Habanero-C (HC) constructs for OpenCL

- async copyin(ARGS1) copyout(ARGS2) at(place);
  - Asynchronously copy data between the device at place and the host.
  - Data to be copied to the device is specified by the copyin clause.
  - Data to be copied from the device is specified by the copyout clause.

- forasync point(ARGS) size(ARGS) seq(ARGS) at(place)(Body);
  - Asynchronously execute the body of the loop on the device, place.
  - Loop indices are specified by the point clause.
  - Loop dimensions are specified by the size.
  - Work-group sizes are specified by the seq clause.
  - Optional scratchpad clause to take advantage of local shared memory.

- finish{};
  - Ensures all the OpenCL tasks inside its scope are completed.

Example Transformation

```c
int i; int tile=64;
finish async copyin(posx, posy, posz) at(gpu);
while(...){
    //compute forces
    forasync indexed(size) seq(tile) at(gpu) scratchpad(posx, posy, posz) {
        int jj = 0; float wp=0;
        for(jj=0; jj<num_bodies; jj++){
            float dx = posx[jj] - posx[i];
            as[i].accx += dx * wp;
        }
    }
}
finish async copyout(accx, accy, accz) at(gpu);
```

```c
int i; int tile=64;
copy_to_device(posx, gpu);
copy_to_device(posy, gpu);
copy_to_device(posz, gpu);
copy_from_device(accx, gpu);
copy_from_device(accy, gpu);
copy_from_device(accz, gpu);
call functor;
}
```

Conclusion and Future Work

- High level languages necessary for portability and productivity.
  - OpenCL, CUDA are hard to learn.
- HC provides simple extensions to the C language to support heterogeneous computing.
- Performance of HC is comparable to hand coded implementation for many applications.
- Improve the code generation to support more benchmarks.
- Support for execution on multiple devices
  - Requires data distribution constructs.