The Open Community Runtime Framework for Exascale Systems

Birds of a Feather Session, SC13, Denver
November 19, 2013

Organizers: Vivek Sarkar (Rice)
Rob Knauerhase (Intel)
Rich Lethin (Reservoir Labs)

OCR web site

SC13 Survey URL -- http://bit.ly/sc13-eval

Open Community Runtime

Agenda

1. Introduction & Motivation
   – Vivek Sarkar, Rice U.
2. Lightning Talks
   – Roger Golliver, UIUC
   – Benoit Meister, Reservoir Labs
3. OCR "state of the union"
   – Rob Knauerhase, Intel
4. Live demo of OCR v0.8 & adaptability
   – Vincent Cave, Rice
5. Next Steps
   – Rob Knauerhase
6. Closing, Q&A/discussion, wrap-up
   – Vivek Sarkar

Challenges for Exascale & Extreme Scale Systems

- Characteristics of Extreme Scale systems in the next decade
  - *Massively multi-core (~ 100's of cores/chip)*
  - *Performance driven by parallelism, constrained by energy & data movement*
  - *Subject to frequent faults and failures*
- Many Classes of Extreme Scale Systems

Mobile, < 10 Watts, $O(10^1)$ concurrency

Embedded, 100's of Watts, $O(10^3)$ concurrency

Departmental, 100's of KW, $O(10^6)$ concurrency

Data Center > 1 MW, $O(10^9)$ concurrency

Key Challenges

Concurrency
Energy efficiency
Locality
Resiliency

References:
- DARPA Exascale Study report, 2008
- DARPA Exascale Software study report, 2009

Open Community Runtime

Performance Variability is on the rise

- Concurrency --- increased performance variability with increased parallelism
- Energy efficiency --- increased performance variability with increased non-uniformity and heterogeneity in processors
- Locality --- increased performance variability with increased memory hierarchy depths
- Resiliency --- increased performance variability with fault tolerance adaptation (migration, rollback, redundancy, …)

Increasing performance variability ➔ runtime becomes the critical component of the exascale software stack

Open Community Runtime

Evolutionary vs. Revolutionary Approaches to Extreme Scale Runtime Systems

- Wide agreement that execution models for extreme scale systems will differ significantly from past execution models
- Shoehorning a new execution model into an old runtime system is counter-productive
- Instead, make a fresh start but carry forward reusable components from current runtime systems as appropriate

➔Motivation for Open Community Runtime framework that …

- is representative of future execution models
- can be targeted by multiple high-level programming systems
- can be mapped on to multiple extreme scale platforms
- is available as an open-source testbed
- reduces duplication of new infrastructure efforts
- **enables us to address revolutionary challenges collaboratively**

Open Community Runtime

Summary of OCR Open Source Project

- Hosted on 01.org since 2012 (details to follow)
- Goals
  - Modularity
  - Support for multiple programming systems e.g., programming languages, libraries, compilers, DSLs, …
  - Support for hardware platforms e.g., homogeneous manycore, heterogeneous accelerators, clusters, …
- Development process
  - Continuous integration
  - Development plans driven by community milestones
- 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
  - Users and Contributors --- members of OCR community i.e., you!!

Current OCR Steering Committee and Core Team Membership

Steering Committee

- Vivek Sarkar (Rice U.)
  - Inaugural Chair
- Barbara Chapman (UH)
- Guang Gao (UD)
- Bill Gropp (UIUC)
- Rishi Khan (ETI)
- Rob Knauerhase (Intel)
- Rich Lethin (Reservoir)
- Wilf Pinfold (Intel)

Core Team

- Zoran Budimlic (Rice)
- Vincent Cave (Rice)
- Sanjay Chatterjee (Rice)
- Romain Cledat (Intel)
- Mark Glines (ETI)
- Benoît Meister (Reservoir)
- Sagnak Tasirlar (Rice)
- Nicolas Vasilache (Reservoir)

OCR Acknowledgments

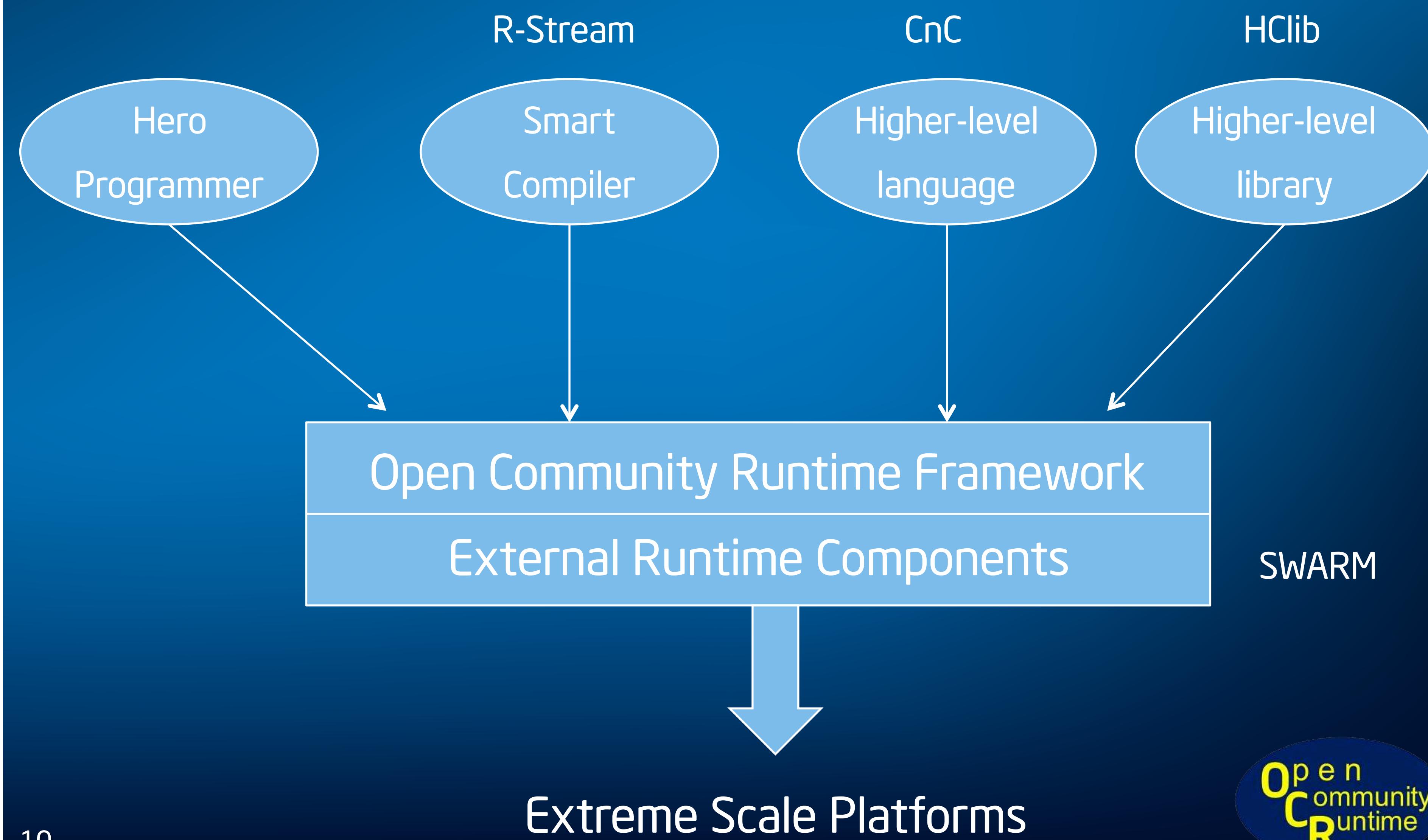
- Design strongly influenced by
  - Intel Runnemede project (via DARPA UHPC program)
    - power efficiency, programmability, reliability, performance
  - Codelet philosophy – Prof. Gao's group at U. Delaware
    - implicit notions of dataflow
  - Habanero project – Prof. Sarkar's group at Rice U.
    - 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

- *Partial support for the OCR development 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 performance variability, data movement, non-uniformity, heterogeneity, and resilience in extreme scale systems
- *Sophisticated observation* enables introspection into system behavior, feedback to OCR client, and adaptation based on algorithmic and performance tuning

OCR Vision


R-Stream
CnC
HClib
Hero Programmer
Smart Compiler
Higher-level language
Higher-level library
Open Community Runtime Framework
External Runtime Components
SWARM
Extreme Scale Platforms

Agenda

1. Introduction & Motivation
   - Vivek Sarkar, Rice U.

Lightning Talks
   - Roger Golliver, UIUC
   - Benoit Meister, Reservoir Labs

3. OCR "state of the art"
   - Rob Knauerhase, Intel
4. Live demo of OCR v0.8 & adaptability
   - Vincent Cave, Rice
5. Next Steps
   - Rob Knauerhase
6. Discussion and wrap-up
   - Vivek Sarkar

UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

LULESH OCR Experience

Roger Golliver

illinois.edu

LULESH 1.0.1 Benchmark

- Proxy app for shock hydrodynamics from LLNL
- Started with the C++/OpenMP version
  - Collected other versions to experiment with and look for where parallelism was exploited
- Perfect sample app for experimentation
  - Reasonable size, for all day edit sessions
  - In C++ but modest use of C++ features, so easily translated down to C
  - Well organized access to data
  - Stable results (gcc,icc)x(-O[0-1])x(Serial,OMP)
  - Modest use of standard libraries

Requirements for OCR

- Transition to C
  - Methods to functions
  - Data Classes to structures
  - Overloaded functions to multiple versions
- Transition array and structures to DBs.
  - malloc, global
- Transition functions to EDTs.
  - move return value to output parameter
  - Parameter Signature to ocrEdt_t
- Transition functions call/return organization to dynamically created and scheduled EDTs
- Transition OMP loop level parallelism to fork/join EDTs
  - Lots of these

Macros for Translation Support

- As part of the translation process I was making the LULESH source more "abstract"
  - DRAM_MALLOC() as malloc()
    run withC99/Cilk
    DRAM_MALLOC() as ocrDbCreate
    and run with OCR
  - DRAM_MALLOC() as upc_global_alloc()/SHARED
    run with UPC and check SHARED pointer usage
- This allowed my typo and parallelization errors to be caught in a familiar debug environment

Macros for Loop Parallelization

- For the parallel loops the same abstraction and the refinement could be done.
  - PAR_FOR for C&OMP is #pragma omp for / for(;;){}
  - PAR_FOR for cilk is cilk_for(;;){}
  - PAR_FOR for UPC is upc_forall(;;;){}
  - PAR_FOR for Habanero C is forasync IN() OUT() INOUT() POINT() SEQ() {}
- Habanero C is particularly nice step before transitioning from arrays/functions to DBs/EDTs
  - Scalars IN() can go to paramv[]
  - Arrays in in() out() inout() get converted to DBs and their guids go in depv[]

illinois.edu

Final Steps to EDT

- Habanero C is close syntactically and semantically to EDTs & data blocks.
- From initial habanero version
  - finish{ async IN(list) {....}}
  - finish { async IN(list) edt(list);
  - finish { async IN() edt(paramc,paramv[],depc,depv[]) }}
- Then as OCR
  - ocrEdtCreateTemplate()
  - ocrEdtCreate()
  - ocrAddDependence()

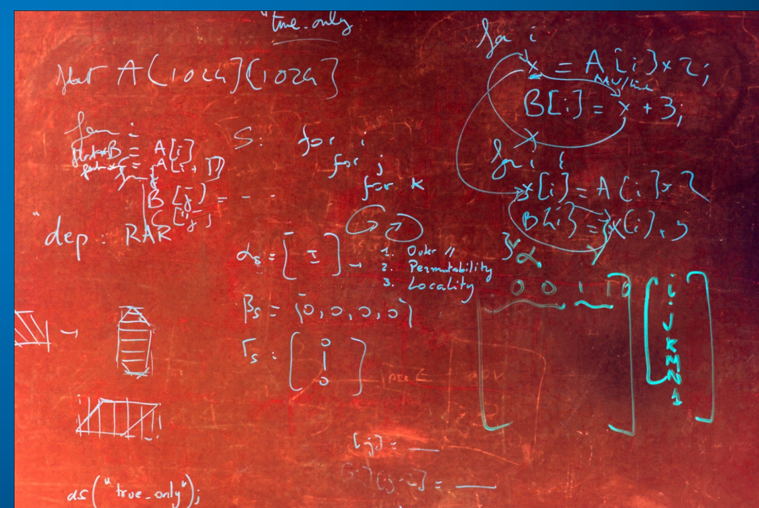
Status/Future Plans

- Initially the parallel abstraction support was done with C #define macros
- But when attempting to do the many parallel loops, there was to much reparative editing to turn the OMP loop bodies into EDTs
  - Compiler support for EDT extraction is critical
- Working on a more powerful set of `m4` macros that minimize the source changes required
- Waiting for a Habanero-C compiler that translates directly to OCR.

Automatic Parallelization to OCR using the R-Stream Compiler

Birds of a Feather Session, SC13, Denver

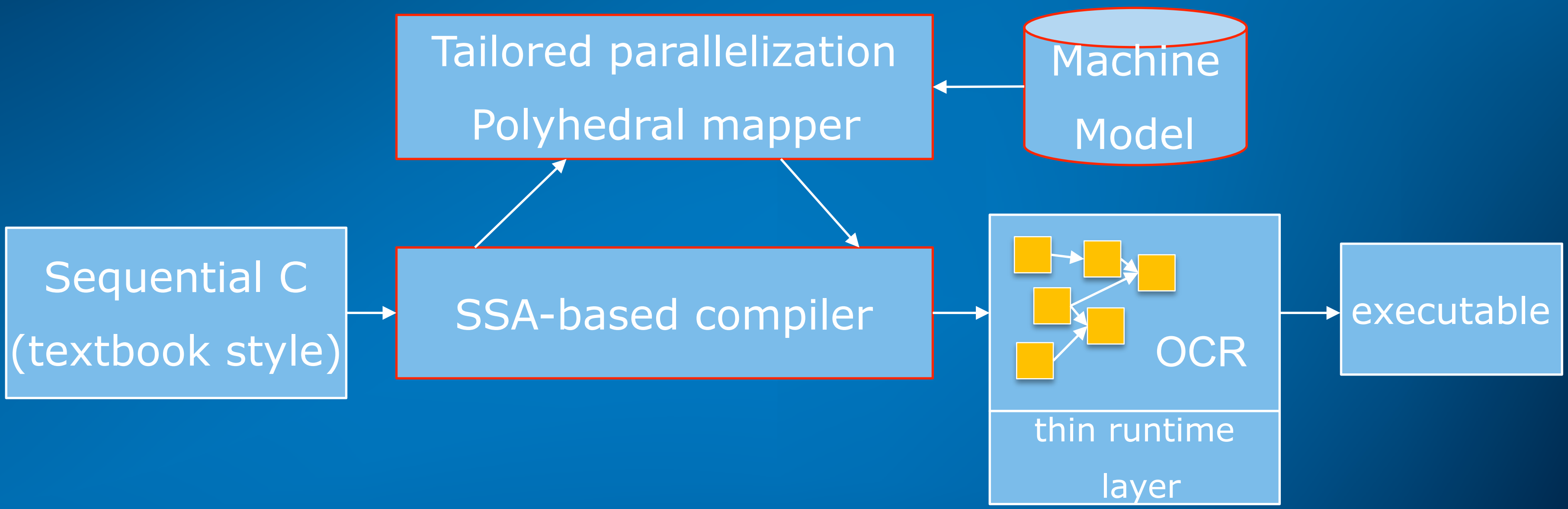
November 20, 2013



Reservoir Labs

http://www.reservoir.com

R-Stream compiler: flow



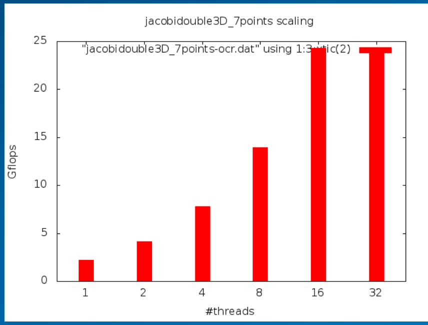
- Source-to-source compiler
- Automatic parallelization of dense array loop codes
- Generates code for a variety of programming models
  - OpenMP, Pthreads, SWARM, CnC, CUDA and more

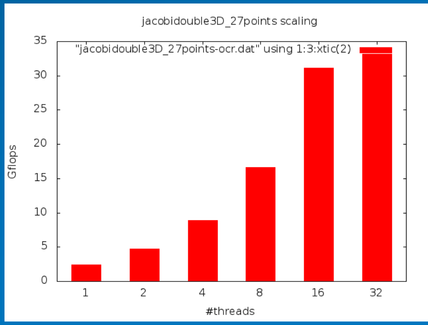
Mapping process to OCR

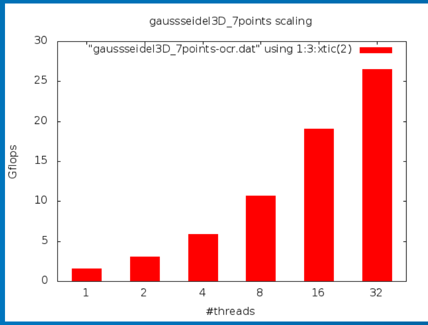
- Model all iterations and their data access
  - Polyhedral model
- Partition iterations into tasks
  - Data locality optimization (tasks must use caches well)
  - Task size: load balancing vs reuse
- Infer inter-task dependences from data accesses
  - Builds a notional task graph
  - Optimizations remove redundant & transitive dependences
- Generate code
  - Global or per-task knowledge of graph edges (events)
- Variants
  - Hierarchical code generation
  - Memory-oblivious code generation

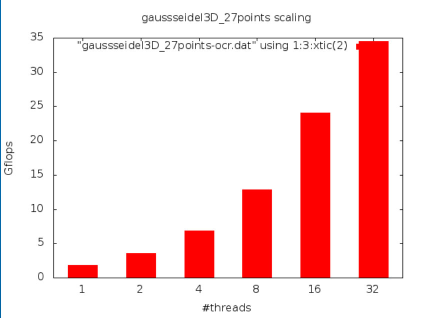
R-Stream: High points

- Fully automatic mapping path from textbook-style C









- Good scaling
  - OCR departs from bulk-synchronous models
  - Point-to-point synchronizations enable scalable load balancing
- Tuning options
  - Push button, hand-tuning or w/ non-invasive auto-tuner (ARCC)
- Commercial product with free academic licenses, government SBIR rights.
- https://www.reservoir.com/product/r-stream/

Agenda

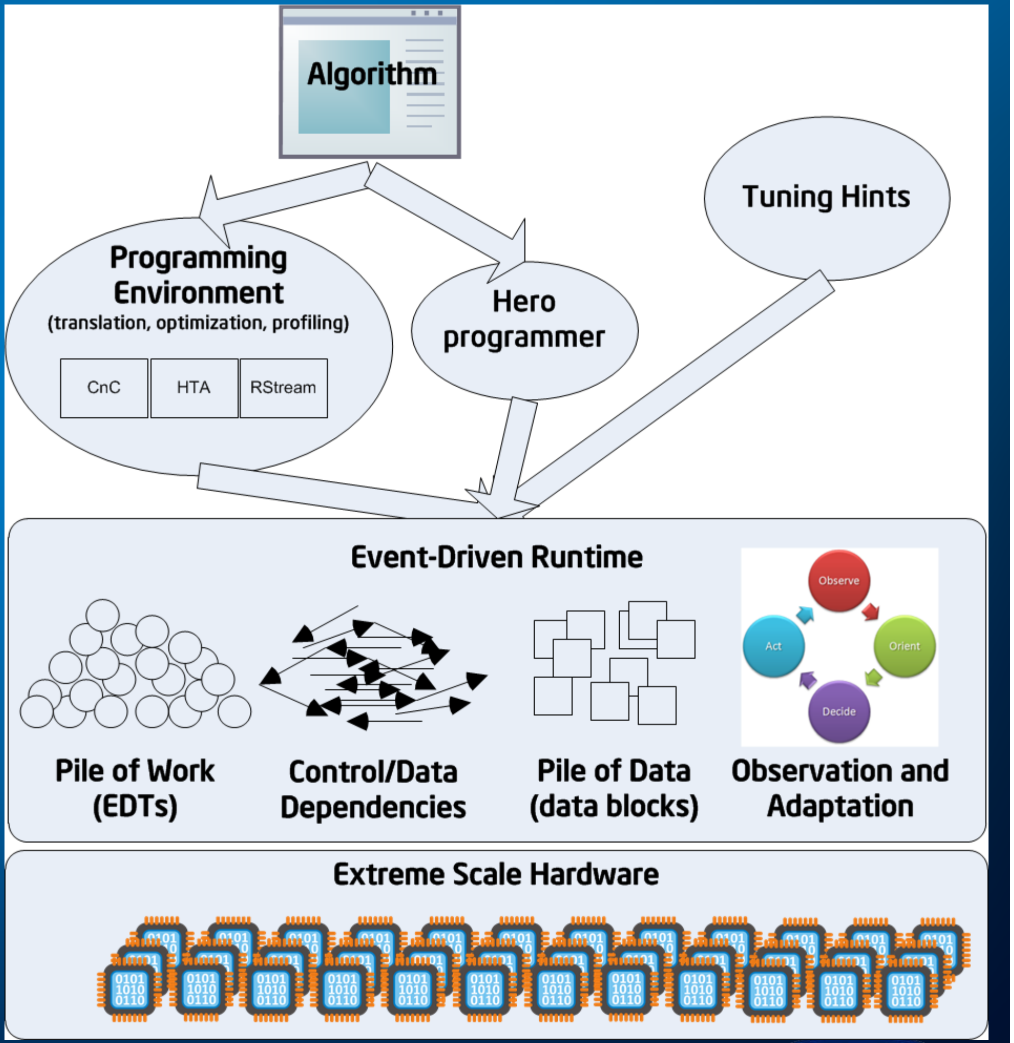
1. Introduction & Motivation
   – Vivek Sarkar, Rice U.
2. Lightning Talks
   – Roger Golliver, UIUC
   – Benoit Meister, Reservoir Labs
OCR "state of the union"
   – Rob Knauerhase, Intel
4. Live demo of OCR v0.8 & adaptability
   – Vincent Cave, Rice
5. Next Steps
   – Rob Knauerhase
6. Closing, Q&A/discussion, wrap-up
   – Vivek Sarkar

Open Community Runtime

Open Community Runtime

- Embodies a Fine-grained, Event-driven execution model
  - application/algorithm decomposition into fine-grained tasks activated by satisfaction of dependences
  - exposes greater parallelism than current thread/barrier models

- Runtime manages tasks and data blocks to adapt to changes in platform behavior & user policies, while obeying all control and data dependences



Recent Advances in OCR

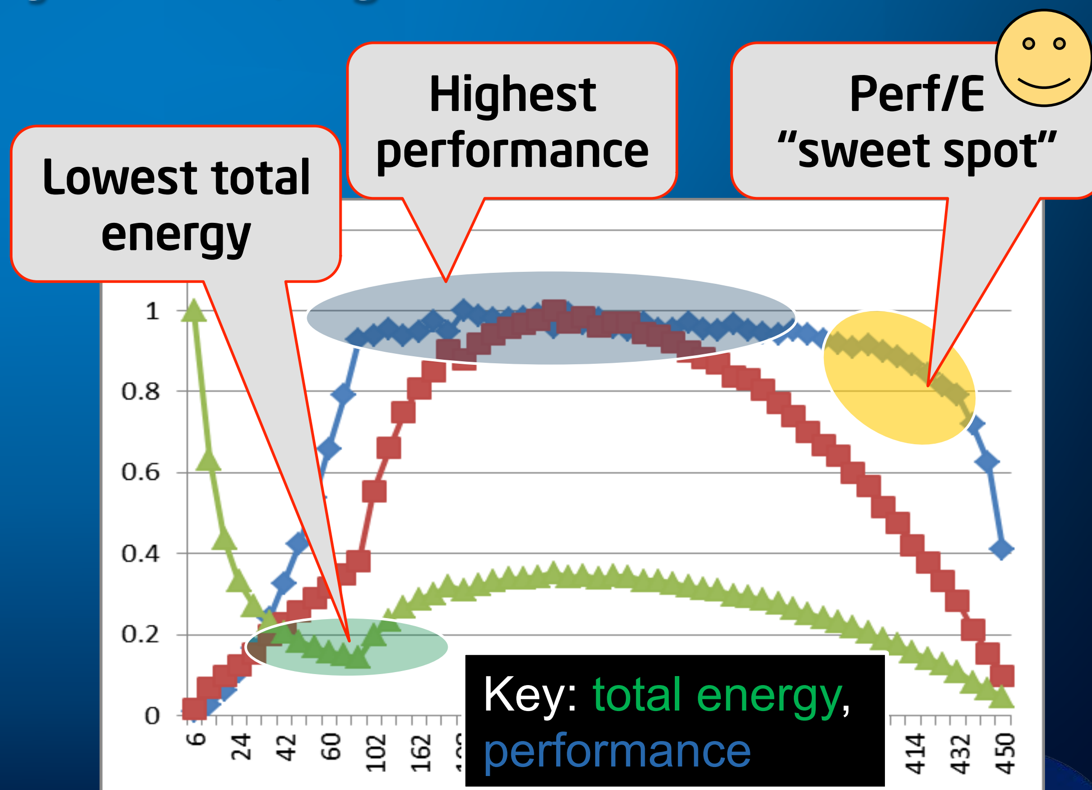
- Compiler generation of tasks
  - R-Stream (mentioned above)
  - Habanero-C HClib (upcoming demo)
  - Intel Concurrent Collections, others
- Better work-stealing scheduler
- Progress on adapting to policy
- Progress on "tuning" language and APIs
  - Including initial dynamic adaptation code
- New machine description features
- Distribution
- Use of hw power monitoring features
- Bugfixes & optimization

Open Community Runtime

Example Policies

- Example of matrix multiply
  - changing tile size/shape changes performance and energy consumed (# tasks available, size of each task, memory access, etc.)
- Allows flexibility of policy choices, e.g.
  - max performance
  - minimum energy
  - best ratio
  - and others

- **Runtime schedules differently based on policy**
  - no changes to source code, algorithm, etc.



Graph data from Cyclops machine,
source E. Garcia Ph.D. thesis, U. Del, (pub. 1/2014).

Example: Unbalanced Tree Search

- What UTS is/does*
  - parallel benchmark: exhaustive search of a large unbalanced tree
  - parameterized generation of tree nodes, predefined benchmark workloads
    - (we're using standard config T3XXL, for those who care)
  - includes app-level work stealing
- Started with thread-parallel benchmark, converted to HClib and OCR
  - we add alteration of chunk size to show differences
- Developed on different machines with diff core counts
  - Running live today on 32-hw-thread machine

*See also `http://sourceforge.net/p/uts-benchmark/wiki/Home/`
"UTS: An Unbalanced Tree Search Benchmark", Olivier et al. LCPC'06

Agenda

1. Introduction & Motivation
   - Vivek Sarkar, Rice U.
2. Lightning Talks
   - Roger Golliver, UIUC
   - Benoit Meister, Reservoir Labs
3. OCR "state of the union"
   - Rob Knauerhase, Intel

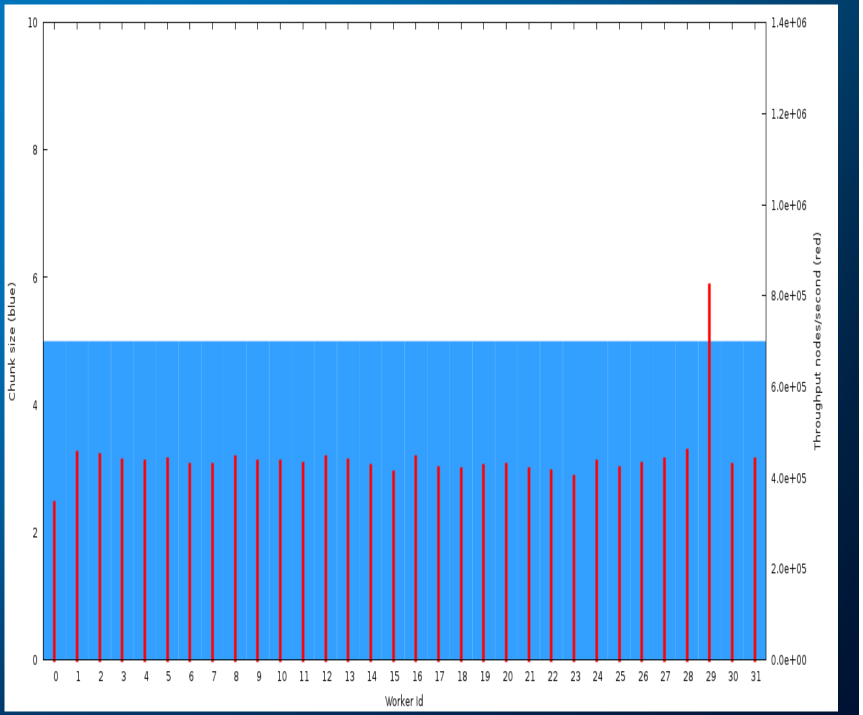
➡ Live demo of OCR v0.8 & adaptability
   - Vincent Cave, Rice

5. Next Steps
   - Rob Knauerhase
6. Closing, Q&A/discussion, wrap-up
   - Vivek Sarkar

Open Community Runtime

Demo Explanation

- OCR adapting to policy changes
  - one instance of application (one run), automatic dynamic adaptation within the run
  - adaptation is "free" from runtime, no source code changes or recompilation etc.
  - imagine a "policy engine" (ours is simple) that translates goals into runtime parameters



- What you'll see
  - blue bar == chunksize
  - red lines == per-core throughput
    - nodes processed per second

Live demonstration

Agenda

1. Introduction & Motivation
   – Vivek Sarkar, Rice U.
2. Lightning Talks
   – Roger Golliver, UIUC
   – Benoit Meister, Reservoir Labs
3. OCR "state of the union"
   – Rob Knauerhase, Intel
4. Live demo of OCR v0.8 & adaptability
   – Vincent Cave, Rice

   Next Steps
   – Rob Knauerhase

6. Closing, Q&A/discussion, wrap-up
   – Vivek Sarkar

Who Should Look Into OCR

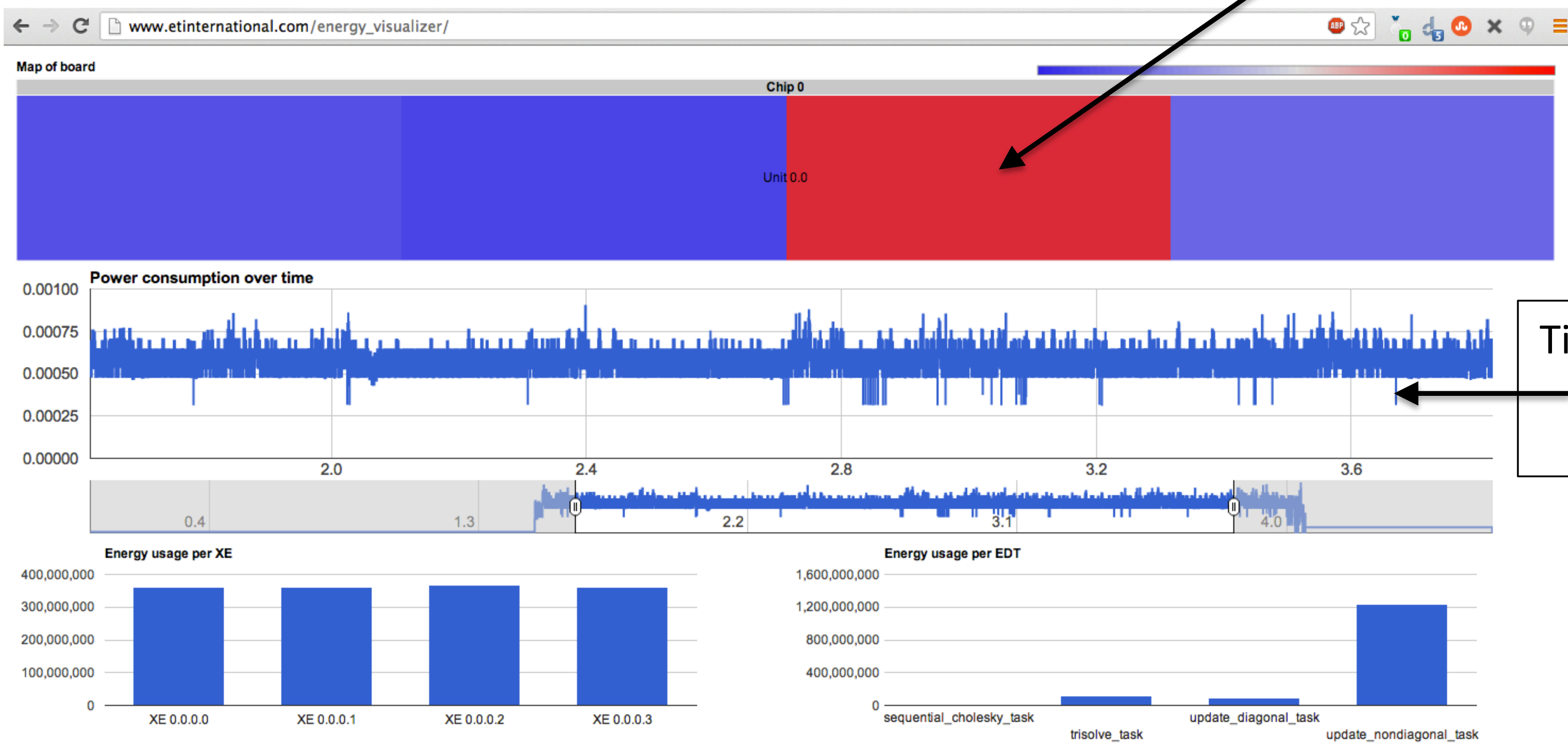
- Application researchers
  - New decomposition for extreme-scale machines, up to 10^18
- Hardware researchers
  - Exploration of chip/platform support for future workloads
  - Ideas of what PMUs will be useful for future environments
- System-software researchers
  - Compilers: how to decompose, offer hints
  - Runtimes: FGED approach, optimizations, …
  - Operating systems: interaction with runtime + storage, memory, …

ETI's involvement in OCR

- **Participating** in OCR steering committee and core team
- **Contributing** to OCR:
  - Statistics Framework
  - Energy Visualization
  - Memory Management
  - Networking
- Will supplement standard OCR support with **commercial levels of support**
  - Bug Fixes, Support, Training, Documentation, Consultation

Energy Visualization

Hierarchical system view of energy usage



Time series view of energy

Energy usage per component

Energy usage per individual EDT

OCR resources (see flyer)

- Project homepage at
  http://01.org/open-community-runtime
- Public repository on github
  http://github.com/01org/ocr
- Mailing lists
  - ocr-announce
  - ocr-devel
  - ocr-discuss
  - ocr-build
- Upcoming OCR white paper

http://01.org/open-community-runtime

Open Community Runtime

Agenda

1. Introduction & Motivation
   – Vivek Sarkar, Rice U.
2. Lightning Talks
   – Benoit Meister, Reservoir Labs
   – Roger Golliver, UIUC
3. OCR "state of the union"
   – Rob Knauerhase, Intel
4. Live demo of OCR v0.8 & adaptability
   – Vincent Cave, Rice
5. Next Steps
   – Rob Knauerhase

Closing, Q&A/discussion, wrap-up
   – Vivek Sarkar

Open Community Runtime

OCR Roadmap for 2014

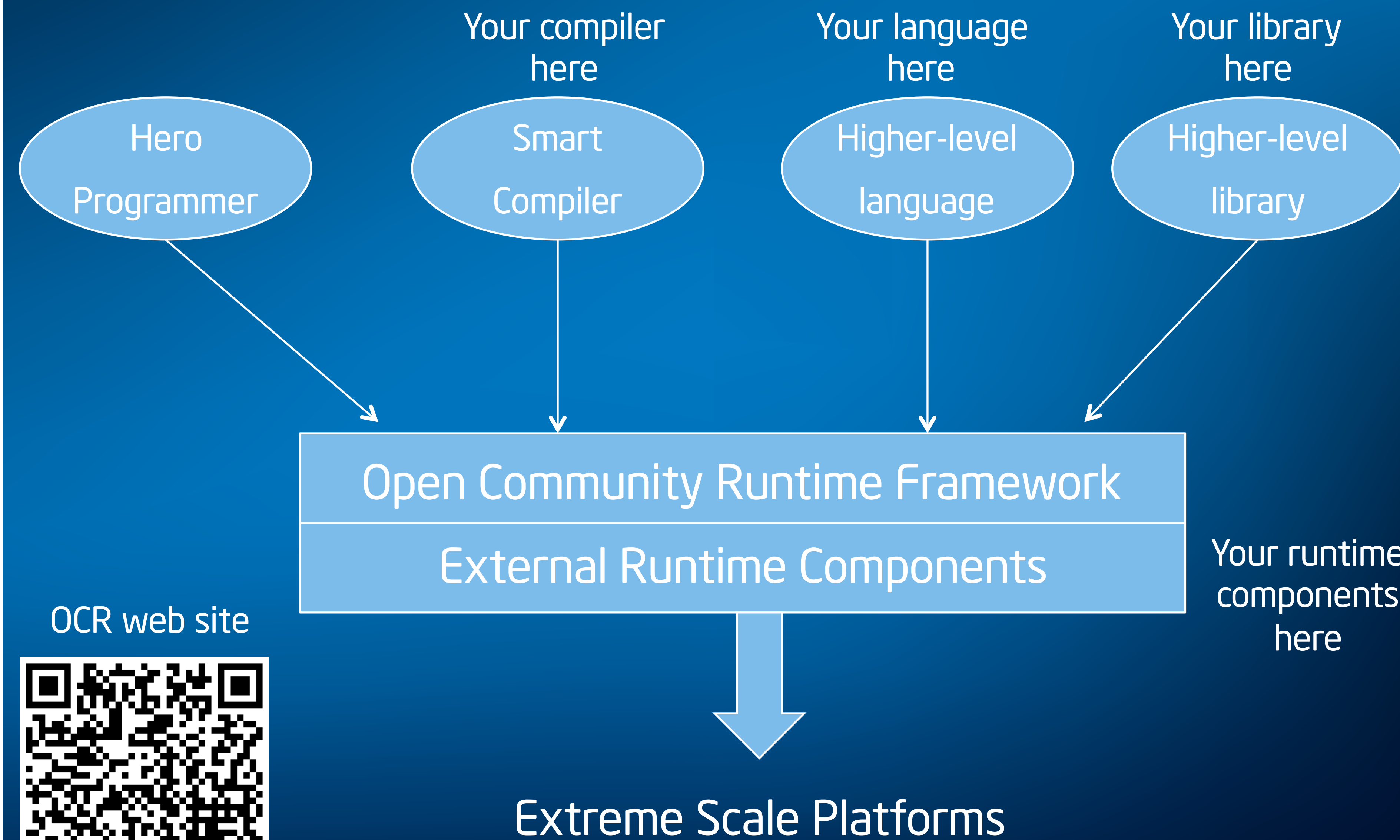
Core team

- Integration with communication runtimes (MPI, GASNet) for distributed clusters
- Extension of programmer-directed hints with automated movement of data and tasks
- Extensions for GPUs and accelerators
- Extensions to machine descriptions
- Support for new policies and tuning annotations
- Extensions to introspection and adaptation including locality-aware scheduling

Related efforts

- CnC on OCR with checkpoint-restart support
- Proxy applications on OCR using HClib, CnC, and R-stream
- Hierarchically Tiled Arrays (HTA) on OCR

OCR Vision



OCR web site

SC13 Survey URL -- http://bit.ly/sc13-eval