UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

Applications Experience with OCR

XStack PI Meeting

May 28, 2014

Roger Golliver

Legal

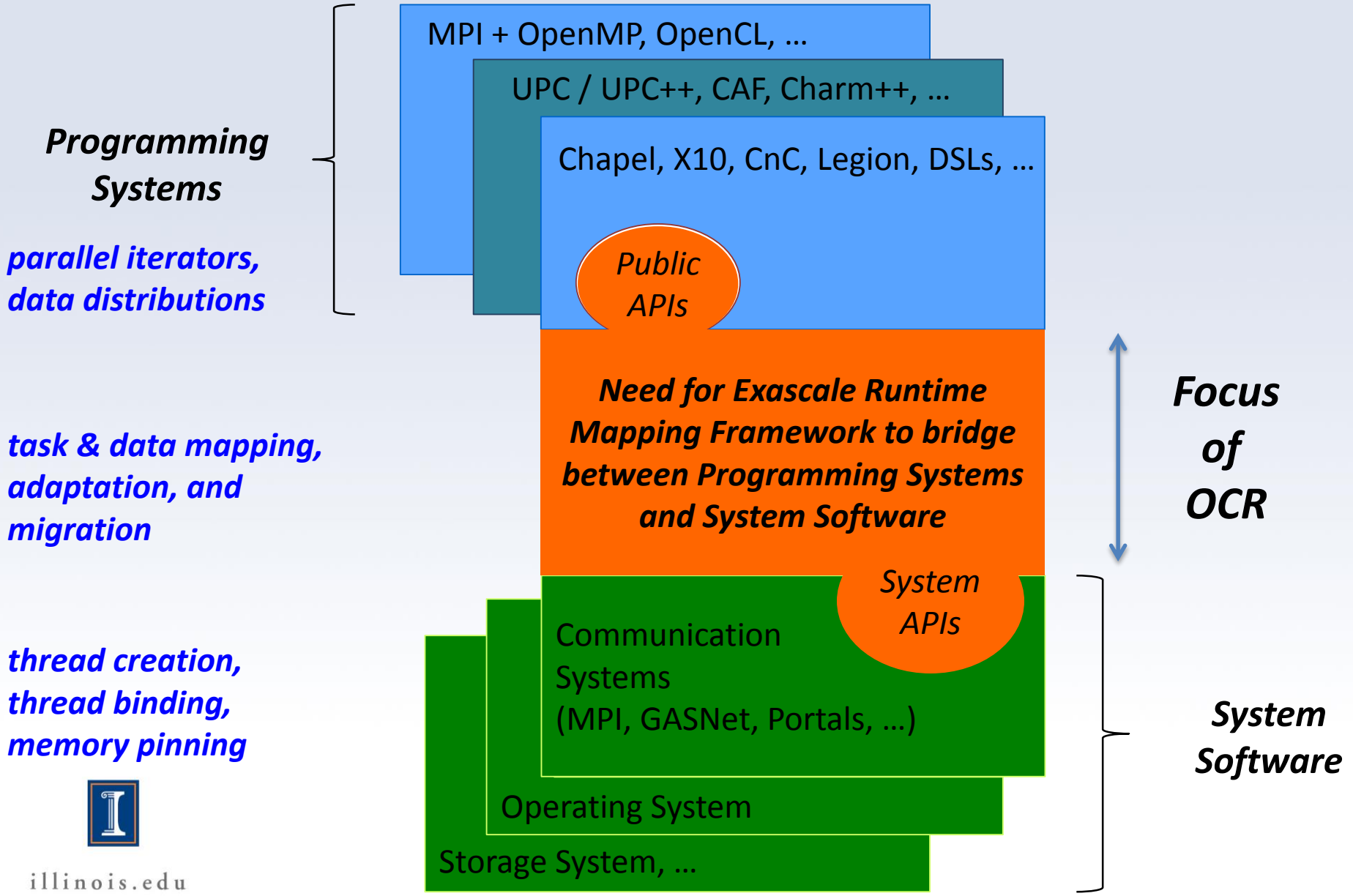
- Acknowledgment: This material is based upon work supported by the Department of Energy [Office of Science] under Award Number DE-SC0008717.

-

Overview

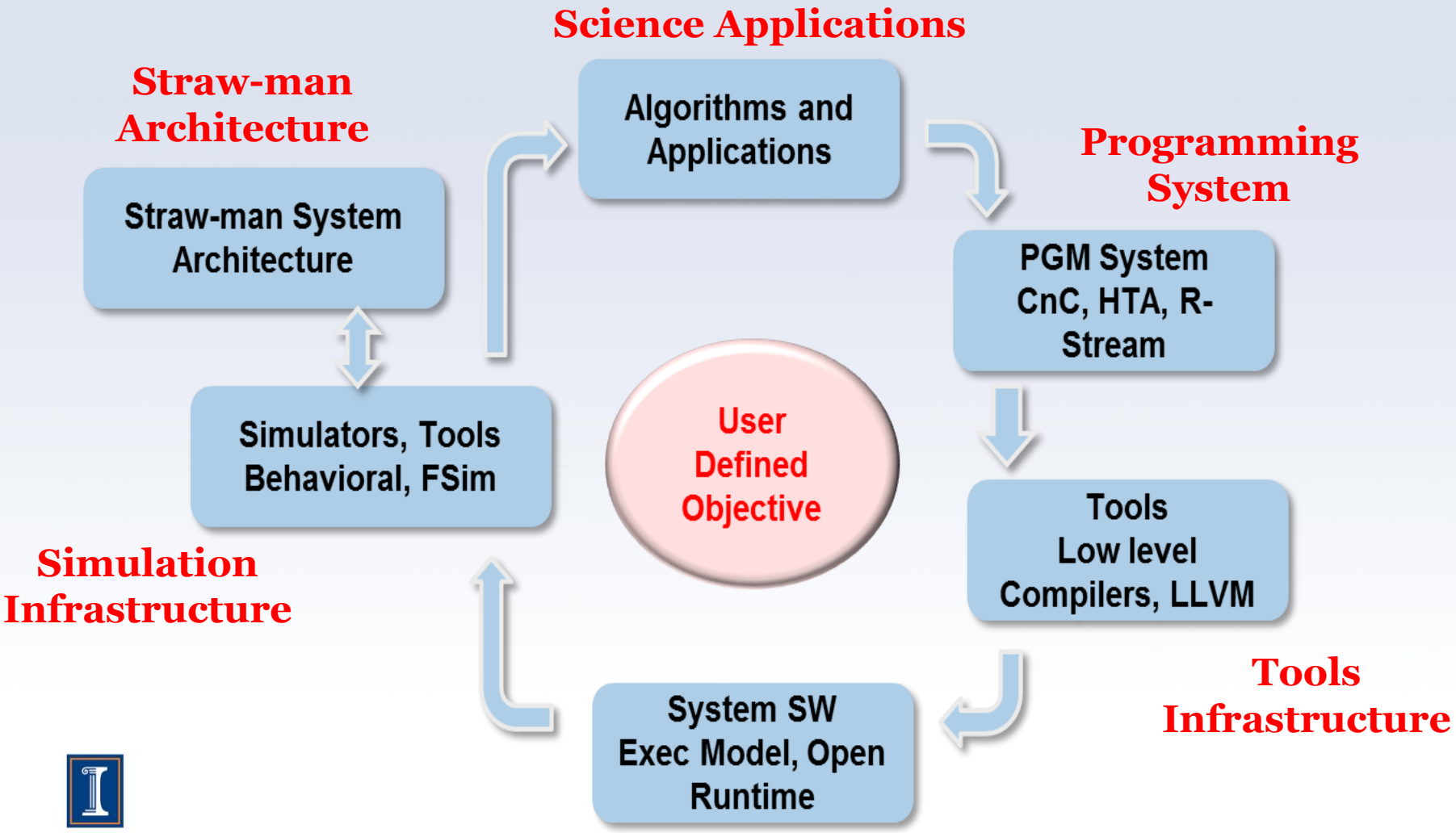
- Motivation for Open Community Runtime (OCR)
- Strategy for translation to OCR
- Partial list of OCR Applications
- Experience with Lulesh
- Detailed Steps
- Development of Translation macros
- Current Status
- Next Steps

OCR Motivation



illinois.edu

OCR Context in Traleika Glacier X-Stack project



illinois.edu

Exascale Execution Model is embodied in OCR

OCR Building Blocks

- Event-driven tasks (EDTs)
  - expresses task-level parallelism in which tasks may themselves contain data/SIMT parallelism
  - each EDT has a Globally Unique ID (GUID)
  - example OCR API:

    ```
    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);
    ```
- Events (Dependences)
  - specified explicitly as conditions that gate EDT enablement
  - several types of dependences (control, data, resource, …)
  - all events are also identified uniquely with GUIDs
- Memory Datablocks
  - support for distributed global name space
  - each datablock has a unique GUID
  - interior pointers can only be reused within an EDT, not across EDTs
  - datablocks are relocatable by runtime for power, reliability, …
  - allows exploitation of non-uniform memories in storage hierarchy

Current approaches to using OCR

- Direct use of OCR API subset
  - Works for both FSIM and real hardware
  - Full OCR API only supported on real hardware and is exploited by tool chains listed below
- Habanero-C library (HClib)
- Habanero-C++ library
- CnC on OCR
- Hierarchically Tiled Arrays (HTA) on OCR
- Compiler generation of OCR calls (R-Stream)
- Habanero-C language on OCR (in progress)

Partial list of OCR Applications

- SCF from NWCHEM
  - Jaime Arteaga (Univ. of Delaware)
  - Identified need for improved transcendental functions on FSIM
- CoMD
  - Peitro Cicotti (SDSC)
  - Worked with ExMatEx scientist to understand algorithmic options
- HPCC and HPCG kernels
  - Matthew Unrath, Kyung Hwan Pak, Grady Ellison (Oregon State Univ.)
  - Use of HTA-style data management at each level of memory hierarchy
- NAS Parallel Benchmarks, BSF, SSSP
  - Adam Smith (UIUC)
  - Using HTA/PIL to map to OCR
- Lulesh multiple versions
  - OCR - Roger Golliver (UIUC)
  - HCLib/HC++lib - Vivek Kumar (Rice)
  - CnC - Kath Knobe (Intel) and Ellen Porter (PNNL) John Feo (PNNL) and Rishi Khan

Lulesh 1.0.1 Benchmark

- Started with the C++/OpenMP version
  - Collected other versions to experiment with and look for where parallelism was previously exploited
- Good proxy app in my opinion
  - 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 and no additional packages
- NOTE: C++ version of LULESH is supported by Habanero-C++ library for real hardware, but not for FSIM

Strategy for translation to OCR

- Transition to C (Needed to run on FSIM)
  - Methods to functions
  - Data Classes to structures
  - Overloaded functions to multiple versions
- Transition array and structs to datablocks.
  - malloc to ocrDBCreate
- Transition high level function flow to EDTs.
  - Function Signature to ocrEdt_t
  - IN scalars and structs passed via paramv[]
  - INOUT and OUT scalars, structs and arrays passed as datablocks via depv[]
- Transition functions call/return organization to dynamically created and scheduled EDTs
- Transition OMP loop level parallelism to using Finish EDTs

C++ to C Overview

- Single Object "Domain" made things simple
  - Global edit to change access methods to direct access to structure element
  - domain.numElems()
    - First define macros like domain_numElems()
    - Then directly to structures domain->m_numElems
- OCR has limitations on use of nested data structures
  - Special care needed in designing and initializing domain's structure elements that were pointers

Macros for Datablock 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
    and run with UPC
    and check SHARED pointer usage
- This allowed 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
  - IN() scalars can go to paramv[]
  - Arrays in IN() OUT() INOUT() get converted to DBs and their ocrGuid_t's go in depv[]

Final Steps to EDT

- Habanero C is close syntactically and semantically to DB/EDT.
- From initial Habanero-C version
  - finish{ async IN(inList) OUT(outList){…}}
  - finish { async IN(inList) OUT(outList) edt(inList,outList);
  - finish { async IN() edt(paramc,paramv[],depc,depv[]) }}
- Then as OCR
  - ocrEdtCreateTemplate()
  - ocrEdtCreate()
    - async{} as EDT with EDT_PROP_NONE
    - finish{} as EDT with EDT_PPOP_FINISH
  - ocrAddDependence()
- This translation from Habanero-C to OCR is in the process of being automated

Lulesh 1.0 Status

- High level function flow translated to OCR
- Some low level leaf (OMP parallel loops) translated to demonstrate methods
- Waiting for the new version of lulesh with new physics (multiscale and plasticity) to be release to return to lulesh

Next Steps

- miniGMG
  - HClib on top of OCR
- New lulesh w/multiscale and plasticity (Lulesh-MP?)
- Updating some of the interesting UHPC Apps to OCR
  - SAR, written to explicitly manage the memory hierarchy
  - UTS, interesting load balancing test
- (CnC, HClib, Habanaro-C, etc.) on OCR implementations of the applications as the tools become available