Tloops
RAJA-Like Transformations in Kripke

LLNL
February 5, 2015

Adam J. Kunen
Overview

- **Kripke** - With a “RAJA-like” approach
- **Simple loops** – What’s going wrong?
- **Conclusion**
Kripke with a “RAJA” approach?

- Kripke is an Sn transport Mini-App
  - Proxy for ARDRA
  - 2k lines of C++
  - Supports Group-Set and Direction-Sets (PDT)
  - Supports 6 data layouts (“nestings”)
    - Explicit code for each layout
  - For details see NECDC 2014 presentation

- Kripke Templated Loops Variant (“tloops”)
  - Used templates to abstract:
    - Data layout
    - Loop nesting order
    - Loop bodies
    - Loop header
  - Why: Can write a kernel once, and use it for any data layout, any threading model, any architecture, etc. (Same reasons you would use RAJA)
Example *Pseudo Code*

**Master Variant (DGZ):**

Foreach (d in D) {
  Foreach (nm in NM) {
    Foreach (g in G) {
      Foreach (z in Z) {
        PHI[nm][g][z] = L[d][nm] * PSI[d][g][z]
      }
    }
  }
}

**Tloops Variant:**

Index<NEST> idx;

Loop<NEST>(
  MomDirLoop(),
  GroupLoop(),
  ZoneLoop(),
  [=]{
    PHI[idx.Phi()] = L[idx.L()] * PSI[idx.Psi()]
  }
)

**Abstraction:**

- Loop Nesting
- Loop Headers
- Lambda Loop Body
- Data Layout
Solver Performance for KP0
Speedup of TLoops over Master Running on RZMERL (TLCC2)
Kripke Serial Performance vs. Run Parameters
KP0 on RZMERL (TLCC2) with Various Compilers
Simplified Loop: \( X = cY \)

- Try and figure out what the compiler does and doesn’t like about Tloops variant of Kripke
  - Kripke Tloops is fairly complex, so it’s difficult to determine what constructs cause problems for compiler optimizers
- Try simple vector scaling kernel
- Loop0 thru loop6
  - Loop0 is simplest, “easiest” to vectorize loop
  - Loop1-6 add more and more of the loop abstractions we have in Tloops.
- “Loop0 should be easy for a compiler to optimize!” – My Assumption
Simplified Loop Results

X = cY Kernel Performance With Increasing Level of Abstraction Running on RZMERL (TLCC2)

Wall Time (Seconds)

- clang 3.5.0
- gcc 4.9.2b
- icc 14.0.174
- icc 15.0.133
- pgi 14.7.0

No Abstraction --------> Most Abstraction

X = cY Kernel Performance With Increasing Level of Abstraction Running on RZUSEQ (BG/Q)

Wall Time (Seconds)

- clang 3.6.0
- bgcc-4.7.2

No Abstraction ---> Most Abstraction