Delgado(bslydelg@nd.edu) University of Notre Dame - http://ccl.cse.nd.edu/

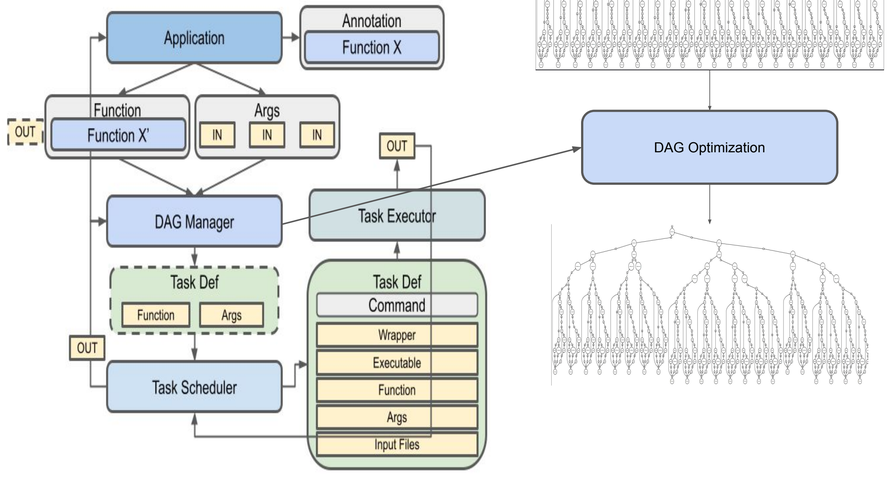
Abstract

Distributed Frameworks allow for users to create large-scale distributed scientific applications within languages such as Python. Within these frameworks users provide functions that represent individual tasks in a broader DAG. Users compose these graphs using the provided API of a given framework. However, the initial composition of the DAG may inhibit the overall performance of the application. That is, the resulting data flow from the initial topology of a graph can become a bottleneck via network and disk utilization. A graph can be reshaped, producing an identical result with a more ideal dataflow that limits network and disk strain. This work provides the methodology used for function annotation, which provides insight to reshape applications and better manage data. The methodology is incorporated into the composition of the frameworks Dask and TaskVine. Additionally, we show improvement with real-world applications in the domain of high energy physics

Workflow Stack



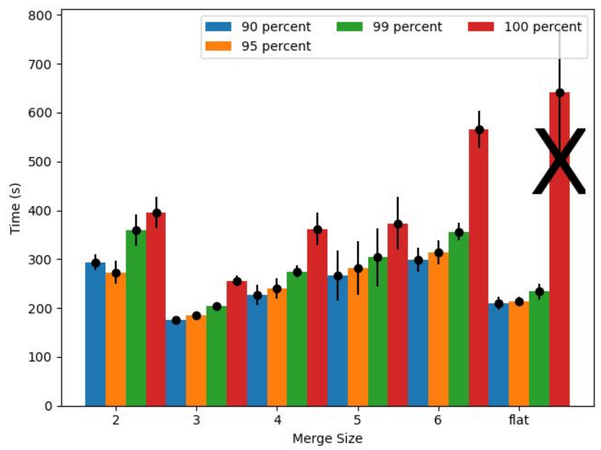
Graph Restructuring

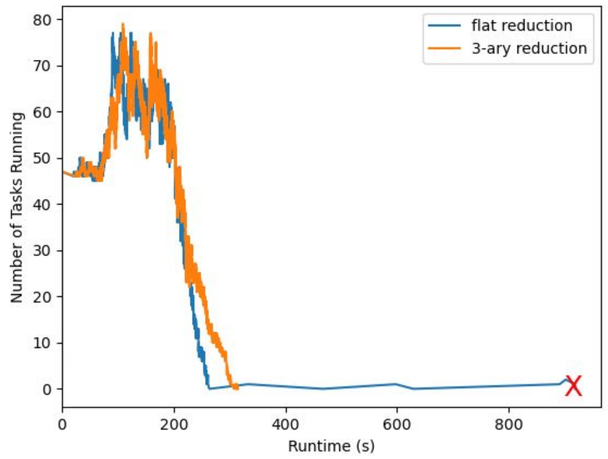


Code

```python
def write(events):
    import awkward as ak
    d = ak.to_parquet(events, "out.parquet")
    return d

@daskvine_merge
def merge(*events):
    import awkward as ak
    return ak.concatenate(events)

graph = {}
events = get_events()
proxies  = []

# Tree Reduction
for i in range(events.npartitions):
    d = events.partitions[i:i+1]
    proxies.append(d)
while(len(delayed) > 1):
    # removes merge_size proxies
    to_merge = get_proxies(merge_size, proxies)
    d = dask.delayed(merge)(*to_merge)
    proxies.append(d)
d = dask.delayed(write)(proxies[0])
graph[f"{dataset_name}_write"] = d

# Single Node Reduction
d = dask.delayed(write)(d)
graph[f"{dataset_name}_write"] = d

# Auto Merge
d = dask.delayed(merge)(events)
d = dask.delayed(write)(d)
graph[f"{dataset_name}_write"] = d
```

Results





(Left)) - Competition times of X% of tasks using a N-ary reductions. Generally, as the size of N increases the the time to complete X% of tasks increases as well. Note the increasing disparity between 99% and 100% completion

(Right) - The amount of concurrent tasks executing during a run of RsTriPhoton using a flat reduction and a 3-ary reduction. Larger reduction sizes produce a large tail during execution.

CCTools

NSF

UNIVERSITY OF NOTRE DAME

TaskVine