LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-MI-692819

DOE-COE Breakouts

J. R. Neely, M. W. Epperly

May 23, 2016

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

OpenMP Futures Breakout

David Richards, LLNL

Sriram Swaminarayan, LANL

Glendale AZ, 4/2016

Topics to cover

- Performance Portability
- Usability
- Memory management
- Execution
- …

Performance Portability Related

- **What is missing in OpenMP for performance portability?**
- **Is the OpenMP abstract machine adequate to reason about performance?**
  - Is it possible to define an adequate abstract machine or do we need to explicitly support/be aware of architectural differences?
- **There are different practices for different architectures**
- **Do you need better support for STL?**

Portability Discussion Notes

- **Current solutions seem to involve lots of ifs and ifdefs**
- **More descriptive options might help compiler Do The Right Thing**
  - Do you believe in Magic? Does this really work for more complex code examples
- **Even portability from compiler to compiler can be problematic**
- **One view is we only have two platforms, just choose your directives. That is enough.**
  - this does tend to limit scope of codes.
- **Basic unit of work - then design a separate mapping**
- **relationships between work units must be defined.**
- **Hardware can set algorithm preference. Then you're just stuck.**
- **Need a better stl.**

Usability Related

- **Best practices**
  - Does a ‘best practices’ document exist somewhere? Who maintains it?
  - What programming patterns work in OpenMP?
  - What are the characteristics of codes that admit such patterns?
  - Do we now need a new set of practices? Will community adopt it?
  - Different annotations for different architectures
- **What are most important interoperability concerns for OpenMP regarding:**
  - third party libraries?
  - other programming models?
  - other threading systems?
  - MPI?

Usability Discussion Notes

- **Appears that we could improve dissemination of best practice information.**
- **Discussion of classifications of codes**
  - Can your code nest? Are the nesting levels self similar?

Interoperability Discussion Notes

- **resource allocation handles to pass to library**
  - Like an MPI Communicator
- **Task constructs sort of help, but they aren't there yet.**
- **What might support for "unbound" threads look like?**
- **Libraries have been typically developer as context free, because there was no context to worry about.**
- **Support for C++ is can be problematic**
- **Co-existing with other threading models could be better**

Memory Management Discussion Notes

- Is OpenMP adequate to manage the lower levels of the memory hierarchy?
- Do we need additional facilities and what would they look like?
- Could code written with such features ever be considered performance portable?

- Question of memory support applies to any language/model?
- I have read only data, used heavily (descriptive approach)
- Talking about data traits seems useful.
- but we probably don't know where the land mines are
- The fact that directives aren't "sticky" is limited. (Solutions can be imagined)

Execution, Runtime, etc…

- **Do you need more control over sequential optimization?**
  - Tiling of loops?
  - Automated unrolling?
  - Support for wavefront loops?
- **Do we need directives for specifying dependencies between loops?**
  - Loop fusion?
  - Pipelining?
  - Intermediate variables?
  - Dependency list?

Execution Discussion Notes

- **Tension between portability and performance is once again evident.**

- **doacross - yes**
- **collapse non-rectangular loops – yes**
  - or non closely nested loops (ugly and hard transformation?)
- **tiling - yes, but limited audience**

- **standardized directives could be a benefit, but feature creep**
- **Knowing staging and interaction of directives is important.**
- **examples needed!!**
- **Explorations of auto-tuning or run time adaptive tuning**

Backups in case discussion dies…

Usability Improvements

- **Best practices**
  - Does a ‘best practices’ document exist somewhere? Who maintains it?
  - Do we now need a new set of practices? Will community adopt it?
- **It is hard to do OpenMP well**
  - How can I do OpenMP wrong? Let me count the ways…
  - Different annotations for different architectures
  - No heuristics easy to find (see best practices above)
- **Should OpenMP be more descriptive?**
- **Is accelerator support adequate?**
- **Is support for function pointers needed?**
  - Do we need access to function pointers generated by compiler?
- **Is support for lambdas needed?**
- **Do we need API routines to understand how the device works?**
- **Are OpenMP tasks usable?**
- **Are overheads too high? What would you give up to get lower overheads?**

Memory Management

- **Do we need Futures in OpenMP?**
  - Easier specification of dependencies
- **What memory management improvements do you want to see?**
  - Hierarchies?
  - Affinity? (NUMA aware)
  - Deep copies?
  - Allocations? (is target / map enough)
- **Should OpenMP infer target directives based on the context?**