Building an Open Community Runtime (OCR) framework for Exascale Systems

Birds of a Feather Session, SC12, Salt Lake City

November 14, 2012

Organizers: Vivek Sarkar, Barbara Chapman, William Gropp, Rob Knauerhase
Agenda

1. OCR Goals and Approach (10 minutes)
   – Vivek Sarkar
2. Lightning Talks (5 minutes each)
   – Barbara Chapman
   – Bill Gropp
   – Rich Lethin
3. Overview of OCR v0.7 open source release (10 minutes)
   – Rob Knauerhase
4. Hands-on demo of OCR v0.7 release (10 minutes)
   – Romain Cledat
5. Discussion and wrap-up
   – All
Runtime Challenges for Exascale and Extreme Scale Computing

- Performance of extreme scale systems will be driven by parallelism, and constrained by programmability, energy, data movement, and resilience.

- Past approaches to parallel runtime systems focused on innovation in isolated layers that focused on isolated resources e.g., communication runtimes for network resources, task-scheduling runtimes for compute resources.

- A cooperative (rather than isolated) approach must be pursued to address key challenges in management of shared resources in extreme scale runtime systems.
Motivation for an Open Community Runtime

• A runtime framework that ...
  – is representative of execution models expected in future extreme scale systems
  – can be targeted by multiple high-level programming systems
  – can be effectively mapped on to multiple extreme scale platforms
  – can be extended and customized for specific programming and platform needs
  – can be used to obtain early results to validate new ideas
  – is available as an open-source testbed

• Approach:
  – Address revolutionary challenges collaboratively
  – Reduce duplication of infrastructure effort, while
Summary of OCR Open Source Project

- Hosted on 01.org (details to follow)
- Goals
  - Modularity
  - Stable APIs
  - Extreme flexibility in implementation
  - Transparency
- Development process
  - Continuous integration
  - Quarterly milestones
  - Mailing lists for technical discussions, build status, etc
- Organization
  - Steering Committee (SC) --- sets overall strategic directions and technical plans
  - Core Team (CT) --- executes technical plan and decides actions to take for source code contributions
  - Membership of SC and CT will turn over periodically based on level of participation
### Inaugural Membership for OCR
#### Steering Committee
- Vivek Sarkar (Rice U.)
  - Inaugural Chair
- Barbara Chapman (UH)
- Guang Gao (UD)
- Bill Gropp (UIUC)
- Rob Knauerhase (Intel)
- Rich Lethin (Reservoir)

#### Core Team
- Zoran Budimlic (Rice)
- Vincent Cave (Rice)
- Sanjay Chatterjee (Rice)
- Romain Cledat (Intel)
- Sagnak Tasirlar (Rice)
OCR Acknowledgments

• Design strongly influenced by
  – Intel Runnemede project (via DARPA UHPC program)
    – power efficiency, programmability, reliability, performance
  – Codelet philosophy – Prof. Gao’s group at U. Delaware
    – implicit notions of dataflow
  – Habanero project – Prof. Sarkar’s group at Rice U.
    – data-driven tasks, data-driven futures, hierarchical places
  – Concurrent Collections model – Intel Software/Solutions Group
    – decomposition of algorithm into steps/items/tags, tuning
  – Observation-based Scheduling – Intel Labs
    – monitoring and dynamic adaptation to load and environment
  – Machine Description – Prov. Sandrieser, University of Vienna

• Partial support for the OCR v0.7 release was provided through the X-Stack program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR)
OCR Assumptions

• A fine-grained, asynchronous event-driven runtime framework with movable data blocks and sophisticated observation enables the next wave of high-performance computing

• Fine-grained parallelism helps achieve concurrency levels required for extreme scale

• Asynchronous events and movable data blocks help cope with data movement, non-uniformity, heterogeneity, and resilience in extreme scale applications and platforms

• Sophisticated observation enables introspection into system behavior, feedback to OCR client, and adaptation based on algorithmic and performance tuning
OCR High-level Design

• Application/algorithm decomposition exposes greater parallelism than current thread/barrier models

• Separation of concerns among programming environment, hero programmer, tuning hints

• Event-Driven Runtime manages tasks and data blocks to adapt to changes in platform behavior (resilience, machine configuration changes, mission/goal changes), while obeying all control and data dependences
Agenda

1. OCR Goals and Approach (10 minutes)
   - Vivek Sarkar
2. Lightning Talks (5 minutes each)
   - Barbara Chapman
   - Bill Gropp
   - Rich Lethin
3. Overview of OCR v0.7 open source release (10 minutes)
   - Rob Knauerhase
4. Hands-on demo of OCR v0.7 release (10 minutes)
   - Romain Cledat
5. Discussion and wrap-up
   - All
Thoughts on an Open Runtime

William Gropp

www.cs.illinois.edu/~wgropp
Hybrid Programming and Shared Resources

• Hybrid model is a good thing
• But resources are shared:
  ♦ Network
  ♦ Memory bandwidth
  ♦ Compute cores
  ♦ Etc.
• How can we make the elements of the hybrid model work together?
Which programming runtime controls resources?

- Currently, most assume that all resources are dedicated to themselves
  - E.g., MPI runtime assumes all cores are used by MPI; OpenMP assumes cores available for OpenMP.
- Allocation of resources is not static
  - E.g., MPI sometimes needs an “agent” for communication progress, esp for nonblocking collective, passive-target RMA, Redezvous point-to-point progress; helpful to take a core for this
- Solution to date: tell programming runtimes at startup what resources they have (if you are lucky)
- Needed: Ways for multiple runtimes to negotiate the resources to share, at startup and during execution
  - Note: Not a common runtime that they all use
Common Capabilities

• Much desire with a common runtime on top of which all parallel programming methods may be implemented
  ♦ Obvious advantages – shared code, more rapid development

• Unfortunately, not realistic
  ♦ Programmer productivity can be related (in part) to reducing the size of basic element that can be used and still get good performance (everyone wants this to be a single word)
  ♦ Performance at this end is extremely sensitive to exact semantics of hardware, implementation (library) overhead, including even length of call list and data alignment
What Can We Do?

• Alternative: Provide common capabilities for cases that are *not* sensitive to these issues (typically operations involving larger blocks of data)

  ♦ Need to be extensible so that customized interfaces and implementations can be used for the performance critical

• Implications

  ♦ Common runtime can provide some services but critical ones will need to designed for and implemented to specific platforms
    • *This work can be shared inside a community, mostly as code examples*

  ♦ Runtime must be extensible, with ability to plug in specialized services
Thoughts on an Open Runtime

William Gropp

www.cs.illinois.edu/~wgropp
Hybrid Programming and Shared Resources

- Hybrid model is a good thing
- But resources are shared:
  - Network
  - Memory bandwidth
  - Compute cores
  - Etc.
- How can we make the elements of the hybrid model work together?
Which programming runtime controls resources?

- Currently, most assume that all resources are dedicated to themselves
  - E.g., MPI runtime assumes all cores are used by MPI; OpenMP assumes cores available for OpenMP.
- Allocation of resources is not static
  - E.g., MPI sometimes needs an “agent” for communication progress, esp for nonblocking collective, passive-target RMA, Redezvous point-to-point progress; helpful to take a core for this
- Solution to date: tell programming runtimes at startup what resources they have (if you are lucky)
- Needed: Ways for multiple runtimes to negotiate the resources to share, at startup and during execution
  - Note: Not a common runtime that they all use
Common Capabilities

• Much desire with a common runtime on top of which all parallel programming methods may be implemented
  ♦ Obvious advantages – shared code, more rapid development

• Unfortunately, not realistic
  ♦ Programmer productivity can be related (in part) to reducing the size of basic element that can be used and still get good performance (everyone wants this to be a single word)
  ♦ Performance at this end is extremely sensitive to exact semantics of hardware, implementation (library) overhead, including even length of call list and data alignment
What Can We Do?

• Alternative: Provide common capabilities for cases that are *not* sensitive to these issues (typically operations involving larger blocks of data)
  ♦ Need to be extensible so that customized interfaces and implementations can be used for the performance critical

• Implications
  ♦ Common runtime can provide some services but critical ones will need to designed for and implemented to specific platforms
    • *This work can be shared inside a community, mostly as code examples*
  ♦ Runtime must be extensible, with ability to plug in specialized services
Agenda

1. OCR Goals and Approach (10 minutes)
   - Vivek Sarkar

2. Lightning Talks (5 minutes each)
   - Barbara Chapman
   - Bill Gropp
   - Rich Lethin

3. Overview of OCR v0.7 open source release (10 minutes)
   - Rob Knauerhase

4. Hands-on demo of OCR v0.7 release (10 minutes)
   - Romain Cledat

5. Discussion and wrap-up
   - All
OpenMP Language and Implementation Technologies Need a Powerful Runtime

Barbara Chapman
University of Houston
OCR BOF, SC12

Acknowledgements: NSF CNS-0833201, CCF-0917285; DOE DE-FC02-06ER25759

http://www.cs.uh.edu/~hpctools
OpenMP 4.0 Release Candidate 1

- Presented at OpenMP BOF (yesterday)
  - Now on OpenMP website
- Candidate topics:
  - Affinity and locality
  - SIMD extensions
  - Error model
- On-going work:
  - Accelerator
  - Tools interface
The Accelerator Model

- **Execution Model:** Offload data and code to accelerator
  - Target construct creates tasks to be executed by devices
  - Initial device thread waits to execute the device tasks
- **Memory Model:**
  - Data may be copied in or out, allocated on accelerator
  - Copies of shared data are synchronized explicitly or implicitly at end of the target construct regions.
- **Integration with tasking extensions**
- See technical report
OpenMP 4.0 Affinity Proposal

- OpenMP Places and thread affinity policies
  - OMP_PLACES to describe places
  - affinity(spread|compact|true|false)
- SPREAD: spread threads evenly among the places
  - spread 8
- COMPACT: collocate OpenMP thread with master thread
  - compact 4
OpenMP Error Model

® Cancel directive

- #pragma omp cancel [clause[ [, ]clause] …]
- !$omp cancel [clause[ [, ]clause] …]
- Clauses: parallel, sections, for, do
Toward Asynchronous OpenMP Execution

- May be difficult for user to express computations in form of task graph
- Compiler translates “standard” OpenMP into collection of work units (tasks) and task graph
- Analyzes data usage per work unit
- Trade-off between load balance and co-mapping of work units that use same data
- What is “right” size of work unit?
  - Might need to be adjusted at run time

Data-Driven Model with OpenMP Tasking Extensions at UH

1) `#pragma omp task out [(data – reference – list)]`
2) `#pragma omp task in [(data – reference – list)]`

Items listed in the data reference list can be thought of as synchronization identifiers called ‘task tags’

Extensions proposed follow a topological sort

- a task can only depend on a task which is before it in program order
DARWIN: Feedback-Based Adaptation

- Dynamic Adaptive Runtime Infrastructure
  - Online and offline (compiler or tool) scenarios
- Monitoring
  - Capture performance data for analysis via monitoring
  - Relate data to source code and data structures
  - Apply optimization and/or visualize
  - Demonstrated ability to optimize page placement on NUMA platform; results independent of numthreads, data size

Besar Wicaksono, Ramachandra C Nanjegowda, and Barbara Chapman. A Dynamic Optimization Framework for OpenMP. IWOMP 2011
Runtime False Sharing Detection

OCR Support for Legacy Applications

OCR needs to be able to support current and future programming model

- Very important to support legacy apps
- Opens up to a wide range of apps
- Novel implementation techniques for existing models
- Explore new features, limitations, new programming models
Goals for Legacy Code Migration

- Support legacy MPI and OpenMP codes in XPRESS
- Develop a migration path for OpenMP and MPI application toward new execution model
- Communicate XPRESS experiences back to standards committee
  - Potentially suggest extensions to OpenMP and MPI with features from XPRESS
The end
Agenda

1. OCR Goals and Approach (10 minutes)
   – Vivek Sarkar

2. Lightning Talks (5 minutes each)
   – Barbara Chapman
   – Bill Gropp
   – Rich Lethin

3. Overview of OCR v0.7 open source release (10 minutes)
   – Rob Knauerhase

4. Hands-on demo of OCR v0.7 release (10 minutes)
   – Romain Cledat

5. Discussion and wrap-up
   – All
Reservoir presentation

• (See embedded PDF – after SC12, we’ll post all the slides in the same format.)
Agenda

1. OCR Goals and Approach (10 minutes)
   - Vivek Sarkar

2. Lightning Talks (5 minutes each)
   - Barbara Chapman
   - Bill Gropp
   - Rich Lethin

3. Overview of OCR v0.7 open source release (10 minutes)
   - Rob Knauerhase

4. Hands-on demo of OCR v0.7 release (10 minutes)
   - Romain Cledat

5. Discussion and wrap-up
   - All
What’s not in OCR v0.7

• It’s **scaffolding**,  
  – just a framework

• It’s **not** the Sears Tower! (yet)
What’s in OCR v0.7

• Event-driven tasks (EDTs)
  – can be processes, functions or codelets (open research question)
    – decomposition is up to programmer & compiler
    – could be data-parallel within themselves

• Events (Dependences)
  – specified explicitly as contingencies on which EDTs are initiated
    – EDTs can fire anytime after all their dependences are met
  – several types of dependences
    – control dependences: B cannot start until A finishes
    – data dependences: B cannot start until inputs D1 and D2 are available, and processing on D3 has finished
    – independent events (e.g. triggers, environment, …)
  – dependences are specified as GUIDs throughout the system
What’s in OCR v0.7

• Memory datablocks
  – replacement for malloc()
  – contains semantically-meaningful metadata that runtime can use
  – relocatable by runtime for power, reliability, ...
    – exploring hardware assistance; no movement in v0.7 release
  – allows exploitation (or modeling) of NUMA, scratchpad memories, etc.
    – e.g. instrumentation to infer energy usage from different placements and configurations

• Machine description
  – XML schema plus conforming XML documents
    – based largely on U. Vienna’s Platform Description Language
  – allows expression of hw configuration (cores, memory, interconnect)
    – exploration of same decompositions on different hardware, real or simulated
  – current state: present, but barebones, not fully used
Implementation Details

• Complete but non-optimized implementation
  – performance is not (yet!) a goal

• Runs on top of Linux
  – shows functionality without having to build a whole OS
  – other versions running on simulation (UHPC, X-stack)

• Supports “hero programmers” for nontrivial apps
  – pending programming model integrations

• Modularity as a goal whenever possible
  – for ease of subsystem replacement, augmentation, …
  – supporting other research using OCR components
What’s coming in OCR v(0.7 ++)

• Distribution
  – runtime functionality across “nodes” w/separate memory spaces
    – MPI integration under the covers

• Tuning expression
  – hints via better groupings for temporospatial locality
    – leverage hierarchical place trees and CnC affinity groups, ...

• Machine description improvements
  – better integration with runtime
  – ongoing observation of machine state (load, failures, ...)

• Different underlying thread support
  – e.g. Sandia Qthreads, direct mapping to hw threads
OCR resources

• Project homepage at http://01.org/projects/open-community-runtime
• Public repository on github http://github.com/01org/ocr
• Mailing lists
  – ocr-announce
  – ocr-devel
  – ocr-discuss
  – ocr-build
• Wiki and so forth coming soon

http://01.org/projects/open-community-runtime
Links to source code and mailman subscription pages

Copy of today’s slides
How you can get involved

• Runtime development
  – soliciting code contributions; we can use more brains/hands!
  – build a new subsystem, or adapt OCR to your existing research

• Develop/port applications
  – by-hand or compiler-driven decomposition into EDTs
  – explore behavior of different types of algorithms and tunings
  – enable execution on different machine types (including research architectures)

• Join the discussion mailing list
  – offer input about connections to other work, insight into areas in which you have expertise/experience
Live demonstration
Smith-Waterman implementation

```c
ocrEdtCreate(&task_guid, smith_waterman_task, 9, NULL,
            (void**) p_paramv, PROPERTIES, 3, NULL);

ocrAddDependency(tile_matrix[i][j-1].right_column_event_guid,
                task_guid, 0);
ocrAddDependency(tile_matrix[i-1][j].bottom_row_event_guid,
                task_guid, 1);
ocrAddDependency(tile_matrix[i-1][j1].bottom_right_event_guid,
                task_guid, 2);

ocrEdtSchedule(task_guid);
```
OCR Comparison with OpenMP
(Smith-Waterman algorithm)

Input set of ~37k nucleotides
(see http://en.wikipedia.org/wiki/Smith-Waterman_algorithm)
Questions?
Comments?
Unbridled enthusiasm?

(If you did not receive a flyer with information and the API cheat sheet, please pick one up on the way out!)