Feedback on OMP Breakout

- Lots of questions clarifying existing standard
- Interoperability with streams
  - General agreement that would be useful
    - Interoperate with existing CUDA code
    - Possible that target pragma could be used for most offloads, but critical kernel could still be CUDA
  - Difficulties in getting agreement from all vendors for a Nvidia specific feature
  - Implementation may not be straightforward
  - Lead into discussion of support for CUDA events
  - Accelerator/tasking subcommittees should take this up

- Specialization of functions for different devices
  - (didn’t Barcelona have a proposal for this over 10 years ago)
  - Not possible without a way to check what device you are on
  - Maybe super-if can help with this
  - Kokkos would be a customer for a feature like this
  - Maybe just host/no-host would be easier to get accepted
- Support for shared memory on GPU can be a big factor for performance
  - There should be ways to do this in OMP 5 with support for different classes of memory
  - May rely on Vendor choices on how different types are mapped to hardware
  - This can lead to interoperability problems if different compilers for the same machine make different choices
  - Currently, although vendors ARE talking to each other, no emphasis on this type of interoperability
    - Perhaps need a reference implementation
    - Clang/LLVM may be that?
    - Major features that are very likely to get into standar may be acceptable to community, but unlikely that constant churn would be acceptable

- Suggested that algorithms other than reduction (Scan for example) could be added to allow vendors to provide optimized implementations
- Decouple teams from target
- Discussion of updating the base languages and what limitations there may be
- First touch/next touch discussion
- Can we define subsets of the spec? … or deprecate features
    - Eg, tasks are required on GPU but impact performance of parallel for etc
- What would people like removed? … no real answers
- Can we add hints to let compiler assume that certain things wont be a problem?
- Error model?

Process improvements?

- More transparency on proposed extensions, maybe examples on the public repository?
- More time or more experimentation with prototype implementations before finalizing
- OpenMP has gotten large, and keeps growing, maybe offer subsets for application areas?
  - Maybe a version with no tasks, or even no runtime, in target regions?
- Increase participation of users at each site, and site reps in the committee
  - Maybe more hackathons, a working group?