Traleika Glacier X-Stack Status Review

March 25, 2014

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

Disclaimer: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Agenda

- **Traleika Glacier X-Stack Overview** – Shekhar Borkar, Intel – 5 min
- **Applications in OCR on FSim** – Roger Golliver, UIUC – 5 min
- **Applications – LULESH & miniGMG** – John Feo/Ellen Porter, PNNL – 5 min
- **Evaluate TG architecture with Co-Design center's applications (ExMatEx – CoMD proxy app)** – Laura Carrington, Pietro Cicotti, UCSD – 5 min
- **Hierarchically Tiled Arrays** – David Padua/Adam Smith, UIUC – 5 min
- **High-level Compiler (R-Stream)** – Rich Lethin/Muthu Baskaran, Reservoir Labs – 5 min
- **Programming Model & Runtime** – Romain Cledat, Intel – 5 min
- **OCR and Extreme Scale Systems** – Vivek Sarkar/Zoran Budimlic, Rice – 5 min
- **Low-level Tools (LLVM & Binutils)** – Rich Lethin/Muthu Baskaran, Reservoir Labs – 5 min
- **Exascale Simulation** – Kelly Livingston, ETI – 5 min
- **Architecture Overview** – Josep Torrellas, UIUC – 5 min
- **Self-Awareness** – Guang Gao, Stephane Zuckerman, U Del – 5 min
- **Summary** – Shekhar Borkar, Intel – 5 min
- **Q&A** – 25 min

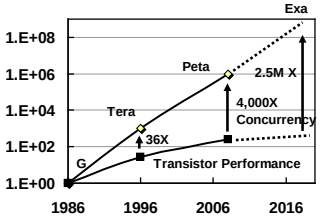
Traleika Glacier X-Stack Overview

Goal: **Intel**

Research and mature software technologies addressing major Exascale challenges and get ready to intercept by 2020

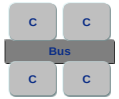
Challenges

Extreme parallelism *O(Billion)*

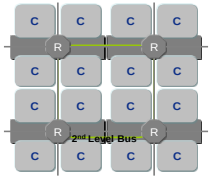


Programming model
Data locality
Legacy compatibility

Data Locality



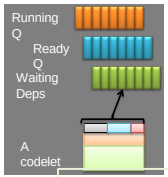
Bus to connect over short distances



Or hierarchical circuit and packet switched networks

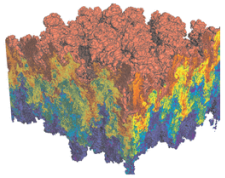
Data movement energy dominates
Code to data vs data to code

Introspective execution model—Self-aware



Event driven tasks
Synchronization
Dynamic scheduling
Runtime system

Refactoring algorithms & applications



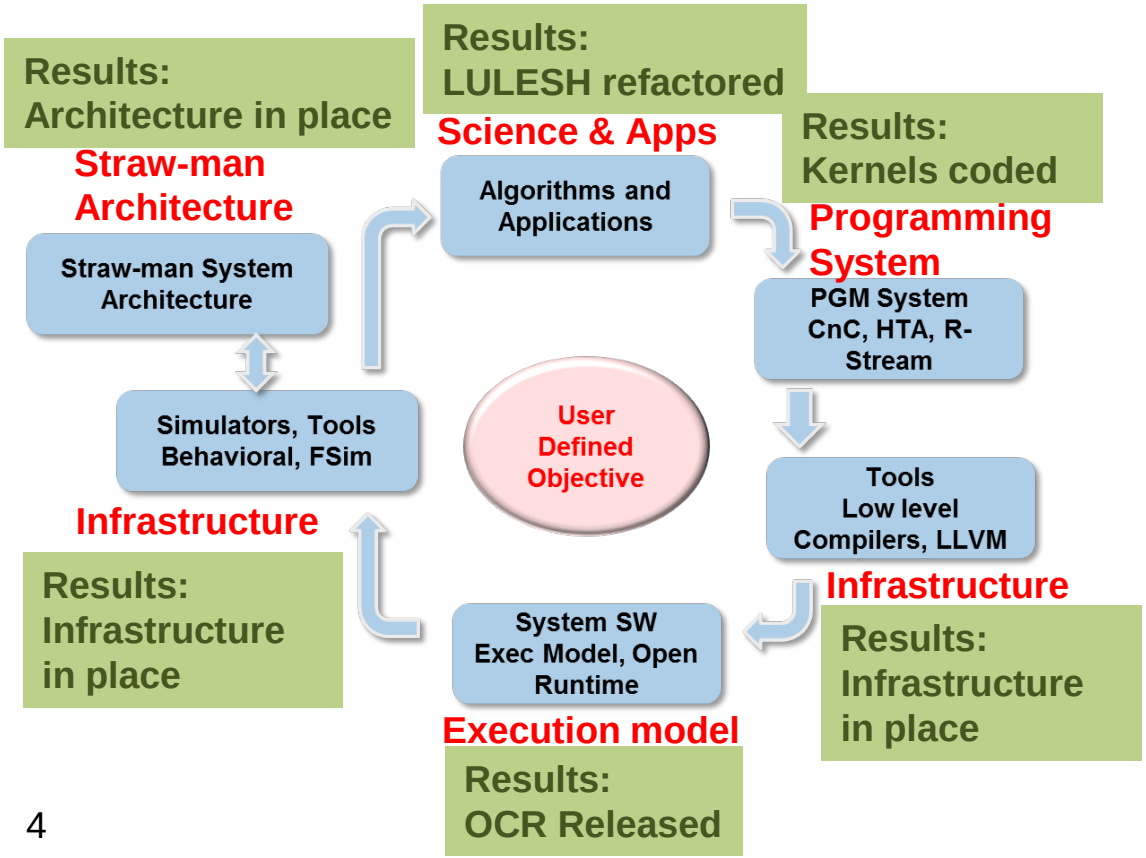
Port the science behind the applications
Co-design

Several technologies identified, need to mature them

Traleika Glacier X-Stack Overview

Intel

Research & Development Flow



Impact

- 2 Application WS with co-design center participation
- Science behind apps
  - LULESH,
  - Combustion,
  - AMR,
  - mini GMG
- Ported on OCR x-86
  - CG-kernel, CoMD
- Will be targeted to TG arch (Fsim)

Traleika Glacier X-Stack Overview

Intel

Desired outcome by Sep'14

- Adaptive-mesh, AMR, refactored as an mini-app
- Mini-GMG, internal to AMR, to be worked by John Bell and Rice
- Show substantial improvement on entire tool chain and FSim
- Ready for May ESR meeting and App Workshop 3 in August

Desired outcome by Sep'15

- Individual SW components matured
- Key applications refactored with science
- Complete SW stack evaluated on the simulator
- System architecture in place

Applications in OCR on FSim

Overview

- **Problem**
  - Need for OCR on FSim implementations of applications of interest for development and evaluation of HW/SW proposals
  - FSim is parallel, there is a need for applications that run on more than a single block
  - No semi-automatic path for proxy level complexity applications to run on FSim efficiently
  - Initial ports to OCR that run well on x86_64 Linux, require additional effort to run effectively on FSim
    - Parallelization via EDTs/DBs is the necessary first step
    - Managing DBs in the memory hierarchy is the more interesting problem
    - No POSIX I/O in OCR on FSim

Applications in OCR on FSim

Overview

- **Solution**
  - Using Humans as compilers working on benchmarks and proxy applications
  - Initially tried to construct source to source mapping from original MPI/OpenMP proxy application to OCR implementation
    - Recent experience suggests starting with a clean OCR implementation of kernel algorithms, informed by how the proxy uses them, will produce a better implementation
  - Range of sophistication of the solutions for evaluation experiments
    - Simple use of data blocks (to allow OCR to manage)
    - More explicit use of data block management within the memory hierarchy as target

Applications in OCR on FSim

Overview

- **Recent Results**
  - SCF (Self Consistent Field from NW-Chem)
  - HPCC Stream and FFT
    - FFT single large DB
  - HPCC matrix multiply
    - Traditionally: explicitly blocked to core's last level cach (LLC) with compiler managing register file size sub-blocks
    - OCR/FSim: HTA style management of data necessary at each level of the memory hierarchy.
  - HPCG
    - Text book implementation of CG
  - LULESH 1.0 (Serial Single Domain on FSIM)

Applications in OCR on FSim

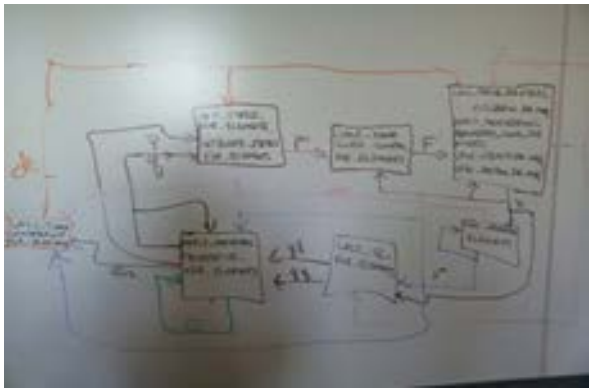
Overview

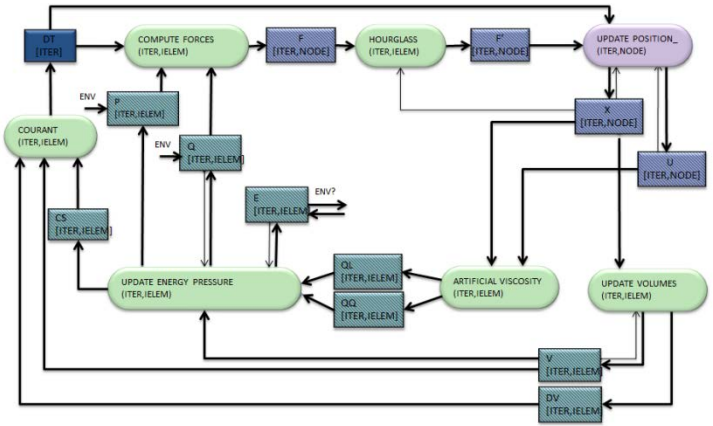
- **Impact**
  - FSim environment and existing tool chain is robust
    - Able to run week long simulation runs without problems
  - SCF identified need for improved transcendental functions, so CRLIBM ported to FSim
  - Changing view on importance of software managed cache
  - Investigating additional common operations needed to help manage DBs
- **Desired outcomes by Sep'14**
  - miniGMG and the soon to be released LULESH 2+ new physics
    - high level CnC
    - low level OCR on FSim
  - Expand implementations of HPCC and HPCG
  - Eg. Single node FFT to parallel distributed 1D-FFT
- **Desired outcomes by Sep'15**
  - Expect productivity to be expanded with the availability of more programming tools (Without some HHL support for programming model progress will continue be slower than desired)

Applications – LULESH & miniGMG

PNNL

- **Problem**
  - Application development costs
  - Good performance requires expertise
    - Programming models
    - Underlying architecture
- **Solution**
  - Develop programming models
    - Reduce complexities
    - Easy to use
- **Recent results**
  - Translated LULESH to CnC
    - From informal whiteboard sketch
    - To a formal specification
    - To application

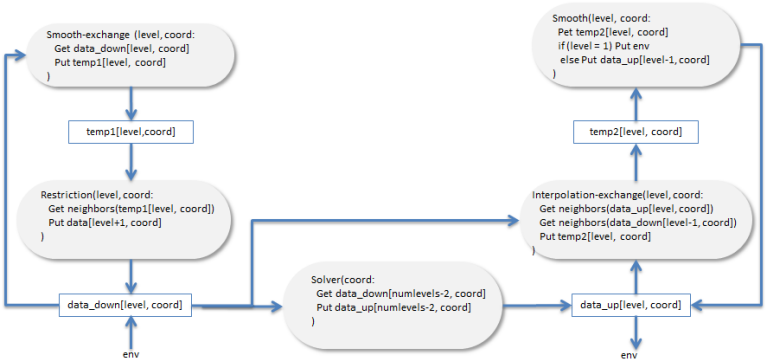




Applications – LULESH & miniGMG

PNNL

- **Desired outcome by September 2014**
  - Execute LULESH on FSim
  - Evaluate and improve code
  - Provide feedback for FSim
  - Transform miniGMG to CnC
- **Desired outcome by September 2015**
  - Execute miniGMG on Fsim
  - Evaluate and improve code
  - Profide feedback for Fsim
  - Test CnC tuning models for both applications



Evaluate TG architecture with Co-Design center's applications (ExMatEx – CoMD proxy app)

UCSD

Problem: How to redesign key applications of the Co-Design centers and evaluate the TG architecture

- Example CoMD algorithm: classical short-range molecular dynamics

Solution: Implement proxy apps in OCR

- Work with ExMatEx scientist to understand algorithmic options
- Characterize & redesign CoMD for OCR programming model
- Explore design space and characterize tradeoffs in performance, data movement, and energy consumption using TG FSIM simulator

Evaluate TG architecture with Co-Design center's applications

UCSD

Desired Impact and Current Status

Impact:

- Test the software stack with scalable benchmarks
- Provide a methodology to characterize algorithms and understand design tradeoffs at Exascale
- Provide guidelines and support to the Co-Design centers in designing applications for Exascale

Recent Results:

- Implemented CoMD Lennard-Jones potential on OCR/x86
- Testing scalability on multi-core systems

Evaluate TG architecture with Co-Design center's applications

UCSD

Desired outcome by Sep'14

- Complete second algorithm implementation of CoMD (Embedded Atom Method) and use for investigation of TG architecture on FSIM
- Compare sensitivity to variations in core speed with the bulk-synchronous implementation
- Characterize data-movement and energy consumption of the different implementations
    - Example: atomic floating point operations vs. OCR exclusive writes

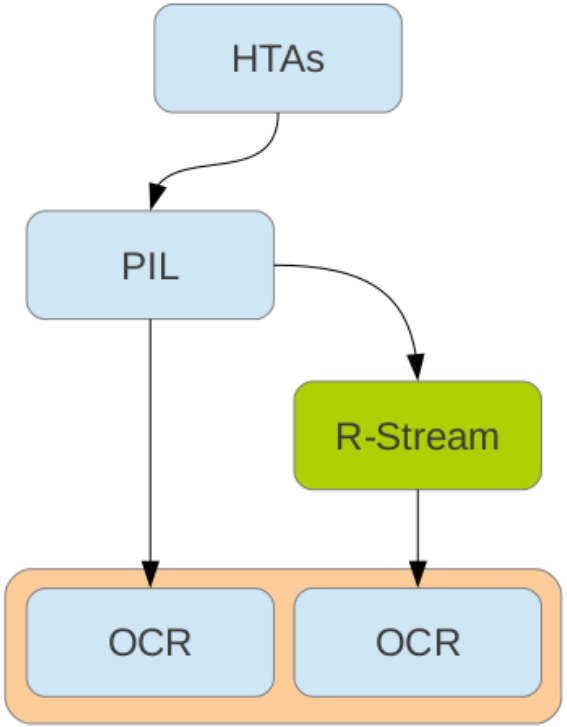
Desired outcome by Sep'15

- Target other MD variants (e.g. fewer atoms/core)
- Target other proxy applications from the Co-Design centers

Hierarchically Tiled Arrays

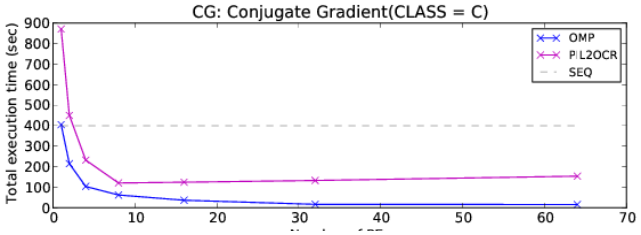
UIUC

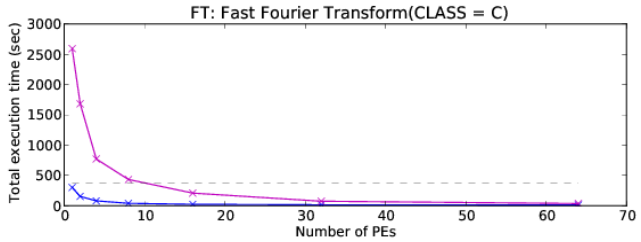
- Overall Goals
  - Provide a convenient and efficient notation to use OCR.
  - Based on abstractions typically representing data parallel operations
- Accomplishments of last two quarters:
  - Refactored implementation to target new OCR v0.8 interface.
  - Implemented NAS benchmarks in HTA (EP, CG, FT, IS, MG, LU).
    - Obtained performance numbers.
    - Generated some of the most complex OCR codes available for testing purposes.
  - Worked on integration with R-Stream.
  - Designed transformation to generate OCR code in SPMD form for efficiency.

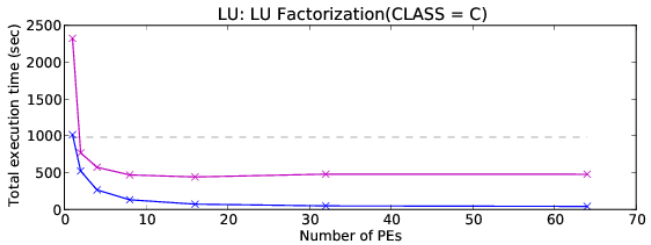


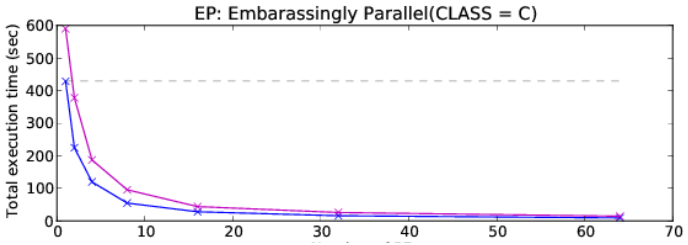
HTA Performance Numbers

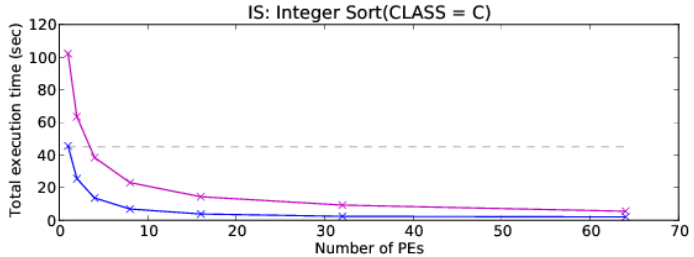
UIUC

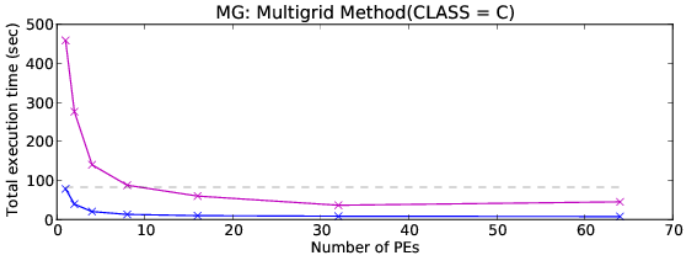




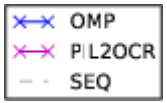








- NAS benchmarks Class C
- Shared memory OCR v0.8
- Numbers collected on four socket, 10 core Intel Xeon E7-4860 machine with 128 GB of RAM
- Results are preliminary
- Overheads introduced by HTA and OCR will be reduced
- CG results are only meaningful to 8 PEs because of configuration file issue



Hierarchically Tiled Arrays

UIUC

- **Goals for September ‘14**
  - Complete SPMD implementation. Evaluate the overhead resulting from HTA to OCR translation.
  - Complete R-Stream integration
- **Goals for September ‘15**
  - Extend HTA design and implementation to include multiple levels of parallelism, irregular parallelism, and irregular partitions.
  - Implement a variety of other benchmarks.
    - Mini GMG, AMR, etc.
  - Evaluate and tune for performance.
  - Evaluate programmability using objective metrics (e.g. number of operations).

High-level Compiler (R-Stream)

Reservoir Labs

Problem

- Need an efficient programming system for addressing Exascale challenges, namely, programmability, energy efficiency, and scalability

Solution

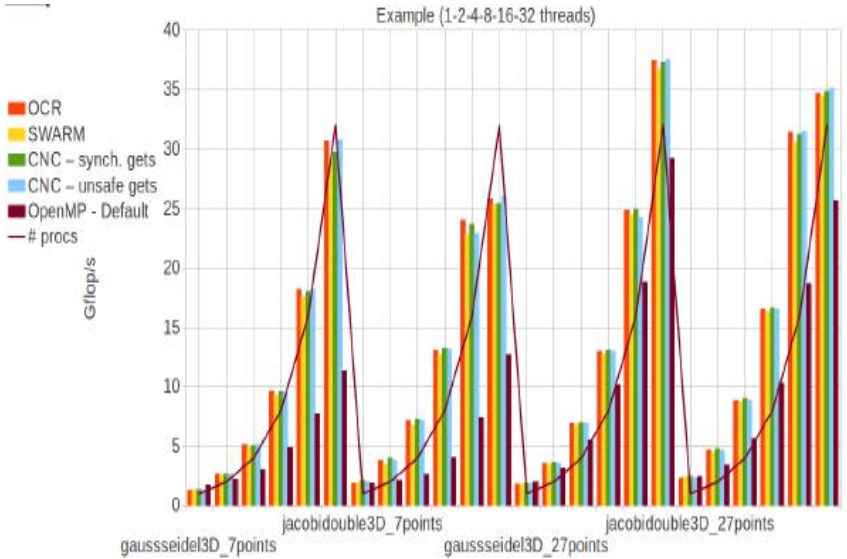
- High-level compiler optimizations that address the Exascale challenges
- Automatic generation of efficient parallel code leveraging runtimes (CnC, OCR)
  - Improve data locality and reduce communication ⇒ energy efficiency
  - Expose high concurrency ⇒ scalability
  - Enable automatic code generation and data management ⇒ productivity

High-level Compiler (R-Stream)

Reservoir Labs

Recent Results and Impact

- Support for automatic generation of code for CnC and OCR
  - ★ Auto-parallelized C benchmarks (stencils, solves, etc.) to to CnC and OCR quickly
  - ★ Showed R-Stream's ability to achieve data locality and expose task parallelism
  - ★ Validated scaling capabilities of EDT runtimes



- Parallelism and data locality optimizations for stencil kernels - prevalent in DoE apps (e.g. stencils from BoxLib and UQ kernels)

High-level Compiler (R-Stream)

Reservoir Labs

Desired Outcomes

- **Sep’14**
  - Support for OCR optimizations - data blocks, hierarchical EDTs, and locality-aware scheduling
  - Auto-generate OCR code for miniGMG benchmark through R-Stream
- **Sep’15**
  - R-Stream optimizations for deep hierarchy of processors and memories exhibiting scalability
  - Support for efficient parallel code generation for OCR with positive interference of R-Stream and OCR optimizations

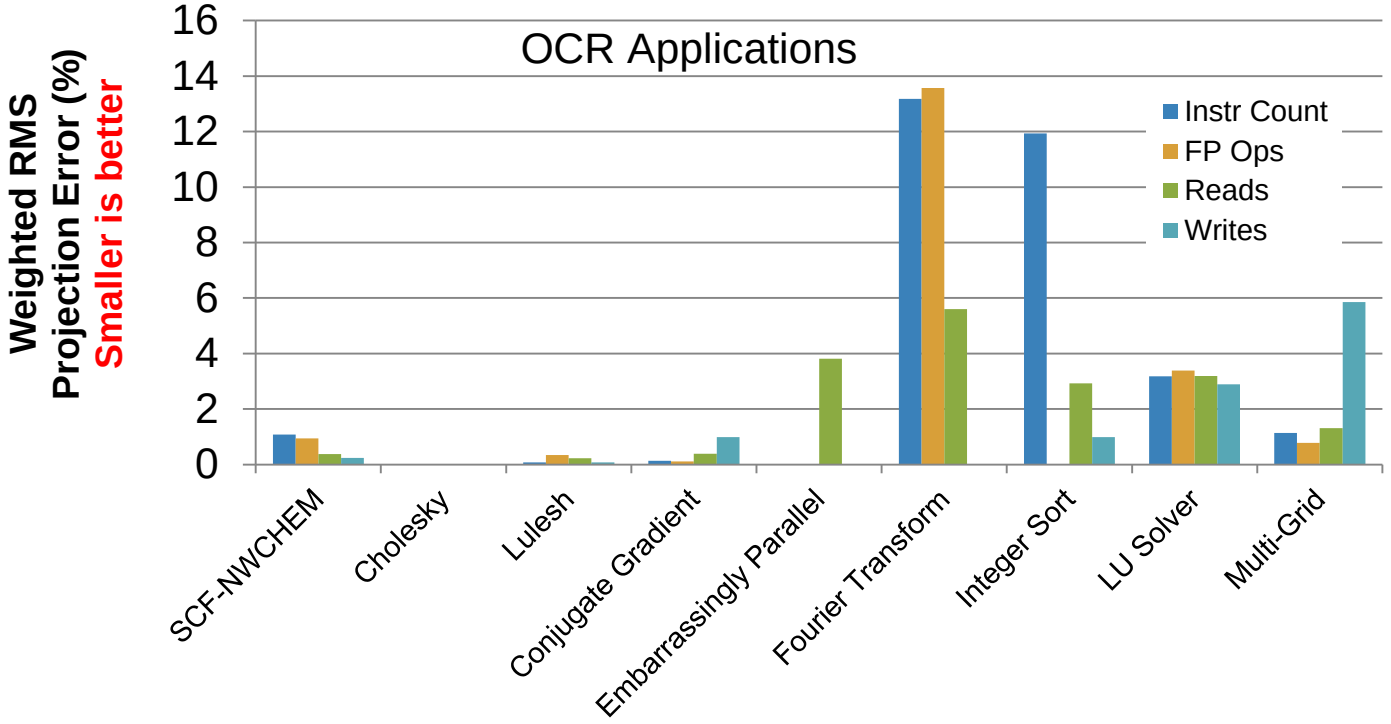
Programming Model & Runtime

Intel

- Problem
  - Explore event-driven programming model & Runtime
- Solution
  - Design and expose a low-level API for hero programmers and higher-level tools
  - Provide an extensible framework to explore runtime components that supports multiple targets
- Recent accomplishments
  - OCR-x86 released with stable user-level API
  - Scalability using workstation clusters for speedy explorations
  - Work continues to port on Fsim to evaluate TG architecture

Dynamic System Optimization—Introspection

Intel



Introspection and our approach looks promising

Programming Model & Runtime

Intel

- Desired outcome by September 2014
  - Stable and scalable OCR on FSim and x-86 clusters
  - Comparison with current programming models
  - Validation on several applications/kernels (LULESH, CoMD, miniGMG, AMR, etc.)
- Desired outcome by September 2015
  - Release reference implementation for OCR
  - Demonstrate benefits of co-designed framework
  - Implement advanced introspection in system SW

OCR and Extreme Scale Systems

Rice

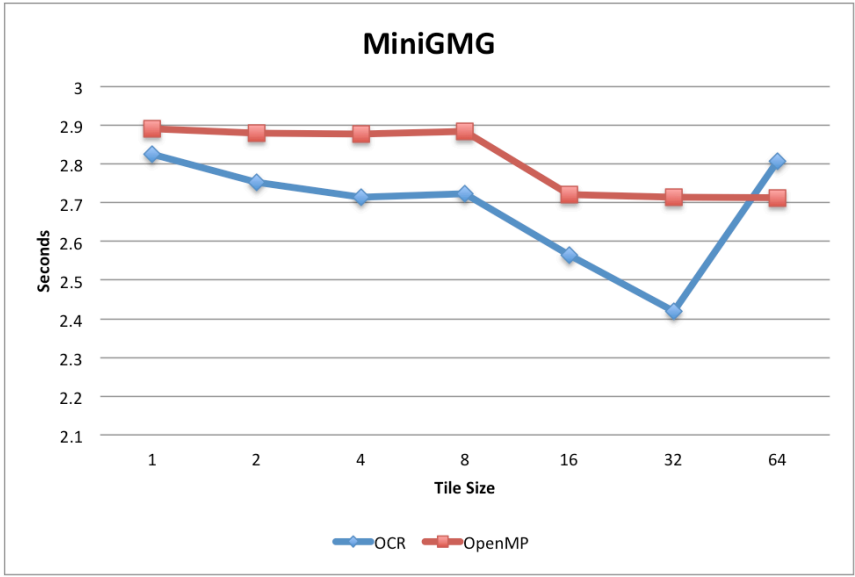
- Massively multi-core (~ 100’s of cores/chip)
- Performance driven by parallelism, constrained by energy
- Subject to frequent faults and failures
- Key challenges: Energy Efficiency, Concurrency, Resiliency
- Open Community Runtime (OCR) is a framework that:
  - is representative of execution models 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

OCR

Rice

- **A fine-grained, asynchronous, event-driven runtime framework**
  - Event-driven tasks (EDT's)
  - Events
  - Data blocks
  - C library implementation
  - Designed to be a target for higher-level compilers and libraries
- **Release 0.8**
  - Works on shared-memory multiprocessors
  - Works on FSim
  - Performance comparable to OpenMP
  - Targeted by RStream, CnC, HTA, HCLib
  - Sparks a lot of interest far beyond the TG project

OCR vs OpenMP



OCR Current and Future Work

Rice

- Distributed OCR: Seamless integration of the OCR model with the cluster architecture
- Accelerated OCR: Integration of the OCR model with GPU's
- Tuning extensions: affinities between EDT's and data blocks
- Distributed CnC on distributed OCR: support for distributed, concurrent maps of GUID's
- Introspection and adaptation
- Porting OCR to other interesting platforms
  - Texas Instruments' Keystone platform with ARM and DSP cores

Low-level Tools (LLVM & Binutils)

Reservoir Labs

Problem

- Need a low-level compiler, assembler, disassembler, and linker for the TG architecture
- Update these low-level components to the evolving TG architecture
  - for e.g. moving the XEs in TG architecture from a 32-bit ISA to a 64-bit ISA

Solution

- LLVM - open source low-level compiler - retargeted for the XE in TG architecture
- Binutils - open source collection of tools including an assembler, disassembler, linker, and debugger - retargeted for TG
- New versions of LLVM and binutils supporting different versions of TG architecture

Low-level Tools (LLVM & Binutils)

Reservoir Labs

Recent Results

- Support for 32-bit ISA
  - Bug fixes (correctness and performance) in LLVM backend for the 32-bit ISA
  - Updated binutils to add support for new queue-management (QMA) instructions
- Supporting Intel’s exploration on the development of 64-bit ISA
  - Reviewed proposed instructions, addressing modes, and encodings, in light of the compiler’s needs
  - Suggested new instructions
- Ongoing development of new versions of binutils and LLVM for 64-bit ISA
  - Moved to latest open source version of LLVM and binutils
  - Moving to Elf 64 executable format for new version of binutils

Low-level Tools (LLVM & Binutils)

Reservoir Labs

Impact

- Basic part of the software stack
  - Up-to-date LLVM and binutils is critical for the stack

Desired Outcomes

- Sep’14 objective
  - Complete new versions LLVM and binutils for 64-bit ISA
  - Continue support for 32-bit ISA
- Sep’15 objective
  - Whatever ISA is eventually chosen, the objective is to have excellent implementations of binutils and LLVM available promptly

Exascale Simulation

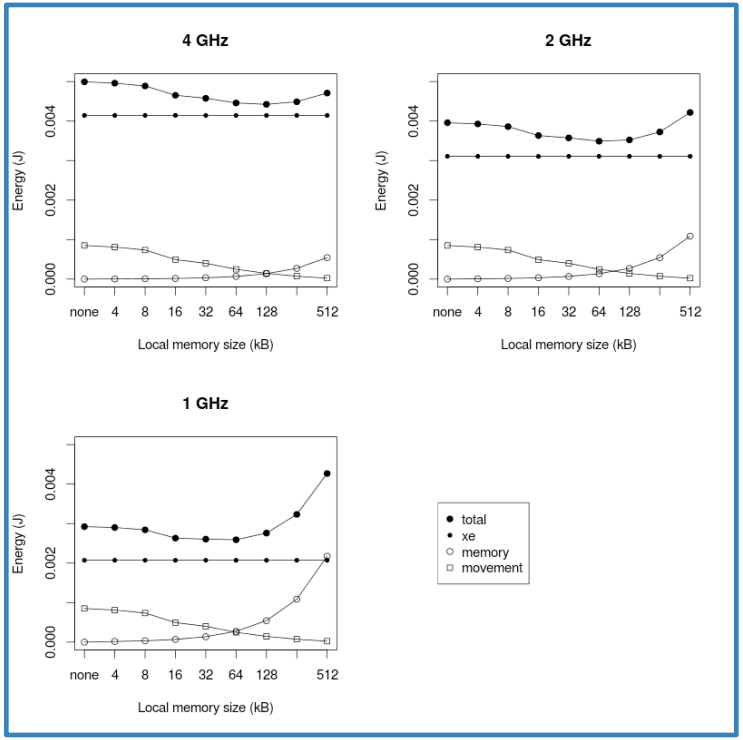
ETI

- **Problem**
  - X-Stack needs actionable information about future chip capabilities and behaviors
- **Solution**
  - Maintain and update the TG ISA using FSim, Intel's functionally accurate high performance simulator
  - Simulate chip performance using Lamport clocks (maintain causal relations of program and provide a basis for static energy)
  - Calculate basic Architecture Level Power Simulations (ALPS) for dynamic energy consumption
- **Recent Accomplishments**
  - Provided energy accounting framework for processor and data movement across chip
  - Upgrading ISA to 64-bit encoding
- **Impact**
  - Simulation of dynamic hardware properties, critical to reaching Exascale, provides realistic view of a future chip and promotes the benefits of dynamic runtime stacks

Energy Consumption

ETI

- **Problem**
  - Tracking and understanding energy consumption in a large asynchronous program can be unwieldy
- **Solution**
  - Construct an energy and data movement visualizer based on the physical layout of the chip and the logical layout of the program
- **Recent Accomplishments**
  - Performed initial study of localized energy usage due to data movement and memory leakage. Determined the impact of both due to memory size
  - Extended study to include memory hierarchy and integrated into the OCR x86 framework to produce a data movement visualizer



Data Movement Study Results

Simulation Goals for ‘14 and ‘15

ETI

- **2014**
  - Finish 64bit ISA implementation
  - Implement resource hazards in pipeline for timing purposes
  - Refine energy framework to include more detailed accounting of energy
  - Improve simulation speed
- **2015**
  - Provide runtime scheduling and memory allocation strategies based on data movement studies
  - Provide support for introspection and power management in FSim

Architecture Overview

UIUC

Recent Accomplishments

- **API for Software-Managed On-Chip Memory Hierarchy**
  - Have developed an API based on Writeback and Invalidate
  - Have a compiler pass [together with Prof. Saddayappan] to automatically use it in codes
  - Working on implementing the API in FSIM

```
1 for (t = 0; t <= tsteps-1; t++) {
2   #pragma omp parallel private (myid, i1, i2) {
3     myid = omp_get_thread_num ();
4     for (i1 =myid; i1 <= floor((n-2)/16); i1 +=8) {
5       if (t >= 1){
6         invalidate_dword (&A[16 i1 -1]);
7         invalidate_dword (&A[16 i1 + 16]);
8       }
9
10      for (i2 =max (i1  16, 2); i2 <=min (i1 16+15, n-2); i2 ++){
11        S1: B[i2]=0.33333 (A[i2-1]+A[i2]+A[i2+1]);
12      }
13
14      if (t == tsteps-1)
15        writeback_range (&B[i1  16], sizeof (double) * 16);
16    }
17 }
18
```

[Tavarageri,Kim,Torrellas,Sadayappan '13]

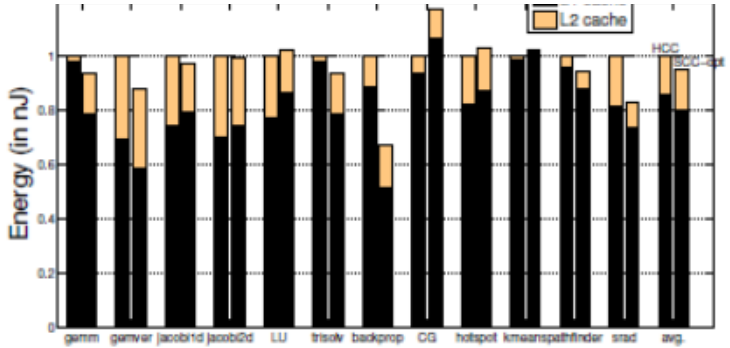


Figure 14: L1 and L2 Cache Energy (lower, the better). The first bar shows HCC energy and second bar SCC-opt energy

Architecture Overview

UIUC

- **API for Energy Management**
  - Have developed an API
  - Have integrated the API with the Rose compiler [Quinlan]. Can call it from Rose
  - Working with Reservoir/ETI on implementing a subset in FSIM

Features:

1. Select the Voltage, f level of a compute or memory resource
2. Set a compute or memory resource into one of three states
   - Power Off, Clock Off
   - Power On, Clock Off
   - Power On, Clock On

List of compute and memory resources can be blocks, processors, memories, caches, func. units

Architecture Overview

Future Plans

- Work with team to decide what exact APIs to implement in FSIM
- Implement the APIs in FSIM (functionality and energy)
- Take small codes and insert by hand calls to the APIs
  - Evaluate them for performance and energy saved
- Longer term: have the compiler automatically insert calls to the APIs for applications

Self-Awareness

U Del

Problem

- Extreme-scale systems must act and react to numerous events to meet user goals

Solution

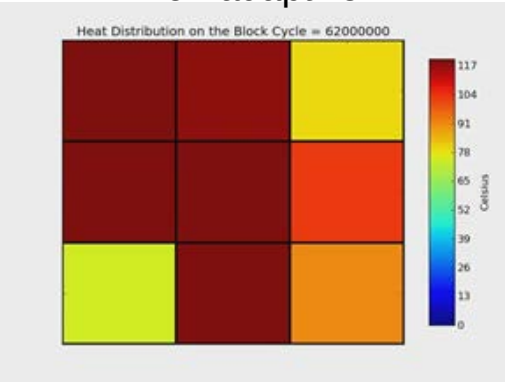
- Design a distributed and hierarchical self-aware system which leverages introspection and adaptation mechanisms to help with resource management

Recent results

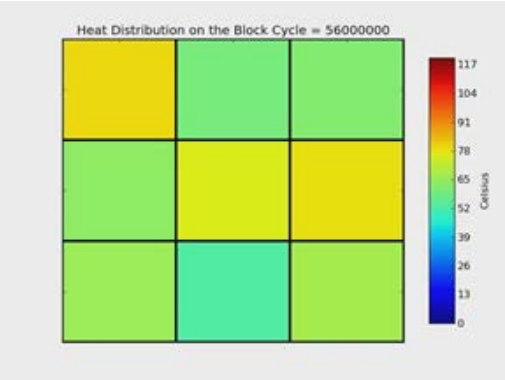
- Specified a self-aware API
- Observation: leverage HW introspection mechanisms
- Heat model to evaluate the impact of temperatures on execution
- Decision: prioritization to activate mechanisms as a function of observed events
- Actions currently implemented at the block level:
  - Clock-gating (CG), to avoid catastrophic failure
  - DVFS, as a function of temperature thresholds

Impact: See the real “heat map” on the right-hand side

Non-adaptive

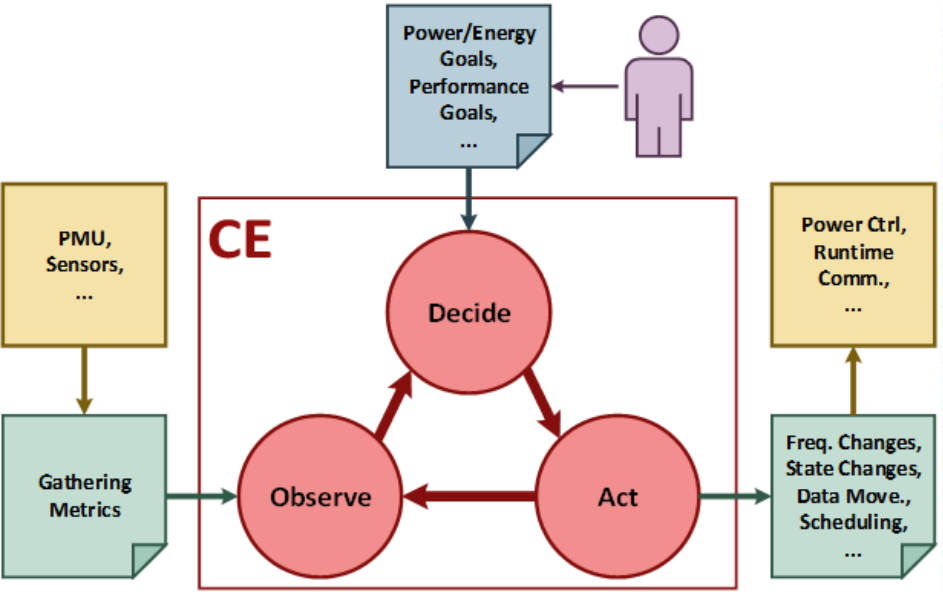


Self-adaptive

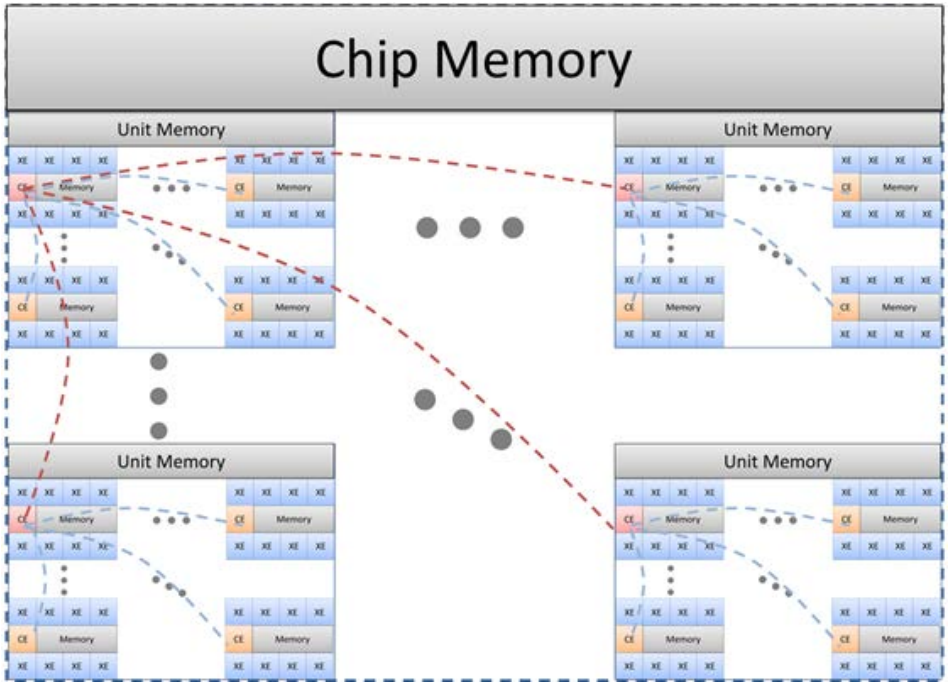


Self-Awareness – High-Level Concepts

U Del



Local Observe-Decide-Act Loop



Distributed and Hierarchical Observe-Decide-Act Loop

Self-Awareness – Desired Outcomes

U Del

Goal for Sep'14: A functioning prototype for a hierarchical and distributed self-aware system on the Traleika Glacier's simulator.

- Leverage experience with our self-aware prototype and integrate it into FSim
- Evaluate our framework on Fsim using existing program
- Propose various resource management policies to achieve self-awareness

Goal for Sep'15: A fully specified fine-grain event-driven execution model which integrates self-aware aspects at its core.

- Augment OCR with our self-aware API
- Run DOE mini-apps already ported to OCR and running on FSim using our self-aware system software
- Evaluate and improve the behavior of the system software using different objective functions for various workloads

TG Summary

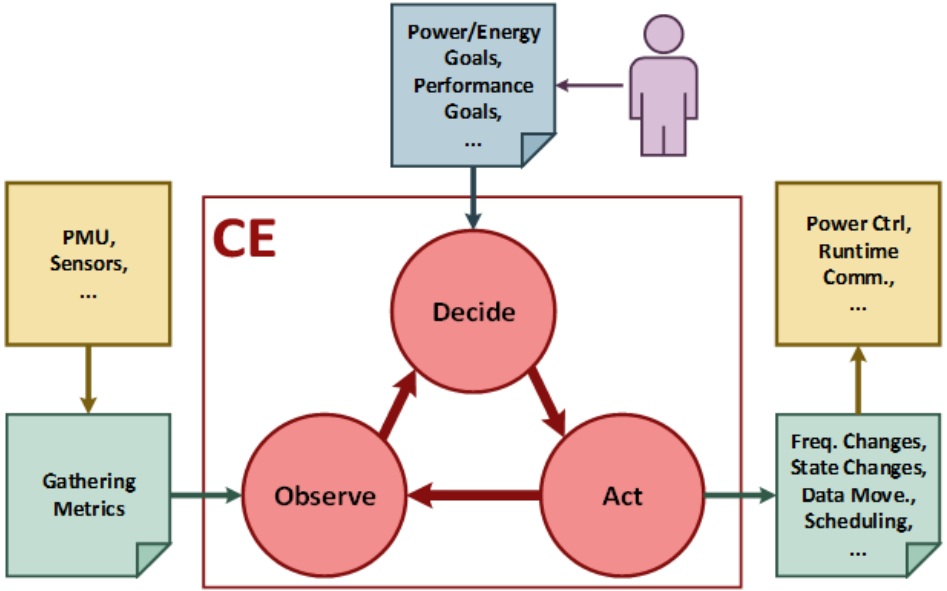
Half-way through the program

	2012	2013				2014				2015		
	Q4'12	Q1'13	Q2'13	Q3'13	Q4'13	Q1'14	Q2'14	Q3'14	Q4'14	Q1'15	Q2'15	Q3'15
Algorithms and applications												
Proxy app evalution for O(compute)	<----------	----------	---------->									
Proxy app evalution for O(comm)			<----------	----------	---------->							
Evaluation of system architecture						-----V2.0	----------	----------	------V2.5	----------	----------	-----V3.0>
Programming system												
Select apps coded, for runtime system	<----------	----------	----------	-----V2.0-	----------	----------	----------	----------	------V2.5	----------	----------	-----V3.0>
Tools	<C+binuti	ls-V2.0-->	<-------De	bugger-->	----------	----------	------V2.5	----------	----------	------V3.0	----------	----------
Resiliency	<---Frame	work----->	<-----Reac	tive------>	<----Reco	very----->	<----Proac	tive------>	<----------	Evaluate	Validate	---------->
Execution Model, Sys SW, Runtime	<--Mod fo	r IRR ----->	<-Intellige	nt Sched-->	<-Eval-----	-----V2.0	----------	----------	------V2.5	----------	------OCR	-----V3.0>
Architecture, simulators	<A V2.0>	<Sim2.0 >	<Fault M>	<Timing >	<--Eval-->	<A V2.5>	<Sim 2.5>	<--Eval-->	<A V3.0>	<Sim 3.0>	<------Eval	uate----->
System Evaluation	<----------	----------	----------	----------	For 2.5--->	----------	----------	For 3.0--->	----------	----------	----------	---------->

- Straw-man architecture is in place
- Programming system components identified, and being matured
- Infrastructure (simulators, tools) developed
- Open community runtime system released
- Applications refactored with understanding of science
- Introspective system SW research gaining traction

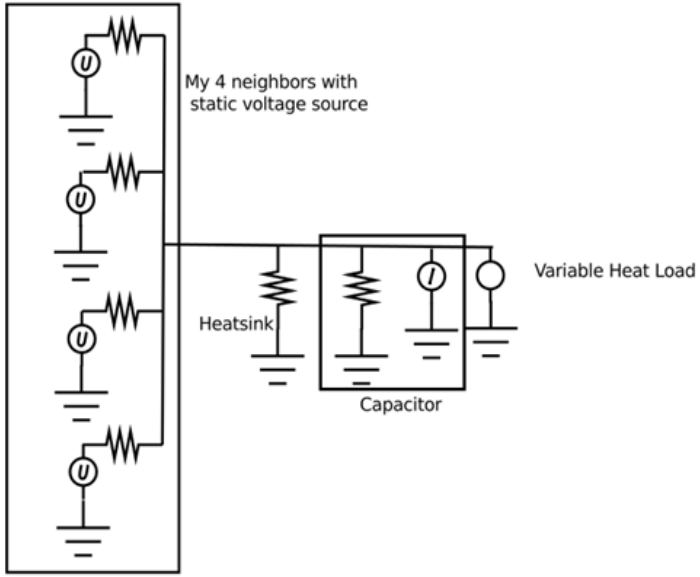
Back-up

Self-aware Feedback Mechanisms for Adaptation

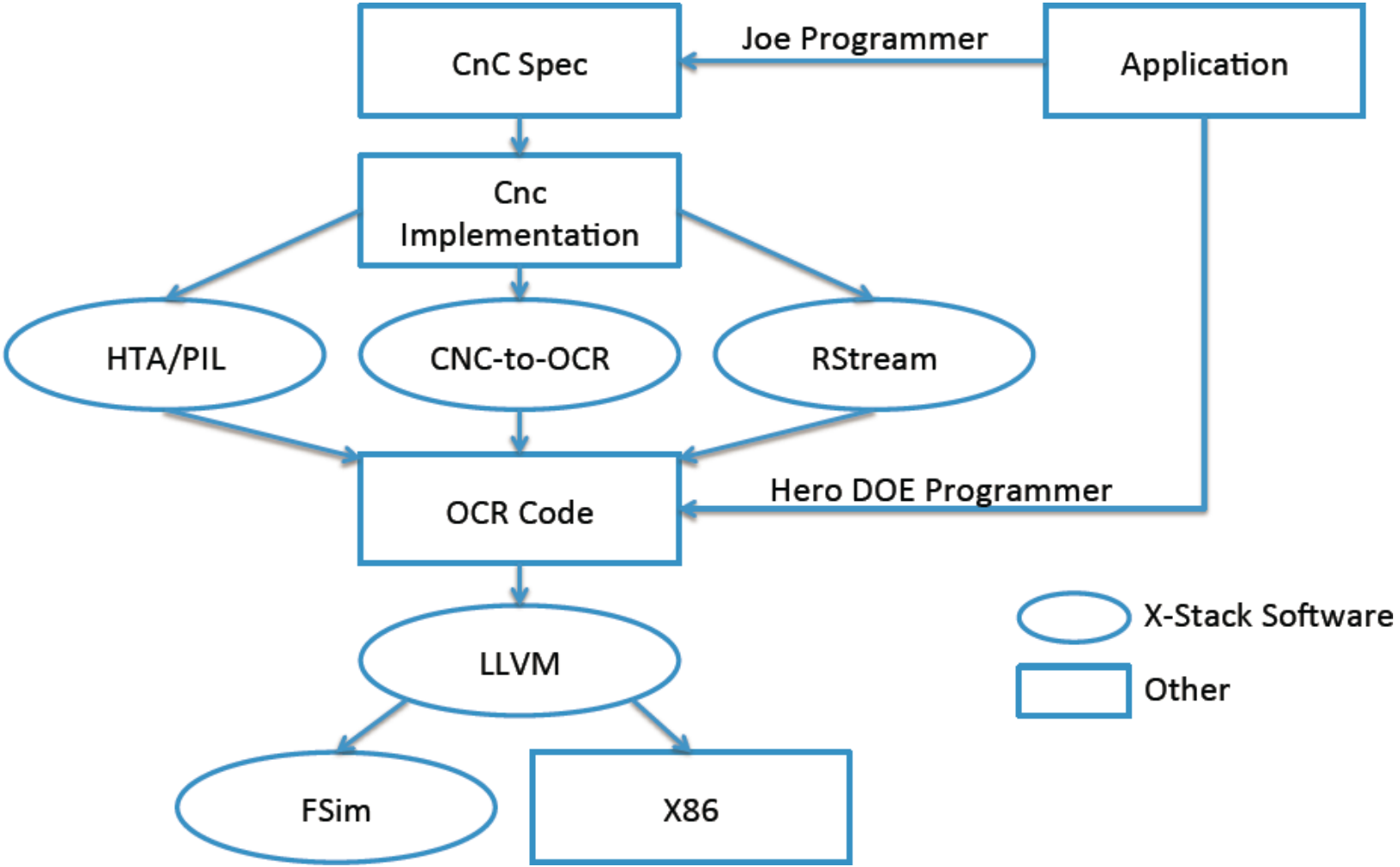


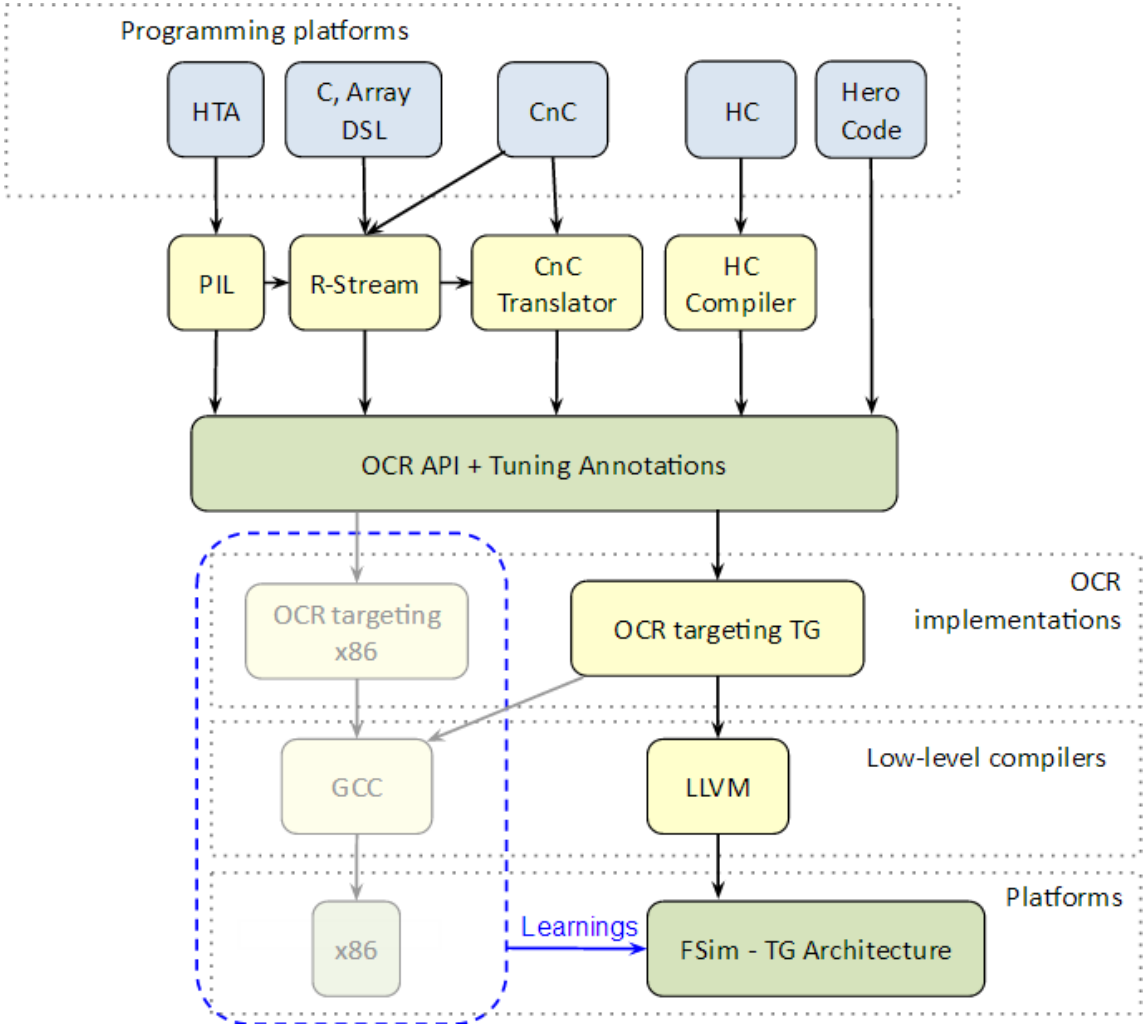
- Observe-Decide-Act (ODA) mechanisms built into the system runtime.
  - User provided power/energy and performance goals.
  - Hardware supported observations and metric gathering mechanisms.
  - Fine-grained control over the system resources.

Heat Modeling



Duality of Conduction	
Electrical	**Thermal**
Voltage (V)	Temperature (T)
Current (A)	Heat Flow (W)
Resistance (Ω)	Thermal Resistance ($W*K^{-1}$)
Capacitance (F)	Thermal Mass ($J*K^{-1}$)


CnC Spec
Joe Programmer
Application
Cnc Implementation
HTA/PIL
CNC-to-OCR
RStream
OCR Code
Hero DOE Programmer
LLVM
FSim
X86
X-Stack Software
Other


Programming platforms
HTA
C, Array DSL
CnC
HC
Hero Code
PIL
R-Stream
CnC Translator
HC Compiler
OCR API + Tuning Annotations
OCR targeting x86
OCR targeting TG
OCR implementations
GCC
LLVM
Low-level compilers
x86
Learnings
FSim - TG Architecture
Platforms