PGAS-X Workshop 2011 Program	
9:00-9:30	Opening Remarks : John Leidel, Convey Computer Corp.
9:30-10:00	OpenSHMEM Research at University of Florida: Vikas Aggarwal & Prashanth Prakash
10:00-10:30	BREAK
10:30-11:30	Qthreads: Adapting to Changing Hardware: Kyle Wheeler, Sandia
11:30-1:30	LUNCH
1:30-2:30	Parallel Programming with Cilk Plus: William Leiserson, Intel
2:30-3:00	Open Discussion
3:00-3:30	BREAK
3:30-4:30	Open Discussion

Heterogeneous Parallelism

- Instruction Level Heterogeneity
  - Device has a rich instruction set capable of executing some number of memory and arithmetic instructions
    - Which implies the ability to actually debug machine state
  - Some sort of cross compilation environment exists
    - Optimizations on each respective architecture may be orthogonal
  - Generally depends upon some small runtime layer [user level or kernel level]
  - *Micro heterogeneity*
- Library Level Heterogeneity
  - Runtime/user libraries are required to access the compute device
    - Regardless of what the compute device really is
  - Memory is generally separate and explicitly mapped
    - Address spaces are not generally shared in this model… but can be partitioned
  - Asynchronous host/accelerator access is generally available, but not always
  - Getting close to the metal is sometimes difficult and/or painful
  - Portability of applications becomes dependent on the portability of the overlying runtime/execution libraries
  - *Macro heterogeneity*

How do we unite these two worlds?

What do we need to focus on in the future?

- Distinct models of parallelism
  - SIMD versus MIMD versus both
  - ILP versus hardware-aware contexts
  - How do the upper layers of the model map to functional equivalents?
- Memory Locality and Affinity
  - All memories are not created equal
  - BUT, programmers really like a shared memory view of the world
  - How does this affect memory allocation? [this is bigger than you think]
  - How does this affect
- Communication
  - Implicit communication primitives w/ explicit directives
  - Handle the communication between physical nodes and across heterogeneous boundaries, but give the user the ability to force the issue
- Execution Model//System Coherency
  - How does one efficiently overlap communication and compute at the runtime and execution model layer?
  - How does one propagate system hard/ soft errors?
    - Parity errors in data transfers?
    - Exception handling?
- Scheduling
  - Not all processors are created equal
  - Not all interconnects are created equal
  - Not all implementations of the same algorithm are created equal
  - *Hope for the best, plan for the worst*
- I/O
  - How do we perform mixed I/O in a system without disrupting the host kernel [or kernels]
  - *"I just want block I/O…. Faster!"*

All of these concepts underscored with ***RAS***