Proposal

Mark Madler

1 General Design

This work will consist of modifying an existing compiler (gcc or llvm**) to support LOCO as a backend for disaggregated memory. Generally this compiler will look up address locations on loads and stores to determine whether it is local/remote and will also determine whether it is cached or not. Additionally all cached data will be tagged with a dirty delta to allow for seamless release consistency. On releases data will be written back to its home-node and only modified data will be written back (scatter). If two nodes wish to write back the same byte-area they are in a data race and it is the programmers fault.

There will be major components in this design. They are as follows:

- Load/store compiler modification. Need to detect these special loads and stores So this might be in LLVM's SelectionDAGBuilder.cpp in "visitStore" and "visitLoad". llvm-project/llvm/lib/CodeGen/SelectionDAG/SelectionDAGBuilder
- LOCO loads/stores need to be tagged at some point. Either these objects should be at special virtual addresses by modifying malloc things. I am thinking this will be done in clang (llvm-project/clang/lib/CodeGen/CGExprCXX.cpp) what I found here should work for calls with new, note sure about malloc. Malloc might be in CGCall??
- synchronization primitive modification for OpenMP.
- non-cached memory will be stored as a hash map.

some outstanding questions to answer:

- does the local node cache its own data?
- does a system like this exist for TCP/IP?
- specifics of design implementation. i.e does loco have all openMP primitives?

2 My TODO list

- get cmatose.c working on r320s to diagnose issue. (is it librdmacm?)
- change all config scripts in Odyssey.
- Finish this proposal.