The Open Community Runtime (OCR) Framework for Extreme Scale Systems

Birds of a Feather Session, SC14, New Orleans
November 20, 2014

Organizers:
Vivek Sarkar (Rice U.)
Barbara Chapman (U. Houston)
William Gropp (U. Illinois)

Live poll at http://pollev/sc14

Agenda

1. Motivation and Introduction
   – William Gropp, UIUC
   – Vivek Sarkar, Rice
2. OCR specification
   – Tim Mattson, Intel
3. Use of OCR in Applications
   – David Richards, LLNL
   – Laura Carrington, SDSC
4. Demos of Distributed OCR and CnC-on-OCR on today's systems
   – Vincent Cave, Rice
   – Zoran Budimlic, Rice
5. OCR on future systems
   – Josh Fryman, Intel
6. Closing, Q&A/discussion, live poll
   – Vivek Sarkar, Rice

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 ideas and components from current runtime systems as appropriate

➔Motivation for Open Community Runtime framework that …

  – is representative of future execution models
  – can express large amounts of parallelism in a task-based model
  – can explicitly capture logical dependences and data movements
  – can be targeted by multiple high-level programming systems
  – can be mapped efficiently on to future extreme scale platforms
  – is available as an open-source testbed
  – enables us to address revolutionary challenges collaboratively

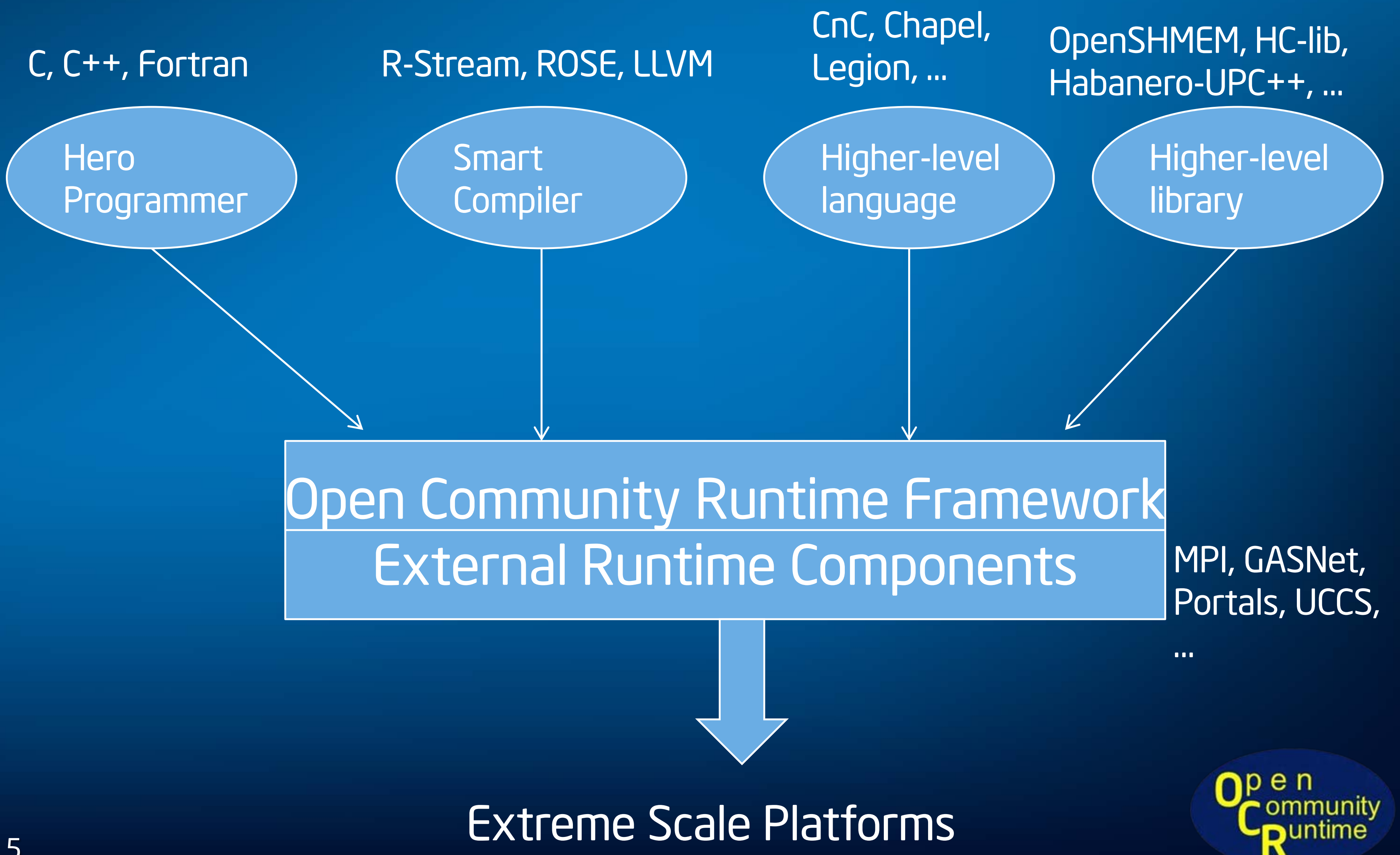
Performance Variability is on the rise

- 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 ➔ need for dynamic adaptive asynchronous runtime systems

Open Community Runtime

OCR Vision
(https://xstackwiki.modelado.org/Open_Community_Runtime)



C, C++, Fortran

Hero Programmer

R-Stream, ROSE, LLVM

Smart Compiler

CnC, Chapel, Legion, …

Higher-level language

OpenSHMEM, HC-lib, Habanero-UPC++, …

Higher-level library

Open Community Runtime Framework

External Runtime Components

MPI, GASNet, Portals, UCCS, …

Extreme Scale Platforms

OCR Acknowledgments

- Design strongly influenced by
  - Intel Runnemede project (via DARPA UHPC program)
    - power efficiency, programmability, reliability, performance
  - Codelet philosophy from CAPSL group at U. Delaware
    - implicit notions of dataflow
  - Habanero project at Rice U.
    - data-driven tasks, data-driven futures, hierarchical places in Habanero-C language
  - Concurrent Collections model – Intel Software/Solutions Group
    - decomposition of algorithm into steps/items/tags, tuning

- *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)*

Agenda

1. Motivation and Introduction
   - William Gropp, UIUC
   - Vivek Sarkar, Rice
2. OCR specification
   - Tim Mattson, Intel
3. Use of OCR in Applications
   - David Richards, LLNL
   - Laura Carrington, SDSC
4. Demos of Distributed OCR and CnC-on-OCR on today's systems
   - Vincent Cave, Rice
   - Zoran Budimlic, Rice
5. OCR on future systems
   - Josh Fryman, Intel
6. Closing, Q&A/discussion, live poll
   - Vivek Sarkar, Rice

The OCR specification

Tim Mattson

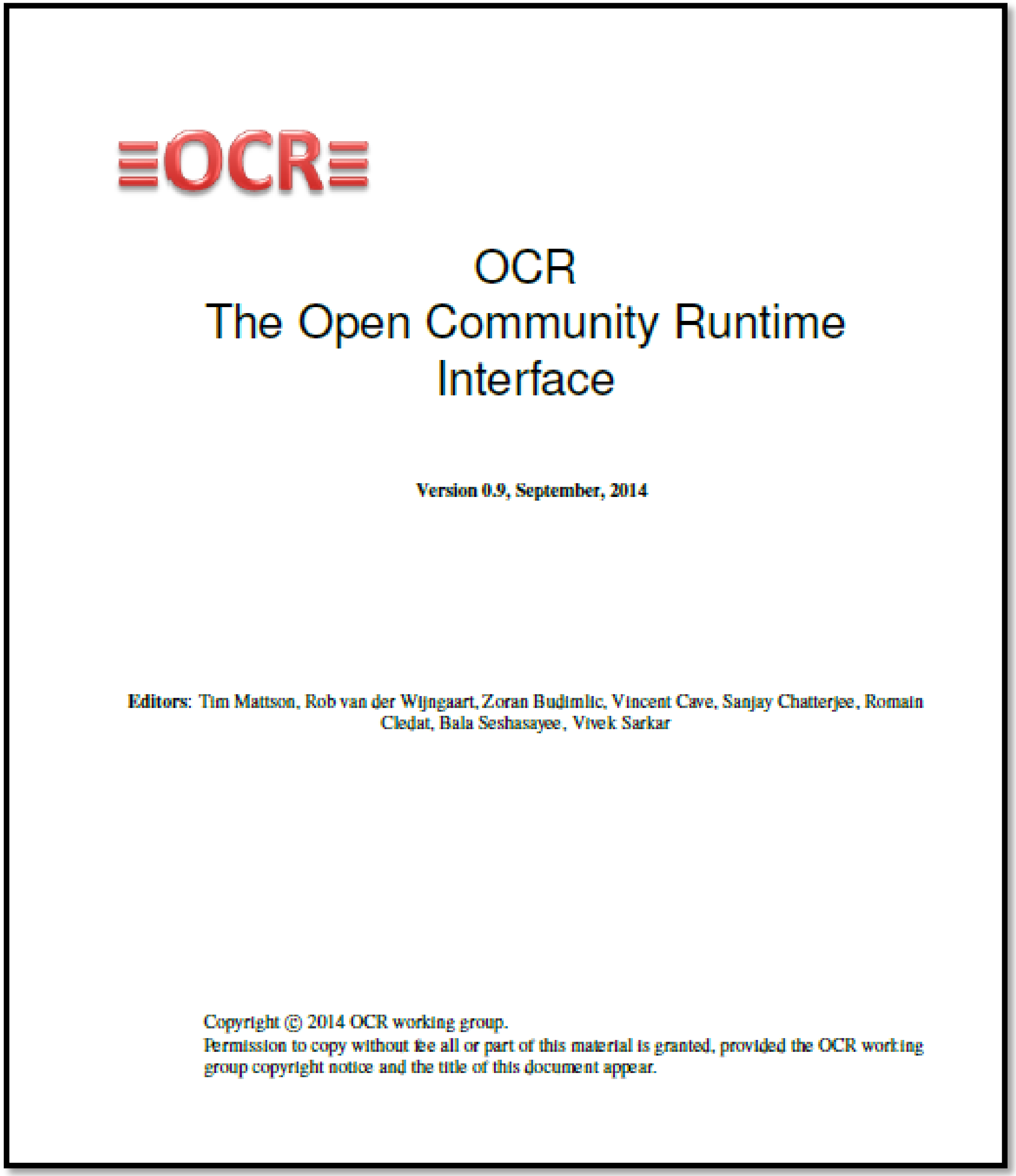
Parallel Computing Lab

timothy.g.mattson@intel.com

My journey into the world of OCR

- I started working with OCR as part of the Advanced computing runtime project about one year ago.
- When I asked for documentation I was sent to doxygen generated text.
  - This described the interfaces to the functions but said nothing about what they meant.
  - There were few (if any) documented examples to help me learn OCR.
  - To learn the jargon of OCR you had to “sit at the feet of the masters” who came before you and learn from them.
  - Evolution of OCR happened “in the code” … which is not a good idea if you want a coherent “big picture” as OCR grows.

If OCR was to be the foundation for our research, it needed a “formal” specification. One didn’t exist ... so I led the effort to write one!



OCR

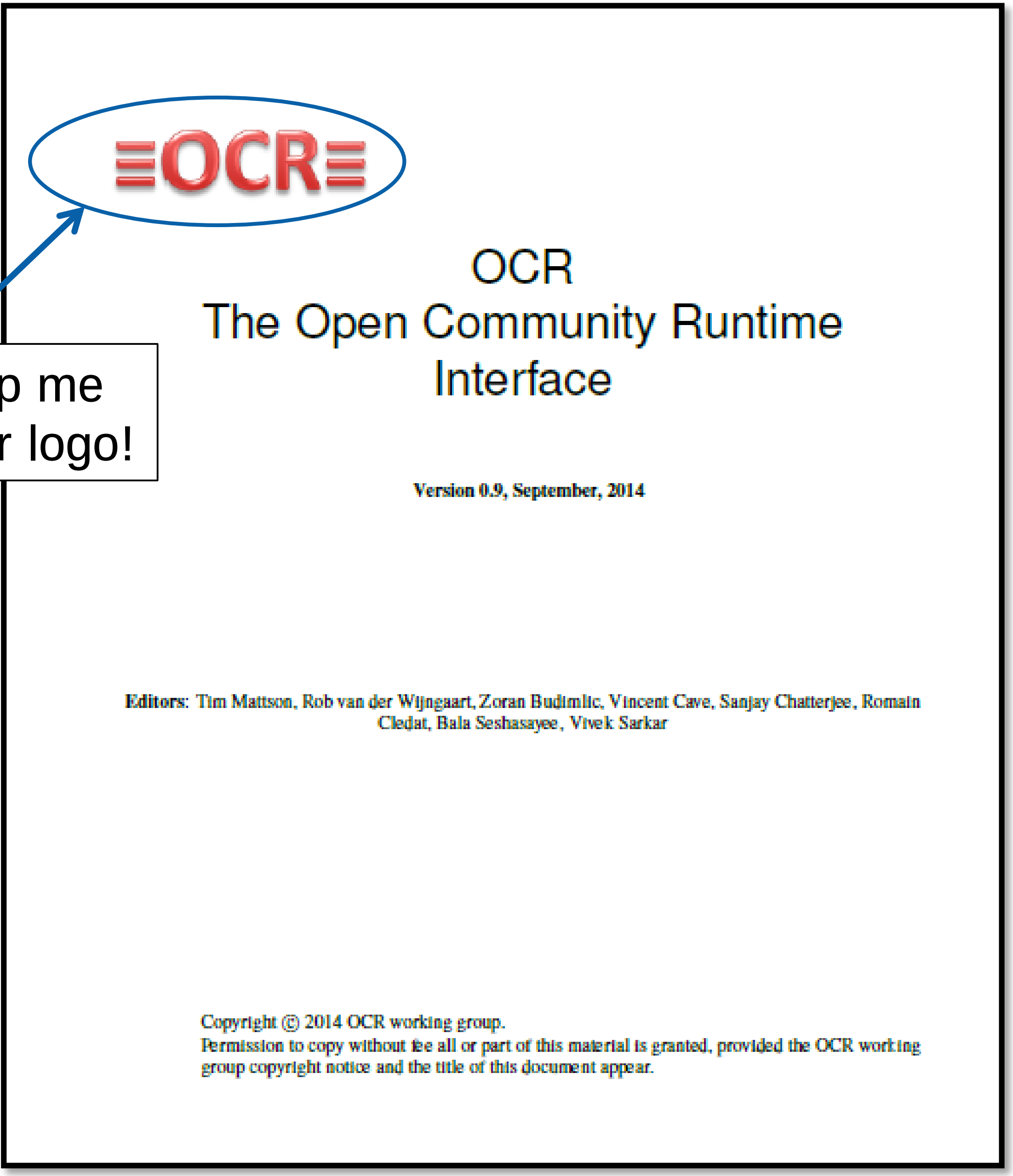
OCR
The Open Community Runtime Interface

Version 0.9, September, 2014

Editors: Tim Mattson, Rob van der Wijngaart, Zoran Budimlic, Vincent Cave, Sanjay Chatterjee, Romain Cledat, Bala Seshasayee, Vivek Sarkar

https://xstackwiki.modelado.org/Open_Community_Runtime

Please help me find a better logo!



OCR
The Open Community Runtime Interface

Version 0.9, September, 2014

Editors: Tim Mattson, Rob van der Wijngaart, Zoran Budimlic, Vincent Cave, Sanjay Chatterjee, Romain Cledat, Bala Seshasayee, Vivek Sarkar

https://xstackwiki.modelado.org/Open_Community_Runtime

The OCR specification: the core spec

- **1 Introduction**
  - – 1.1 Scope
  - – 1.2 Glossary
  - – 1.3 Execution Model
  - – 1.3.1 OCR Platform
  - – 1.3.2 OCR objects
  - – 1.3.3 Trigger rule
  - – 1.3.4 OCR program execution
  - – 1.4 Memory Model
  - – 1.5 Organization of this document
- **2 OCR API Documentation**

The OCR specification: Examples Appendix

- **A OCR Examples**
- A.1 OCR’s “Hello World
- A.2 Expressing a Fork-Join pattern
- A.3 Expressing unstructured parallelism
- A.4 Using a Finish EDT
- A.5 Accessing a DataBlock with “Intent-To-Write” Mode
- A.6 Accessing a DataBlock with “Exclusive-Write” Mode
- A.7 Acquiring contents of a DataBlock as a dependence input

Conclusion

- The fact we have a formal specification for OCR is a REALLY BIG DEAL!!!!
- Download the specification and read it.

  https://xstackwiki.modelado.org/Open_Community_Runtime

- Help us make OCR (and the specification) better. Contact me directly if you have feedback or suggestions for advancing OCR as defined by the specification.

timothy.g.mattson@intel.com

Agenda

1. Motivation and Introduction
   - William Gropp, UIUC
   - Vivek Sarkar, Rice
2. OCR specification
   - Tim Mattson, Intel
3. Use of OCR in Applications
   - David Richards, LLNL
   - Laura Carrington, SDSC
4. Demos of Distributed OCR and CnC-on-OCR on today's systems
   - Vincent Cave, Rice
   - Zoran Budimlic, Rice
5. OCR on future systems
   - Josh Fryman, Intel
6. Closing, Q&A/discussion, live poll
   - Vivek Sarkar, Rice

CoMD in OCR:
How a lazy programmer can use many tasks

OCR Birds-of-a-Feather

November 20, 2014

David Richards

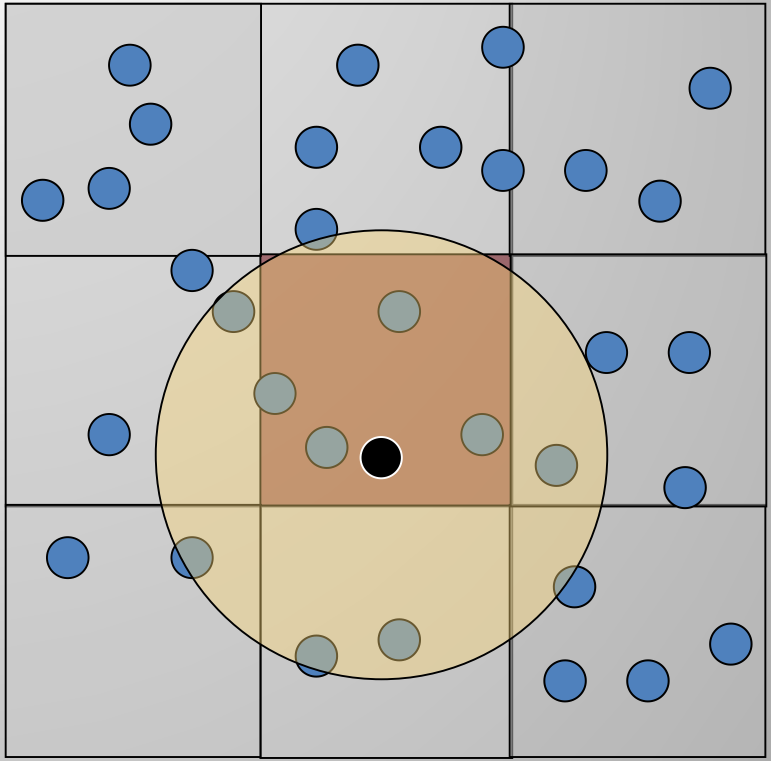
Lawrence Livermore National Laboratory

LLNL-PRES-XXXXXX

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

ExMatEx 2014 Summer School

9 Students and a Whiteboard

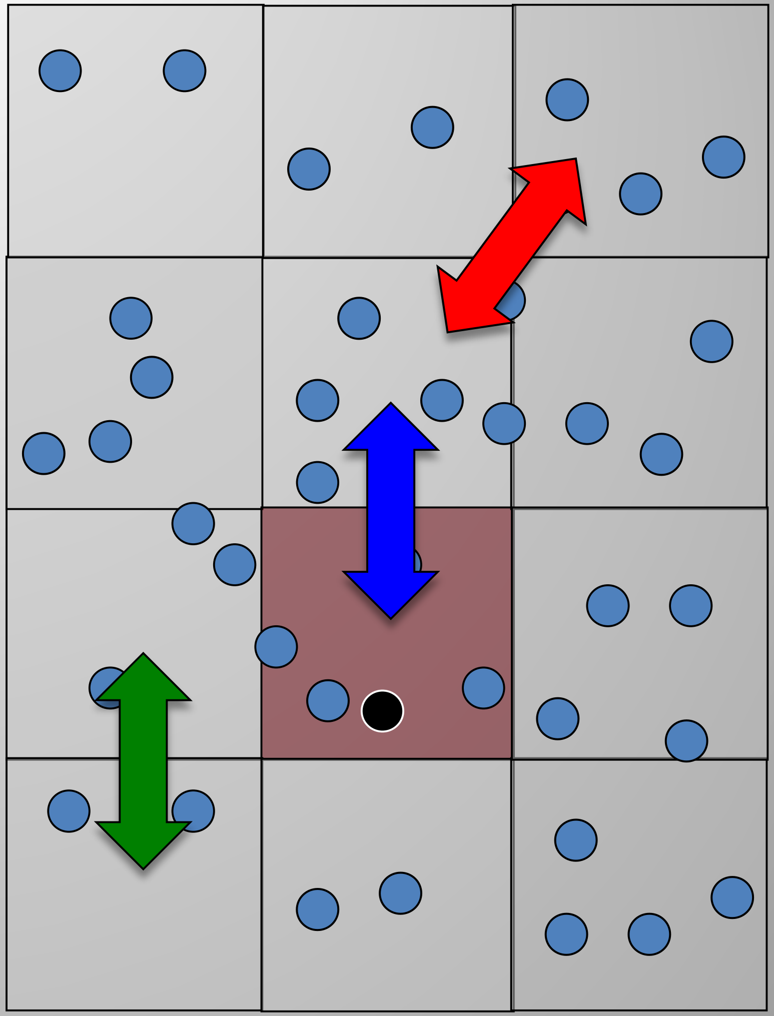


CoMD Proxy App for Molecular Dynamics

OCR could guarantee correctness

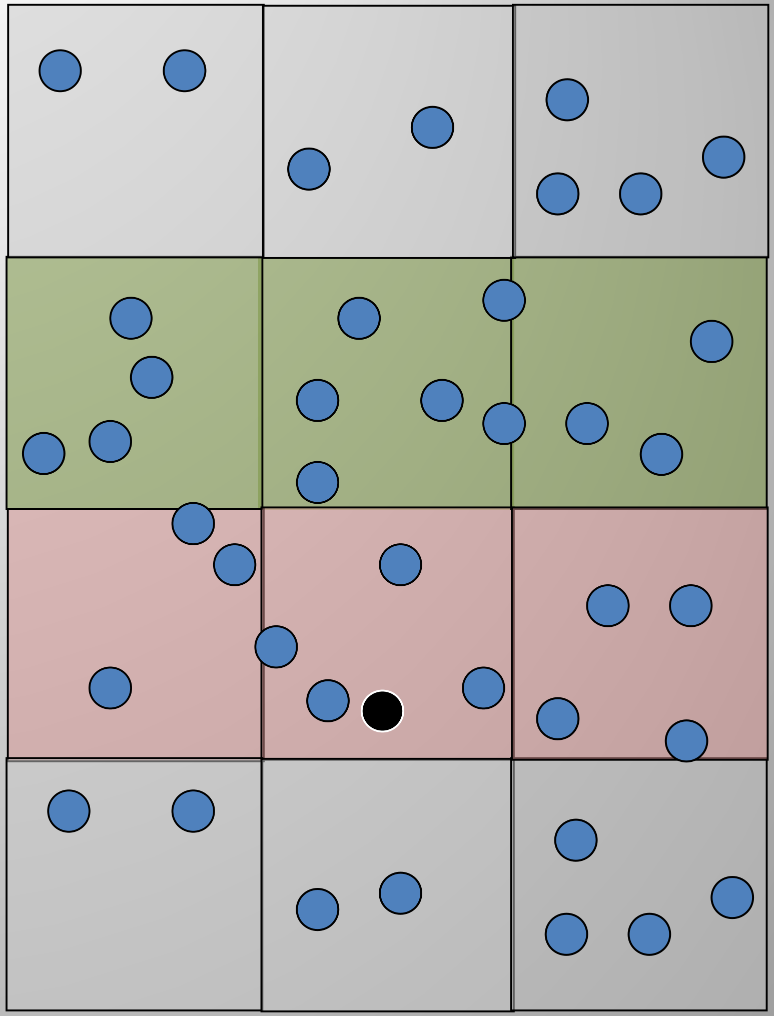
MD Force loop (simplified)

```
forall iBox in boxes
   forall jBox in nbrs(iBox)
      forall iAtoms in iBox
         forall jAtoms in jBox
            fij = f(ri, rj)
            Forcei += fij
            Forcej -= fij
```



Blue and Green can be concurrent,
Blue and Red cannot (race condition)

OCR could optimize performance

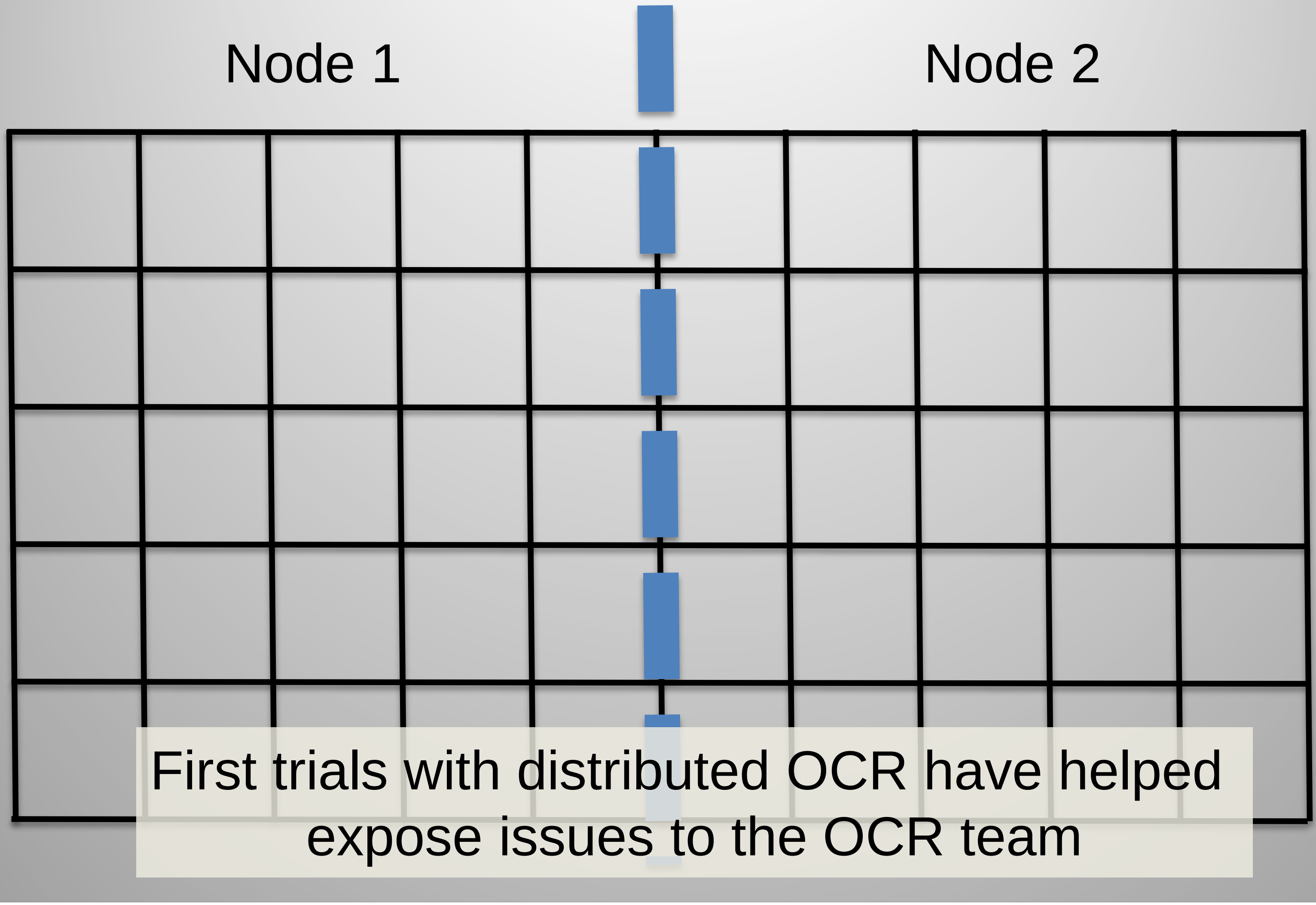


MD Force loop (simplified)

```
forall iBox in boxes
   forall jBox in nbrs(iBox)
      forall iAtoms in iBox
         forall jAtoms in jBox
            fij = f(ri, rj)
            Forcei += fij
            Forcej -= fij
```

Execute tasks that utilize data already in cache

OCR automates domain decomposition and data motion



First trials with distributed OCR have helped expose issues to the OCR team

Students and Models

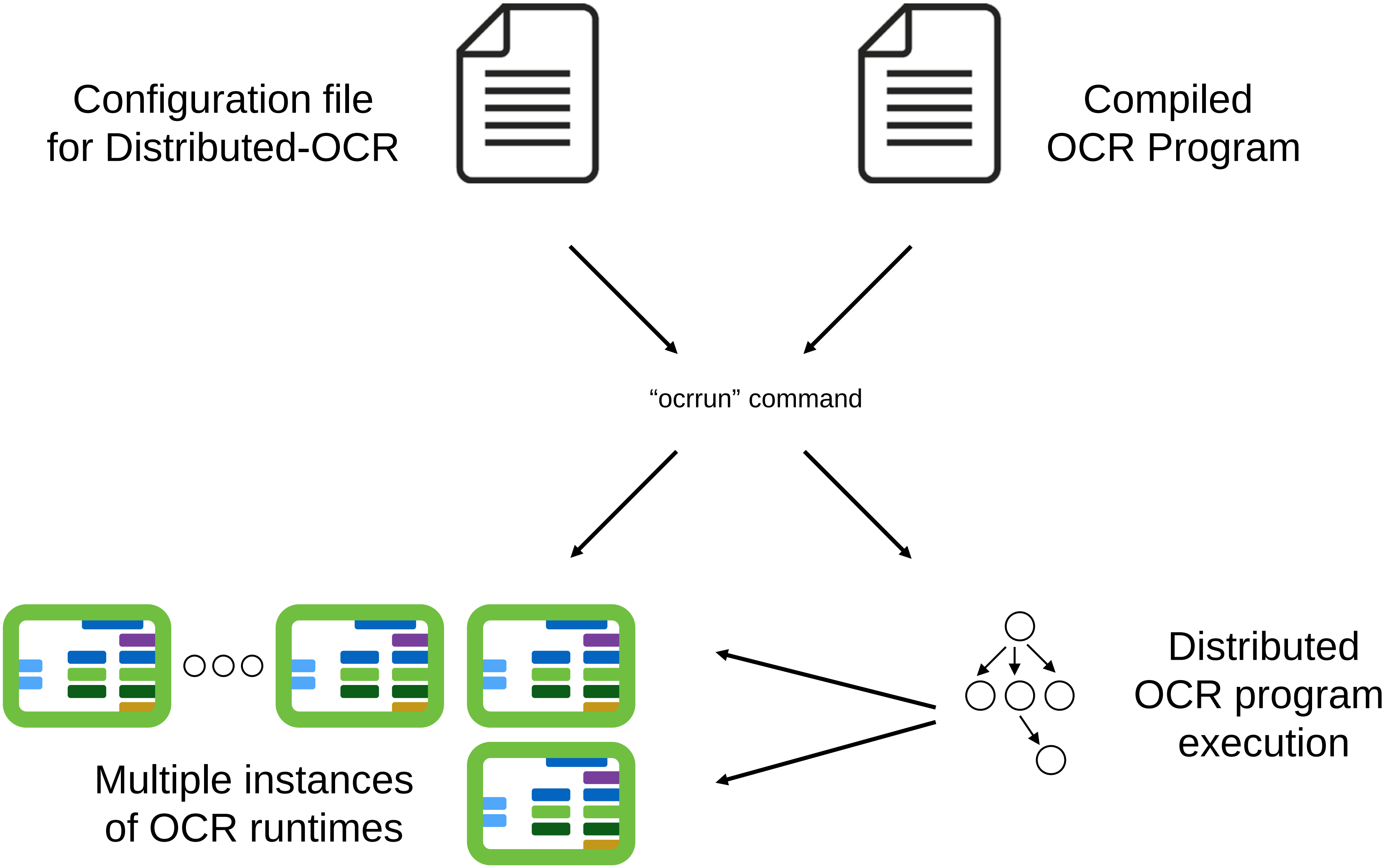
- Sam Reeve (Purdue) LAMMPS, CoMD, leanMD
- Riyaz Haque (UCLA), CNC (Habanero-C)
- Luc Jualmes (Barcelona) OmpSs, Chapel
- Sameer Abu Asal (LSU) C++11 futures, HPX
- Aaron Landmehr (Delaware) OCR
- Sanian Gaffer (NM-State) UPC++
- Gheorghe-Teodor Bercea (Imperial) PyOP2
- Zach Rubinstein (Chicago), Troels Henriksen (Copenhagen) Embedded DSL

Agenda

1. Motivation and Introduction
   - William Gropp, UIUC
   - Vivek Sarkar, Rice
2. OCR specification
   - Tim Mattson, Intel
3. Use of OCR in Applications
   - David Richards, LLNL
   - Laura Carrington, SDSC
4. Demos of Distributed OCR and CnC-on-OCR on today's systems
   - Vincent Cave, Rice
   - Zoran Budimlic, Rice
5. OCR on future systems
   - Josh Fryman, Intel
6. Closing, Q&A/discussion, live poll
   - Vivek Sarkar, Rice

Open Community Runtime

Distributed-OCR



Configuration file for Distributed-OCR

Compiled OCR Program

“ocrrun” command

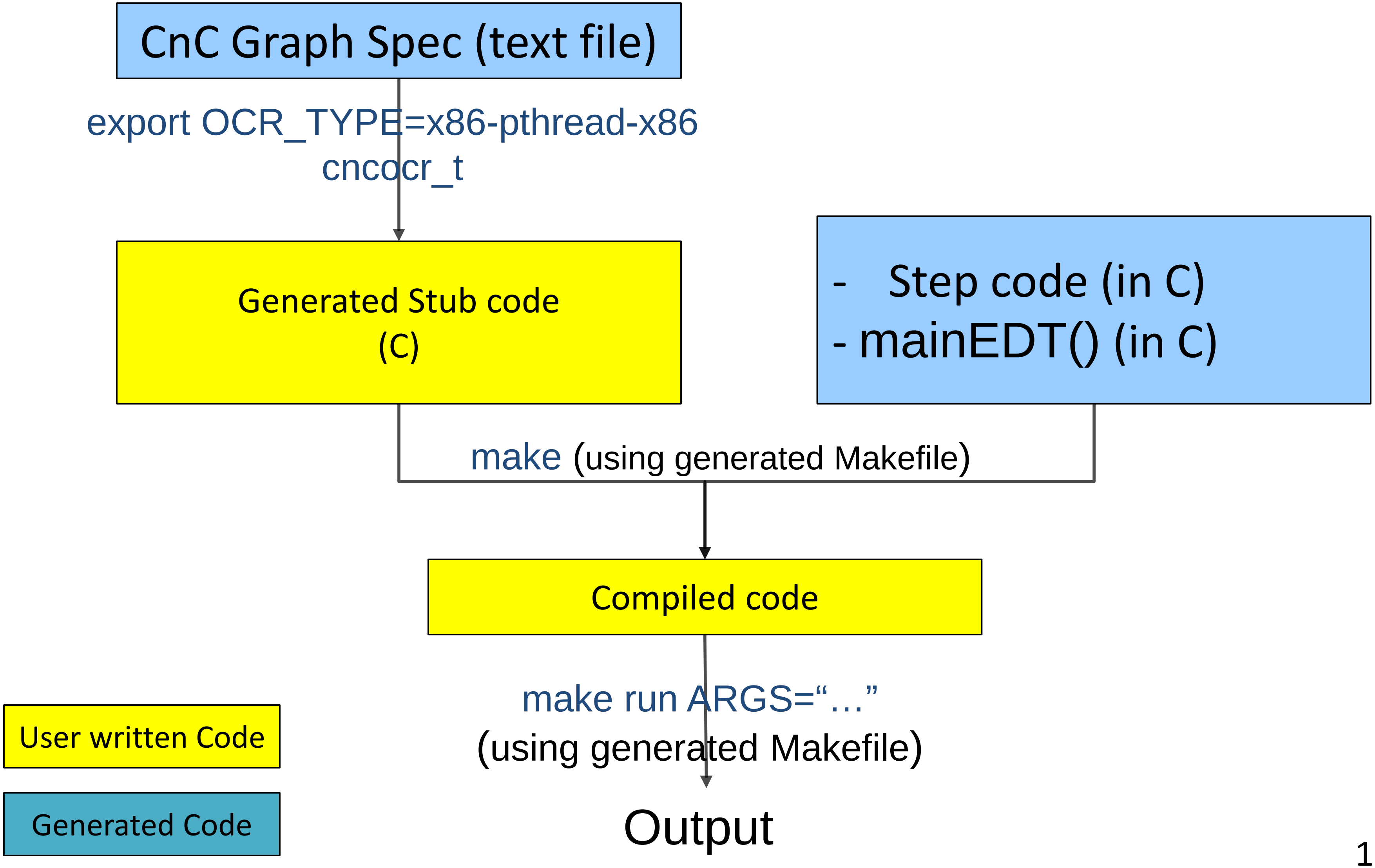
Multiple instances of OCR runtimes

Distributed OCR program execution

Obtaining OCR and CnC-OCR

- git clone -b sc14 --depth 1 https://github.com/01org/ocr.git
- cd hll
- git clone -b sc14 --depth 1 https://github.com/habanero-rice/cnc-ocr.git
- cd cnc_ocr
- source setup_env.sh

CnC-OCR Shared-Memory Build Model



CnC Graph Spec (text file)

export OCR_TYPE=x86-pthread-x86
cncocr_t

Generated Stub code
(C)

- Step code (in C)
- mainEDT() (in C)

make (using generated Makefile)

Compiled code

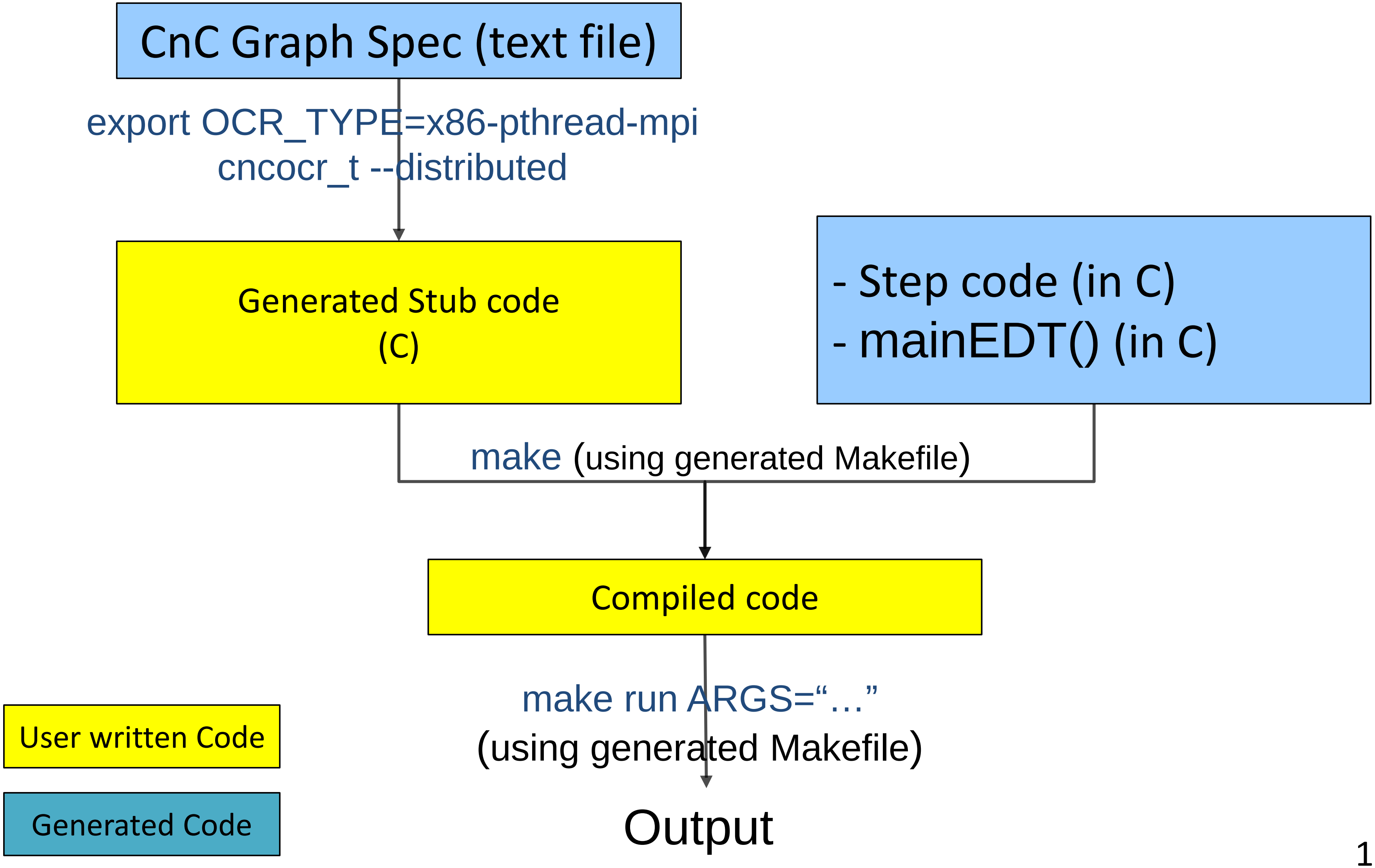
make run ARGS="…"
(using generated Makefile)

Output

User written Code

Generated Code

CnC-OCR Distributed-Memory Build Model



CnC Graph Spec (text file)

```
export OCR_TYPE=x86-pthread-mpi
cncocr_t --distributed
```

Generated Stub code (C)

- Step code (in C)
- mainEDT() (in C)

make (using generated Makefile)

Compiled code

make run ARGS="…" (using generated Makefile)

Output

User written Code

Generated Code

Agenda

1. Motivation and Introduction
   - William Gropp, UIUC
   - Vivek Sarkar, Rice
2. OCR specification
   - Tim Mattson, Intel
3. Use of OCR in Applications
   - David Richards, LLNL
   - Laura Carrington, SDSC
4. Demos of Distributed OCR and CnC-on-OCR on today’s systems
   - Vincent Cave, Rice
   - Zoran Budimlic, Rice
5. OCR on future systems
   - Josh Fryman, Intel
6. Closing, Q&A/discussion, live poll
   - Vivek Sarkar, Rice

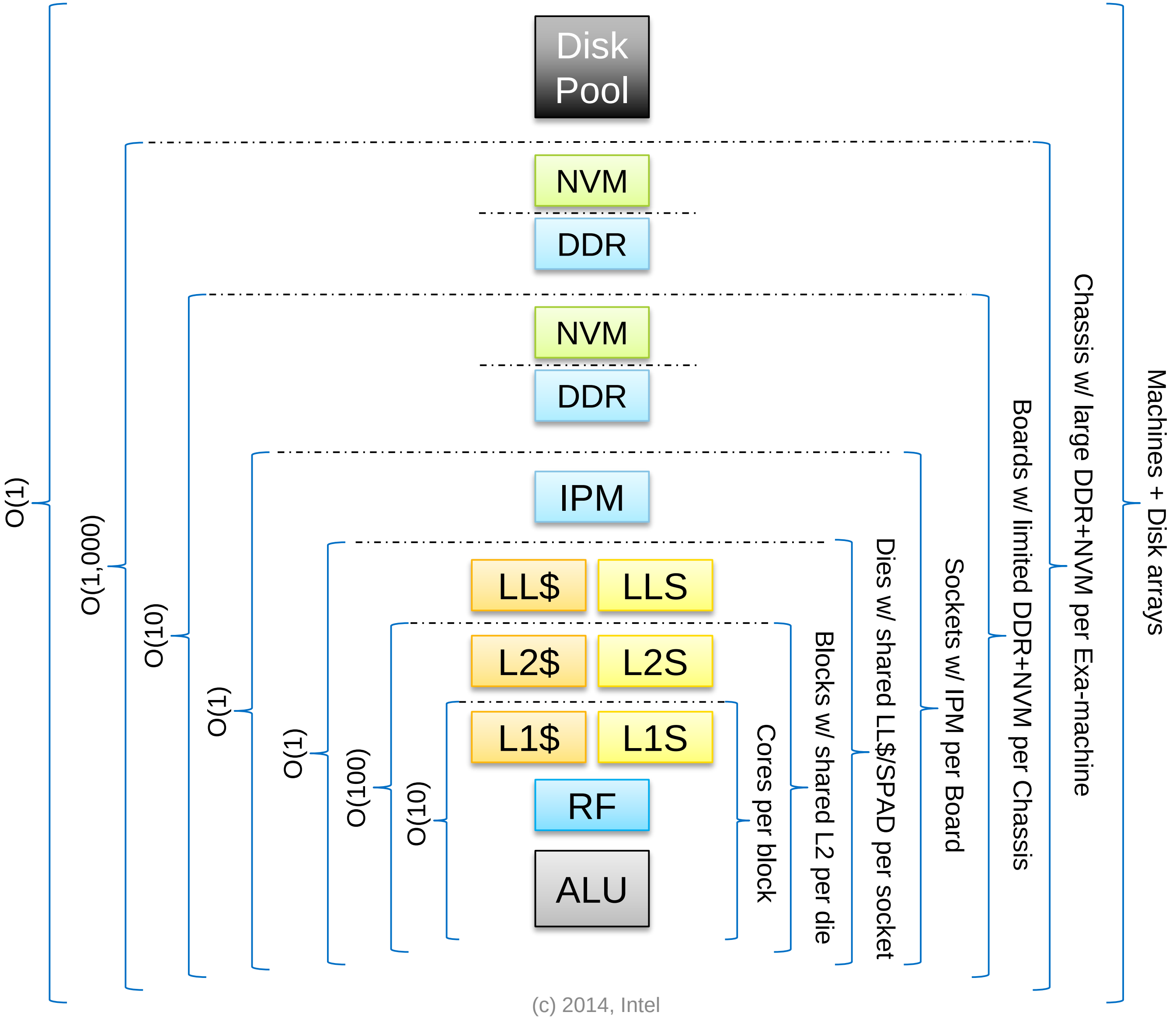
OCR and the “TG” Architecture: FSim and Future Architectures

Josh Fryman, System Architect

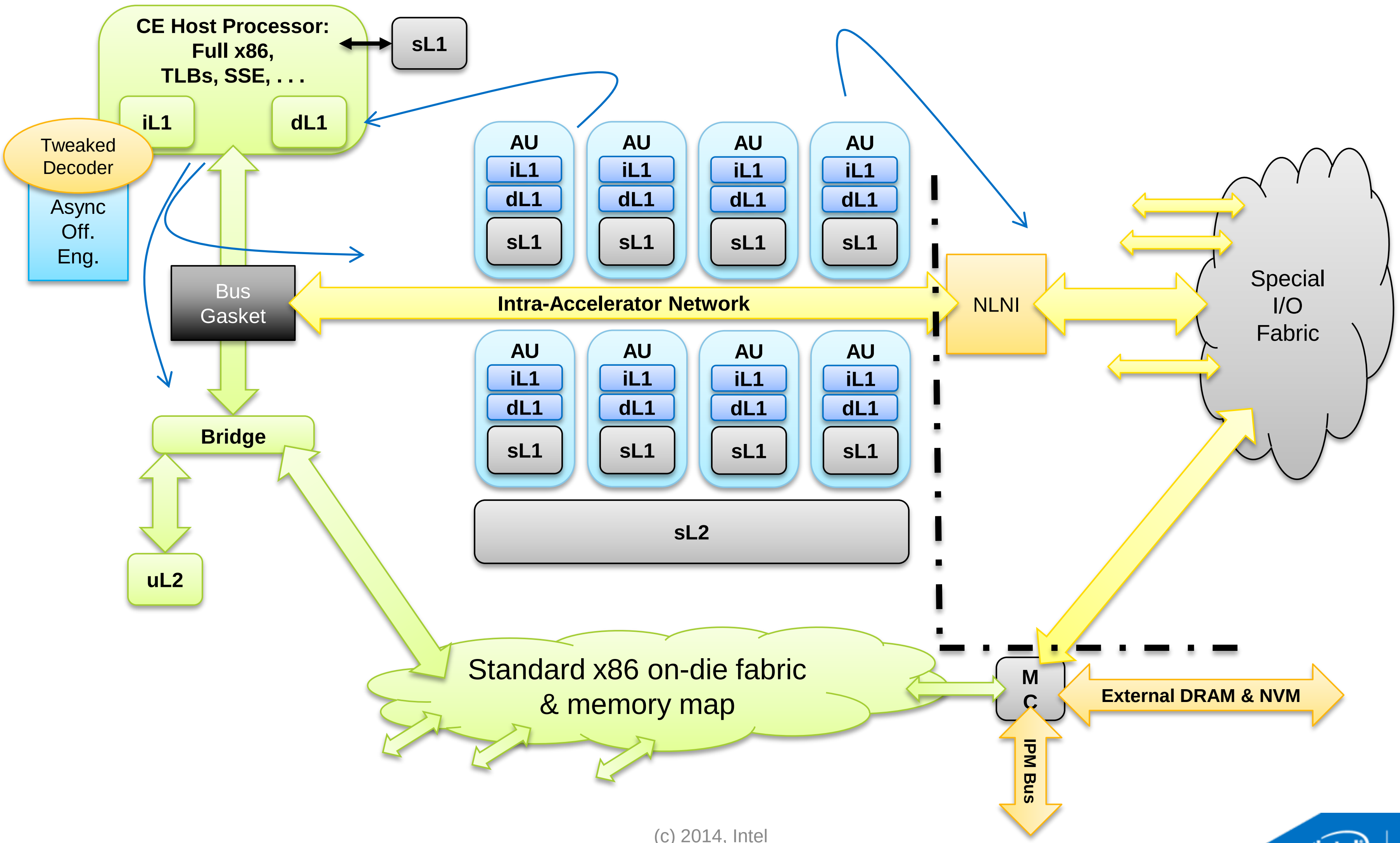
Extreme Scale Architecture Pathfinding

Datacenter and Communications Group

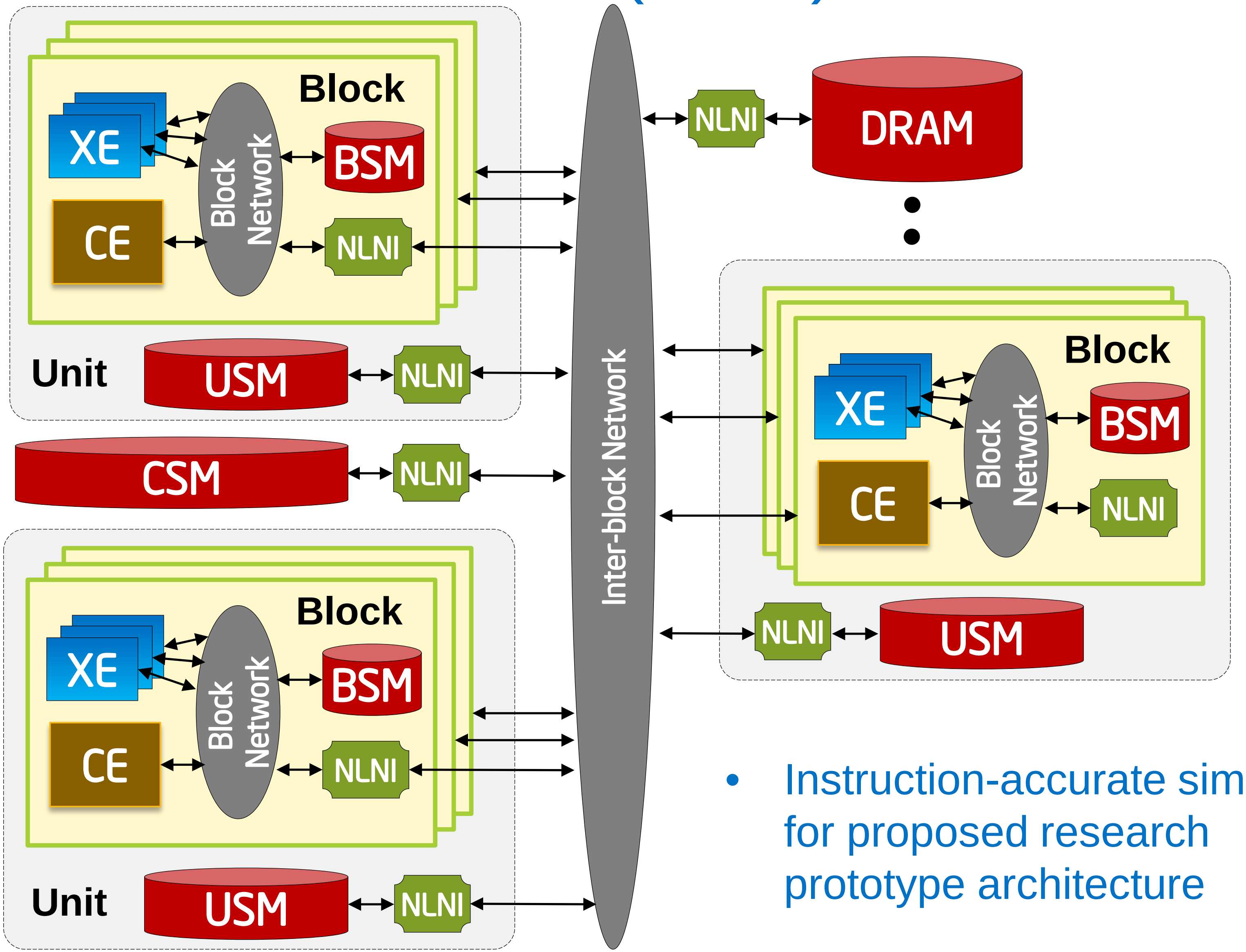
10+ Levels of Memory, O(100M) Cores



“Sea of Blocks” Compute Model



Functional Simulator (FSIM)



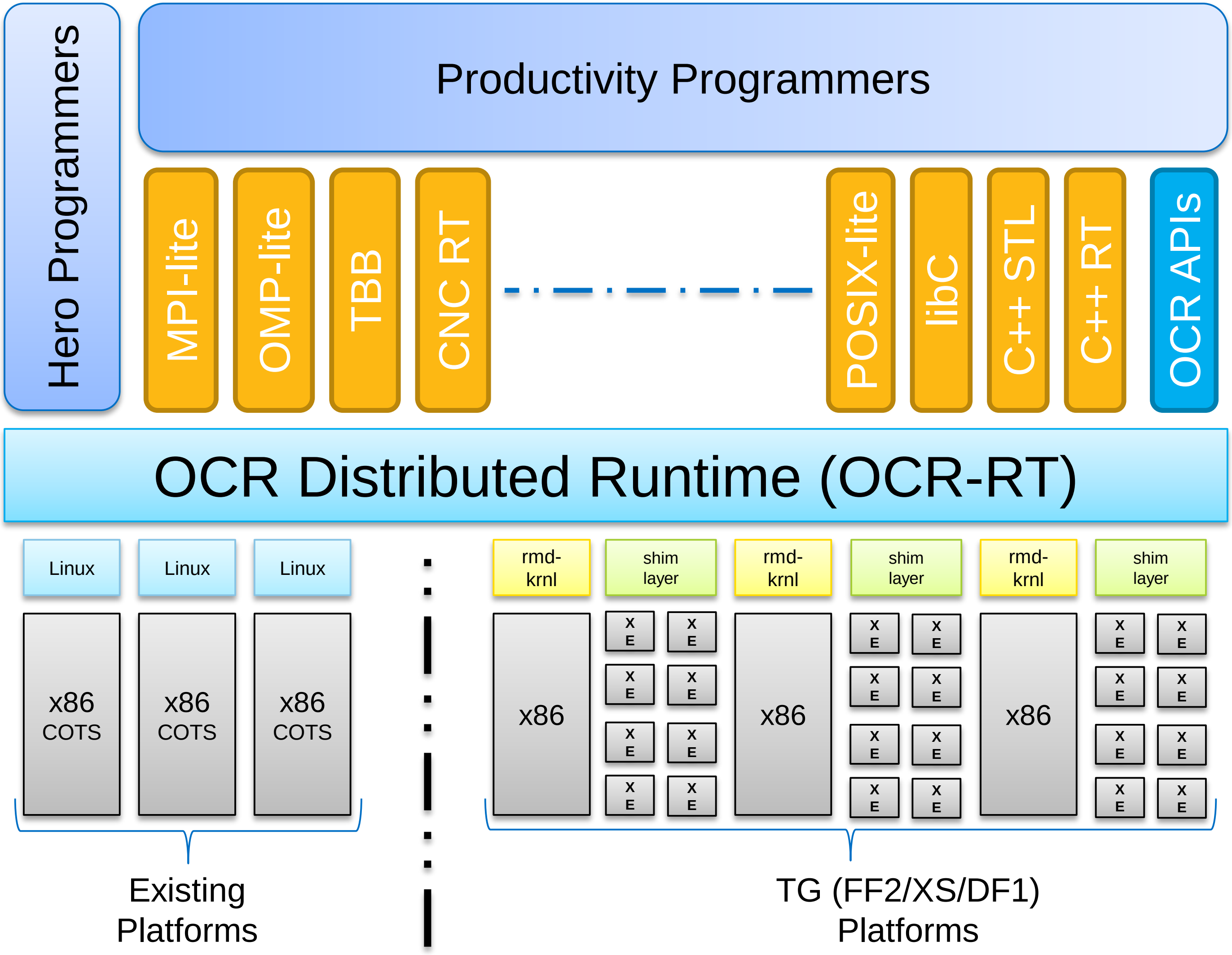
- Instruction-accurate simulator for proposed research prototype architecture

(c) 2014, Intel

FSim and OCR

- OCR is ported to and runs inside FSim, along with ELF apps
- Used in various forms for internal research efforts
- Limited in simulation scale by cluster to run against
- Some clocking, power models in design & automated tools
- Will be open-source released by Aug 2015
  - Will not include active end-user support
  - Multiple academic and industry partners are familiar with internals
- FSim has no x86 IP issues
- Can be used according to OSS license for other projects
  - Specific license TBD

SW Ecosystem Vision for OCR Systems



(c) 2014, Intel

Agenda

1. Motivation and Introduction
   - William Gropp, UIUC
   - Vivek Sarkar, Rice
2. OCR specification
   - Tim Mattson, Intel
3. Use of OCR in Applications
   - David Richards, LLNL
   - Laura Carrington, SDSC
4. Demos of Distributed OCR and CnC-on-OCR on today's systems
   - Vincent Cave, Rice
   - Zoran Budimlic, Rice
5. OCR on future systems
   - Josh Fryman, Intel
6. Closing, Q&A/discussion, live poll
   - Vivek Sarkar, Rice

Who Should Look Into OCR

- Application researchers
  - Those who would like to explore new ways of expressing intrinsic parallelism in exascale applications
  - Early adopters who would like to provide feedback on our execution model and API
- Programming model researchers
  - Higher-level language/model/DSL implementers who are interested in determining if their abstractions can be mapped onto OCR
- System-software researchers
  - Compilers: how to decompose, offer hints
  - Runtimes: scalable dynamic scheduling, optimizations, …
  - Operating systems: interaction with runtime + storage, memory, …
- Hardware researchers
  - Those who would like to experiment with OCR as a proxy for an exascale runtime

BACKUP SLIDES START HERE

OCR Building Blocks

- Event-driven tasks (EDTs)
  - fine-grained uninterruptible unit of computation with well-defined inputs (events) and bounded memory accesses
  - pointers cannot be reused across task boundaries
  - support for dynamic mapping --- no hard assumptions can be made about where task will be executed (though tuning hints can be provided)
- Events (Dependences)
  - specified explicitly as preconditions on which EDTs are initiated
  - several types of dependences
- Memory datablocks
  - explicit datablock allocation is a replacement for malloc()
  - relocatable by runtime for power, resilience, …
  - allows exploitation (or modeling) of NUMA, scratchpad memories, etc.
- Implemented as C APIs e.g.,
  - u8 ocrEdtCreate(ocrGuid_t * guid, ocrGuid_t templateGuid, u32 paramc, u64* paramv, u32 depc, ocrGuid_t *depv, u16 properties, ocrGuid_t affinity, ocrGuid_t *outputEvent);