HYDRA: Extending Shared Address Programming for Accelerator Clusters

Putt Sakdhnagool, Amit Sabne, Rudolf Eigenmann
School of Electrical and Computer Engineering, Purdue University

Highlights

- HYDRA: extending shared address programming to accelerator clusters
- Programmer only specifies parallel regions
- Fully automatic translation system generates MPI + accelerator code
- Optimized accelerator data placement and transfer
- High-level IR supports multiple accelerator architectures
- Average 64-node cluster speedups: 24.54x on Xeon Phi, 27.56x on GPU

HYDRA Programming Model

HYDRA is a directive-based shared address programming model offering a single parallel loop construct.

```
#pragma hydra parallel for [clauses]
```

```
#pragma hydra parallel for
for (i=1; i<SIZE_H+1; i++) {
    for (j=1; j<SIZE+1; j++) {
        a[i][j] = (b[i-1][j] + ...;
  }
}
```

Figure 1: HYDRA Program Example

All available clauses for the HYDRA parallel loop directive are listed in Table 1. These clauses are syntactically optional but might be needed for program semantics.

Clauses	Format
shared	shared(varlist)
private	private(varlist)
firstprivate	firstprivate(varlist)
reduction	reduction(op:varlist)

Table 1: HYDRA Parallel Loop Clauses

Extending Shared Address Programming Beyond CPU Clusters

Data Transfer Analysis

- Precise data transfer between host and accelerator memory is critical. Excessive transfer overhead can limit scalability

Algorithm

- Perform local analysis of shared data access
- First step: Identify necessary shared data for a program block
  - Use LUSE information to determine a live-in and live-out data
- Second step: Determine the transfer range of the shared data
  - Transfer range can be defined by the minimum lower bound and the maximum upper bound of local accesses

Memory Allocation Optimization

- Full data allocation could exceed its capacity
  - Failure of single accelerator execution
  - Inability to scale to multiple nodes

Algorithm

- Perform global analysis to summarize all accesses of the shared data
- The allocation size can be found using the minimum lower bound and maximum upper bound of all accesses
- Compiler deal with the misalignment of the newly allocated and old shard data

HYDRA Translation System

- Compiler: Translate HYDRA programs to accelerated MPI programs.
- Support multiple target accelerator architectures

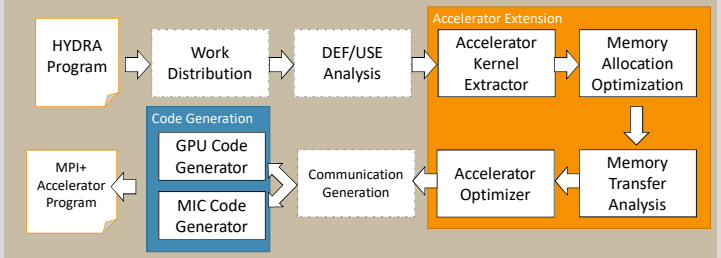
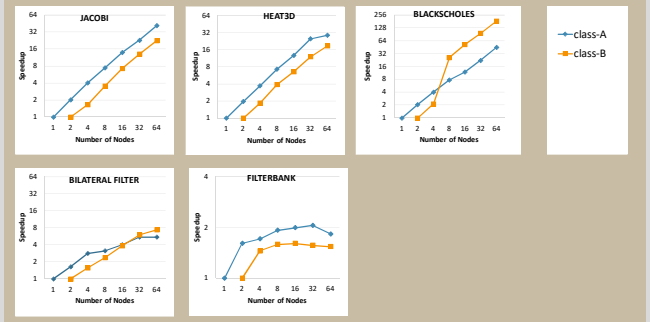


Figure 2: HYDRA Compiler Translation Process

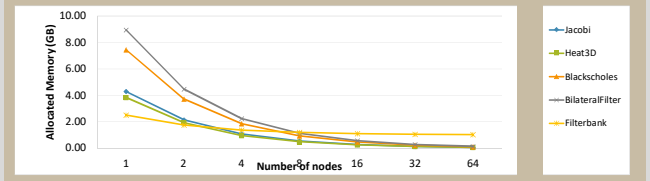
- Runtime: Handle remote accelerator communication

Evaluation

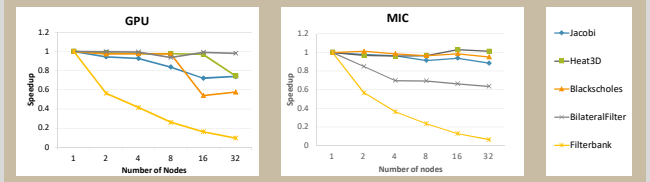
Strong Scaling - Scalability



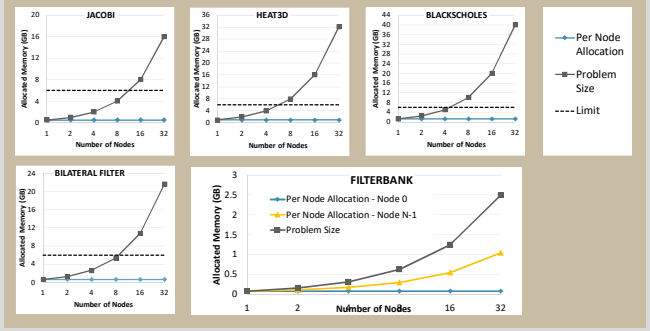
Strong Scaling – Memory Allocation



Weak Scaling - Scalability



Weak Scaling – Memory Allocation



Acknowledgments
This work was supported, in part, by the National Science Foundation under grants No. 0916817-CCF and 1449258-ACI. This research used resources of the Keeneland Computing Facility at the Georgia Institute of Technology and the Extreme Sciene and Engineering Discovery Environment (XSEDE), which are supported by the National Science Foundation under awards OCI-0910735 and ACI-1053575, respectively.