A Data-Centric Optimization Workflow for the Python Language

Alexandros Nikolaos Ziogas, Torsten Hoefler

Motivation
Python is the language of choice for scientific computing due to its high productivity. However, its execution is often slow. How do we make Python the language of choice for writing highly performant code too?

From Python...
Python with type annotations for Ahead-Of-Time compilation

```python
@dace.program
def jacobi2d(tsteps: dace.int32, arr: dace.float64[2, N, N]):
    for t in range(tsteps):
        req = np.empty((8,), dtype=dace.comm.Request)
        # ...
```

Implementation in Python and optimization of Quantum Transport Simulation

```python
arr[(t+1)%2, 1:-1, 1:-1] = 0.2 * (arr[t%2, 1:-1, 1:-1] +
                        arr[t%2, 2:, 1:-1] +
                        arr[t%2, :-2, 1:-1] +
                        arr[t%2, 1:-1, :-2])
```

... to a Data-Centric Representation ...

Stateful DataFlow multiGraphs
States encapsulate dataflow
- Interstate edges annotated with control-flow
- Array slicing
- Dataflow edges annotated with the subset of data being accessed
- Parametrically parallel scope
- Data access
- Intermediate data

MPI-compatible syntax for distributed-memory programming

```python
@dace.program
def jacobi2d_distr(tsteps: dace.int32, arr: dace.float64[2, Nx+2, Ny+2]):
    @dace.program
    for t in range(tsteps):
        req = np.empty((8,), dtype=dace.comm.Request)
        # ...
```

... optimized via Graph Transformations ...

Optimization driven by:
- User/performance engineer
- Automatic optimization heuristics
- Performance modeling

... generating Architecture-Specific Code

FPGA

Multilinear Algebra and Einsums
Using Data-Centric Python to implement Deinsum, an automated framework for distributed multilinear algebra computations expressed in Einstein notation.

For references, resources, and other information, please follow the link below: