Minimizing Data Movement Using Distant Futures

Barry Sly-Delgado, Douglas Thain - University of Notre Dame

TaskVine CCTools NSF

Background

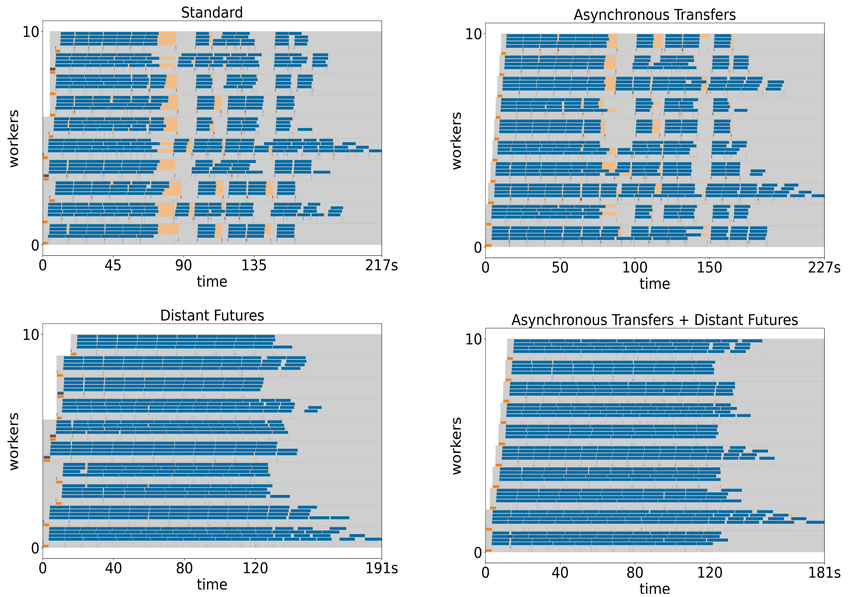
Scientific workflows execute a series of tasks where each task may consume data as an input and produce data as an output. Within these workflows, tasks often produce intermediate results that may serve as inputs to subsequent tasks within the workflow. These results can vary in size and may need to be transported to another worker node. Data movement can become the primary bottleneck for many scientific workflows thus minimizing the cost of data movement can provide a significant performance benefit for a given workflow. Distant futures enable transfers between worker nodes, eliminating the need for intermediate results to pass through a centralized manager for future tasks invocations. Additionally, asynchronous transfers enable increased concurrency by preventing the blocking of task invocations. This poster shows the performance benefit received from the implementation of distant futures within a workflow that produces numerous intermediate results.

Workflow Improvements

Distant Futures - A future is a reference to a result which will be resolved at a later point in time. Distant futures expand this idea to be utilized on a series of worker nodes spread throughout a compute cluster. Once a result is resolved workers can transfer results between themselves removing a hop through the manager for results.

Asynchronous Transfers - data transfers are characterized by operations in which a worker is transferring data via a remote source, between another worker, or staging data. Performing these operations asynchronously allows for tasks to execute faster.

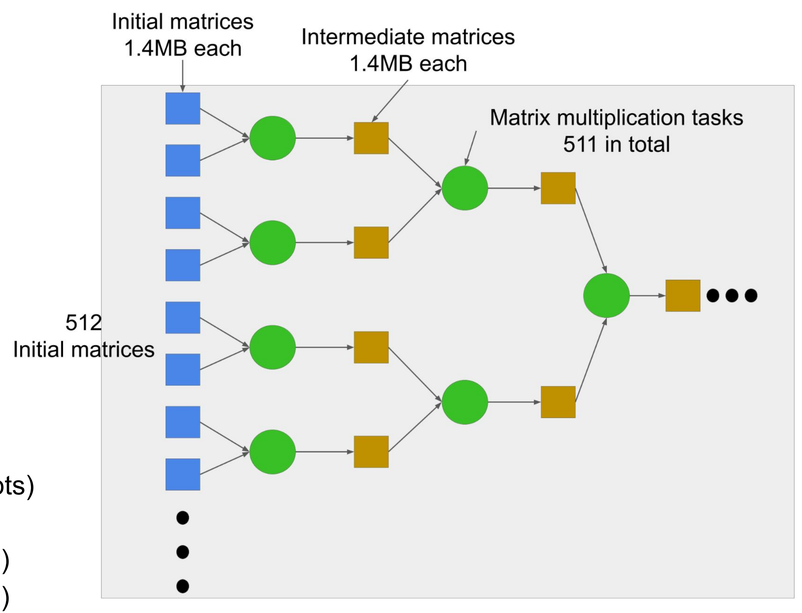
Performance Results



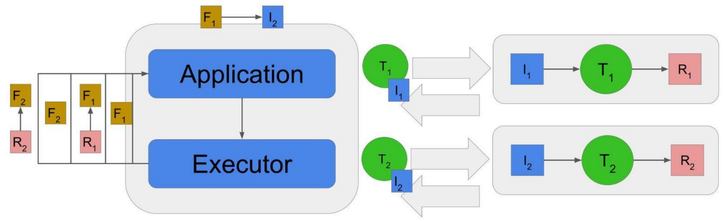
Code Example

```
Import ndcctools.taskvine as vine
def gen_matrix(n):
    ...
def empty_matrix(n):
    ...
def matrix_multiply(a,b):
    result = empty_matrix(len(a))
    for i in range(len(a)):
        for j in range(len(b[0])):
            for k in range(len(b)):
                result[i][j] += a[i][k] * b[k][j]
    return result
# create executor with these options
opts {'memory':8000,'disk':8000}
e=vine.Executor(name="add_app', batch_type='sge',opts=opts)
# submit function invocations
a = e.submit(matrix_multiply, gen_matrix(20), gen_matrix(20))
b = e.submit(matrix_multiply, gen_matrix(20), gen_matrix(20))
c = e.submit(matrix_multiply, a, b)
print(c.result())
```

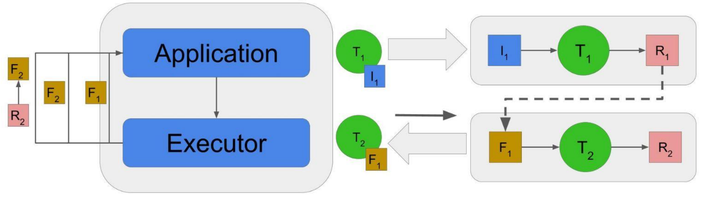
Workflow



print(e.future(T2, e.future(T1, I)))



Normal Futures



Distant Futures