HETEROGENEOUS HIERARCHICAL WORKFLOW COMPOSITION

Orçun YILDIZ (ANL), Jorge Ejarque (BSC), Henry Chan (ANL), Subramanian Sankaranarayanan (ANL), Rosa M. Badia (BSC), Tom Peterka (ANL)

9th JLESC Workshop
Knoxville, April 2019
MOTIVATION

SCIENCE APPLICATIONS WITH DIVERSE REQUIREMENTS

WORKFLOWS TO THE RESCUE

- In situ workflows:
  - Address the mismatch between I/O and computing
  - Run within a single HPC system
  - Data exchange through memory or HPC interconnect

- Distributed workflows:
  - High-throughput, task-parallel workflows
  - Can run across several independent systems
  - Data exchange through files

- Problem: These workflow systems lack interoperability with each other!

“A supercomputer is a device for turning compute-bound problems into I/O-bound problems” – Ken Batcher
OUR APPROACH

HETEROGENEOUS HIERARCHICAL WORKFLOW COMPOSITION

End-to-End Workflow

Pre-processing workflow

In-situ computation

Post-processing workflow

Distributed Computing Engine (PyCOMPSs)

- Data dependency detection
- Data Management
- Task Scheduling

In-situ Computing Engine (Decaf)

- Memory-to-Memory Transfers
- In-situ execution

Execution Management

Results Collection & Data Synchronization
METHODOLOGY

Employing Decaf in situ subworkflows as tasks of PyCOMPSs

Challenges:
• Describing in situ computations in a distributed workflow without adding extra complexity
• Efficient execution of the in situ computations without interfering with the other tasks of the distributed workflow.

PyCOMPSs-Decaf integration:
• Simple interface:
  • Extended PyCOMPSs syntax with Decaf decorator
  • No Decaf implementation details required

• Efficient execution by PyCOMPSs:
  • Treats the entire Decaf subworkflow as a single task
  • Generates the task graph by analyzing its dependencies
  • Schedules the subworkflow according to its constraints (e.g., MPI processes)
SAMPLE PyCOMPSs PROGRAM

Decaf and normal task description and invocation in PyCOMPSs

```python
@constraint(ComputingUnits=num_cores)
@decaf(dfScript="decaf-config-script.py")
@task(infile=FILE_IN, outfile=FILE_OUT, returns=int)
def decaf_task_method(infile, outfile):
    pass

@constraint(ComputingUnits=num_cores)
@task(outfile=FILE_OUT)  # in_param input object as default
def normal_task(in_param, outfile):
    ...

if __name__ == "__main__":
    normal_task(param, file1)
    decaf_task_method(file1, file2)
```

Decaf task

Normal PyCOMPSs task
SCIENCE USE CASE
NUCLEATION IN FREEZING WATER

- Stochastic rare event: difficult to capture in simulations
- Two ways to simulate:
  - A single large simulation -> massive computing power
  - Many instances of small simulations -> difficult to observe in small systems
DIAMOND STRUCTURE DETECTOR

- Nucleation process: liquid (water) -> ice crystals
- Identifies the different phases of ice (e.g., liquid, hexagonal, cubic)
- Scientists perform this detection manually by using OVITO* visualization software:
  - Developed a parallel version and coupled it in situ with simulation

EXPERIMENTAL SETUP

- MareNostrum 4 supercomputer:
  - Two 24-core Intel Xeon Platinum 2.4 GHz CPUs
  - 2 GB or 8 GB DDR4 RAM memory per core
  - 10 Gb Ethernet network

- Studied nucleation with two different simulation sizes:

<table>
<thead>
<tr>
<th>Simulation Size</th>
<th># of concurrent Decaf instances</th>
<th># of processes per Decaf instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large (2M)</td>
<td>1</td>
<td>2,016</td>
</tr>
<tr>
<td>Small (4K)</td>
<td>15</td>
<td>48</td>
</tr>
</tbody>
</table>
CAPTURING NUCLEATION IN A SINGLE LARGE SIMULATION

<table>
<thead>
<tr>
<th>Approach</th>
<th>Simulation Time</th>
<th>Post-processing Time</th>
<th>End-to-End Execution Time</th>
<th>Storage Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous workflow</td>
<td>11 hours</td>
<td>-</td>
<td>11 hours</td>
<td>250 MB</td>
</tr>
<tr>
<td>Traditional (post hoc)</td>
<td>11 hours</td>
<td>17+ hours</td>
<td>28+ hours</td>
<td>16 GB</td>
</tr>
</tbody>
</table>

Our hierarchical workflow model brings significant time and storage savings by using these different workflows for what they do best.
By launching dynamic ensemble of simulations, we can capture nucleation by using 94% fewer computing resources.
SUMMARY

• Integrating PyCOMPSs and Decaf allows:
  • Accelerating the performance of distributed workflows by providing access to HPC dataflows
  • Extending in situ workflows with dynamicity and different programming models (e.g., looping and bag-of-tasks)
  • Easier implementation of multilevel workflows for science applications

“Stochastic rare event phenomena such as nucleation often require large length-scale/long timescale simulations and represent popular examples of materials science applications that will benefit from this workflow” – S. Sankaranarayanan

- Decaf is open source and available at https://github/tpeterka/decaf
- PyCOMPSs is open source and available at http://compss.bsc.es
CURRENT WORK: HEP WORKFLOW

In situ neutrino event generation and parameter optimization

Courtesy Tom Peterka
THANK YOU!

Orçun YILDIZ
oyildiz@anl.gov