Deepak Majeti (deepak@rice.edu), Kamal Sharma (kgs1@rice.edu), Vivek Sarkar (vsarkar@rice.edu)

Extended TALC: User-Specified and Automatic Data Layout Selection for Portable Performance

Heterogeneous Habanero-C (H2C): A Portable Programming Model for Heterogeneous Processors

Data Layout Motivation

```
for (kk = kmn; kk < kmx; kk++) {
    for (jj = jmn; jj < jmx; jj++) {
        b[i] = a[i] + jj; 
        s[i] = d[i] + kk;  
    }
}
```

IRSmk Source Code

Relative performance of various data layouts on different hardware

- (27 x 1): 27 arrays are stored separately
- (9 x 3): interleaving groups of three arrays
- (3 x 9): interleaving groups of nine arrays
- (1 x 27): interleaving 27 arrays

Many DSLs and embedded DSLs have restricted aliasing semantics that allows for extensive data layout transformations.

H2C constructs

```
finish { Body }
```

- Execute Body, but wait until all tasks in Body have completed before proceeding to next statement.

```
async copyin(args1) copyout(args2) at(device)
```

- Asynchronously copy data between the device and the host.

```
forasync point(args) range(args) seq(args) 
[scratchpad(args)] at(device) { Body }
```

- Asynchronously execute iterations of Body on the specified device, using seq and scratchpad clauses to tune performance.

Overall Compilation Framework

H2C compiler implemented using the ROSE Compiler Framework

Highlights of H2C

1. Shared Virtual Memory: Supports recursive pointer data-structures on a GPU.
2. Data Layout Framework: Extends TALC to heterogeneous architectures.

Empirical Evaluation

Portability

- H2C code can be executed on various machines.
- All major vendors support OpenCL today.

Productivity

- Easy to write new applications in H2C.
- Minimal (~2%) code rewrite required to port existing Lattice Boltzmann Method (LBM) application.
- (LBM) Simulation took **2 Weeks** of hand coding low level OpenCL as opposed to **2 Hours** of H2C programming.

Performance

- Performance of H2C is similar or better to hand OpenCL.
- LBM showed a speedup of **60x** compared to sequential implementation
- K-Means showed an improvement of **7x** compared to original hand coded OpenCL version due to data layout transformations — good candidate for future DSL extensions.