CASPER: Compiler Abstractions Supporting high Performance on Extreme-scale Resources

Connor Imes, Alexei Colin, Naifeng Zhang, Ajitesh Srivastava, Viktor Prasanna, John Paul Walters

ESPM2 @ Supercomputing – November 11, 2020
Goals and Challenges for HPC Application Developers

**Challenges**
- Scale
- Diversity
- Heterogeneity
- Dynamics

**Goals**
- Efficiency
- Portability
- Productivity

Domain Specific Languages + Extensions
Application Domains: SAR and CFD

Raw echo data

Patch extraction

Range compression

Range cell migration correction

Azimuth compression

Processed Patch Concatenation

Multilook processing

Final SAR image

Construct Mesh

Write CFD kernel

Construct simulation initial conditions

Parallelize over each cell in mesh

Perform fluid physics computation

Update local matrix, vector

Reduce over global matrix, vector

Solve Linear System

For each iteration
CASPER Compiler Architecture

**SAR Program**
- Halide Expression Kernel 0
  - `out0(x) = in(x-1) + in(x)`
- Halide Expression Kernel N
  - `out1(x) = out0(x/2) + ...`

**CASPER Metaprogram C++ API**
- `casper::TaskGraph tg;` 
- `d0 = tg.createDat();` 
- `T0 = tg.createTask(CKernel(., {d0});` 
- `T1 = tg.createTask(HalideKernel("out0"), {d0}, {T0});` 
- `casper::compile(tg);`

**CFD Program**
- UFL Expression Kernel 0
  - `L = dt*dot(grad(q), u)*t)*dx`
- UFL Expression Kernel N
  - `a = q*t+t*dt...

**CASPER Metaprogram C++ API**
- `casper::TaskGraph tg;` 
- `sol = tg.createPyObj();` 
- `T0 = tg.createTask(CFDKernel("flow"), {sol});` 
- `casper::compile(tg);`

**PyOP2 Compiler**
- `def flow(): return solve(L==a)`

**CASPER Scheduler**
- `IN set` 
- `OUT set` 

**Halide Compiler**
- Task Graph (ATIG) IR
  - Task 0
  - Task 1
  - Task N-1
  - Task N

**ATIG Learning Annotations**
- Neural Networks
  - `L = dt*dot(grad(q), u)*t)*dx`
  - `UFL Expression Kernel 0`
  - `a = q*t+t*dt...

**CASPER Backend**
- Native x86 obj. code
  - addps xmm0, [v2]
  - PTX source or SASS GPU
  - `add.f32 r2, r2, 0.5`

**Memory Allocations**
- `CUDA RT`
  - `MPI RT`
  - `CUDA RT`

**CASPER Runtime**
- Groove
  - ATIG Mapping on HW Platform
  - ATIG Learning Annotations
  - CASPER ATIG Optimizer

**CASPER Scheduler**
- CASPER Mapper
  - CASPER Backend
  - CASPER Frontend

**CASPER Metaprogram C++ API**
- `casper::TaskGraph tg;` 
- `tg.createDat();` 
- `tg.createTask(CKernel(., {d0});` 
- `tg.createTask(HalideKernel("out0"), {d0}, {T0});` 
- `casper::compile(tg);`
ATIG Optimization: Variant Selection

• Implementations of tasks expose some “knobs” that can be tuned affecting performance without affecting the program output.

• **Variant Selection Problem:** For a given <task, hardware> pair, identify the set of parameters, i.e., settings of the knobs, that result in the optimal performance (execution time).
  – Need to be able to evaluate performance given the program inputs and knobs.
    • Halide: tile size, vectorization width.
    • MPI: number of processes, max threads per process.

• **Performance prediction for each variant:** Our compact neural networks (< 100 weights) can accurately predict the execution time for various program inputs and knobs for given <task, hardware> pair.
  – Small size ensures fast training/retraining and fast inference at runtime.
Runtime Optimization

Manage application dataflow and resource utilization

Within a given scope:
• Dataflow Remapper
  – Optimizes dataflow for current scope using Deep Reinforcement Learning
• Decision Engines \([n \geq 0]\)
  – Configured for a dataflow
  – Tune HW/SW knobs to meet goals
Schedule Optimization

- Auto-scheduler: 3.8
- Auto schedule shape: 6.6
- Auto schedule min: 2.63
- Auto schedule pred: 2.89
- Manual schedule shape: 2.44
- Manual schedule min: 1.8
- Manual schedule pred: 1.85

Schedule:

- auto-scheduler
- auto_cs_average
- auto_cs_min
- auto_cs_pred
- man_cs_average
- man_cs_min
- man_cs_pred
SAR: FFTs on CPU vs GPU

Total runtime – FFTW (CPU) vs cuFFT (GPU)

Kernel runtime – FFTW (CPU) vs cuFFT (GPU)
CFD: Process and Thread Counts

MUMPS

SuperLU_Dist

Processes (MPI ranks)

OpenMP threads

Processes (MPI ranks)
Conclusions and Summary

• CASPER is a domain-specific compiler and runtime framework to enable domain scientists to productively and portably write efficient and scalable HPC applications.

• CASPER uses Annotated Task Interaction Graphs (ATIGs) to efficiently map kernels to diverse and heterogeneous resources.

• The CASPER runtime supports adaptation in dynamic operating environments.

• We have demonstrated the need for CASPER with:
  – The benefits of using ATIGs to optimize resource mappings
  – Challenges in determining a priori the resources and knob settings for common SAR and CFD operations on different hardware and inputs