Traleika Glacier

Shekhar Borkar

Acknowledgment: TG Team

Intel Corp.

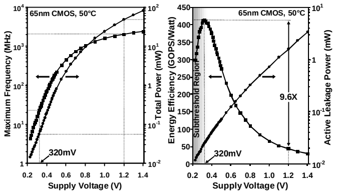
May 28, 2014

Outline

- Technology outlook and challenges
- Vision & Status
- SW Stack for both:
  - Evolutionary & Revolutionary approaches
- Open community runtime for research
- Applications and results
- Summary

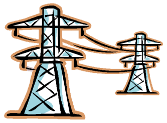
Exascale Technology Challenges

NTV Logic & Memory for low energy



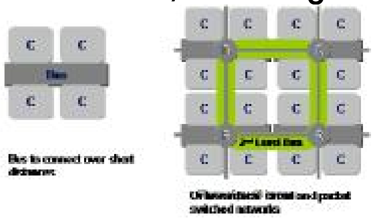
Break the Vmin barrier
FPU, logic & latches
Register Files,
SRAM

Fine grain energy & power management



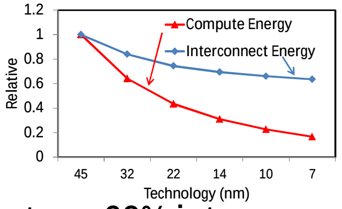
Voltage Regulator
Buck or Switched Cap
Power gating, frequency control

Hierarchical, heterogeneous IC fabric


C
Bus
Bus to connect over short distances
2nd Level Bus
Or hierarchical circuit and packet switched networks

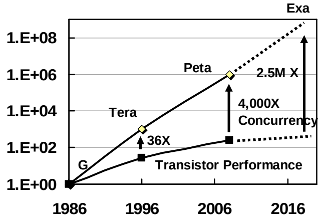
Busses
X Bars
Circuit & Packet
Switched

Data movement becomes expensive



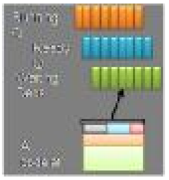
6X compute vs 60% interconnect energy

Extreme parallelism *O(billion)*



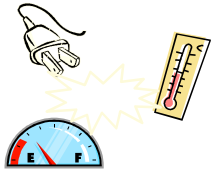
Programming model
Data locality
Legacy compatibility

New introspective execution model



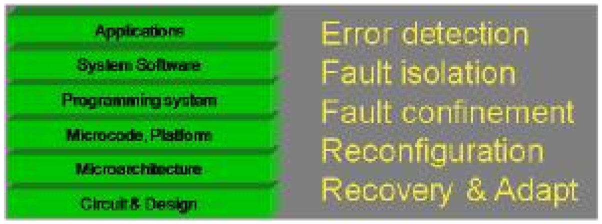
Tiny threads
Synchronization
Dynamic scheduling
Runtime system

Self awareness



Observation based
Monitor, continuously adapt
Objective function based
runtime optimization

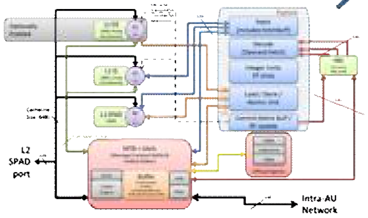
System level resiliency research

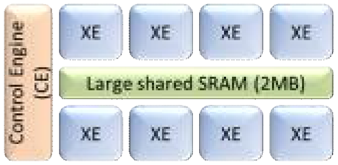


Error detection
Fault isolation
Fault confinement
Reconfiguration
Recovery & Adapt

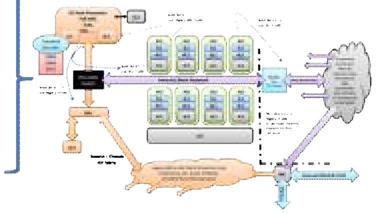
Straw-man HW System Architecture

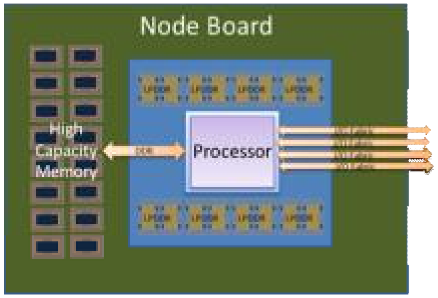
Core Microarchitecture



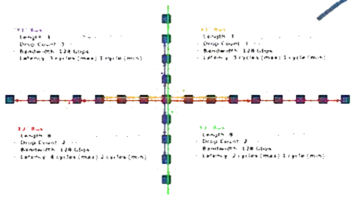


Block Microarchitecture

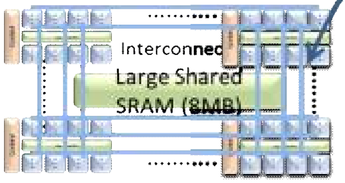




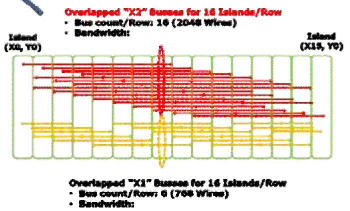
Interconnect Architecture

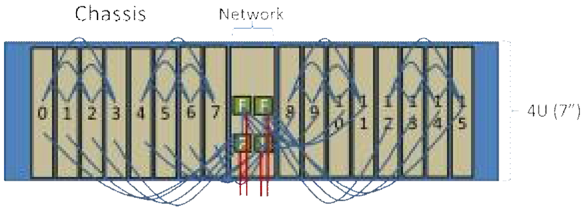


Cluster (16 x)

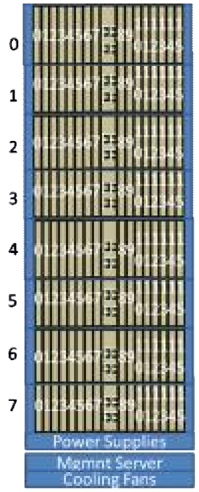


Local Interconnect Plan

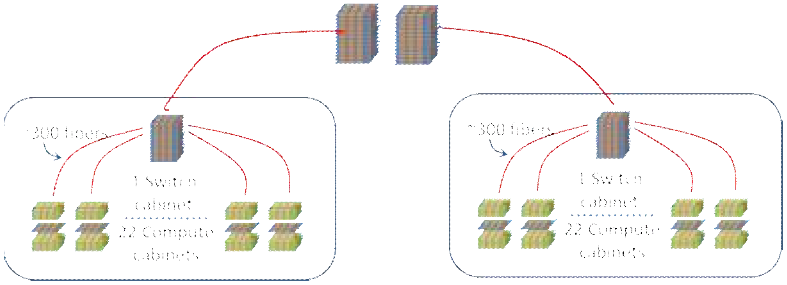

Overlapped "X2" Busses for 16 Islands/Row
• Bus count/Row: 16 (2048 Wires)
• Bandwidth:
Island
(X0, Y0)
Island
(X15, Y0)
Overlapped "X1" Busses for 16 Islands/Row
• Bus count/Row: 6 (768 Wires)
• Bandwidth:



Cabinet

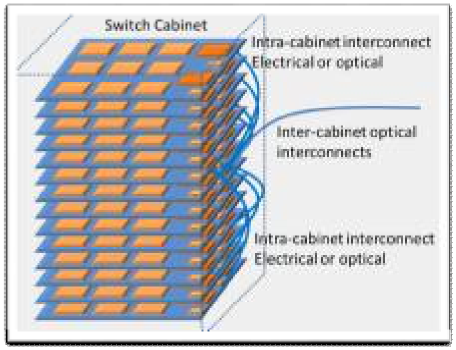


Exascale System (2022)

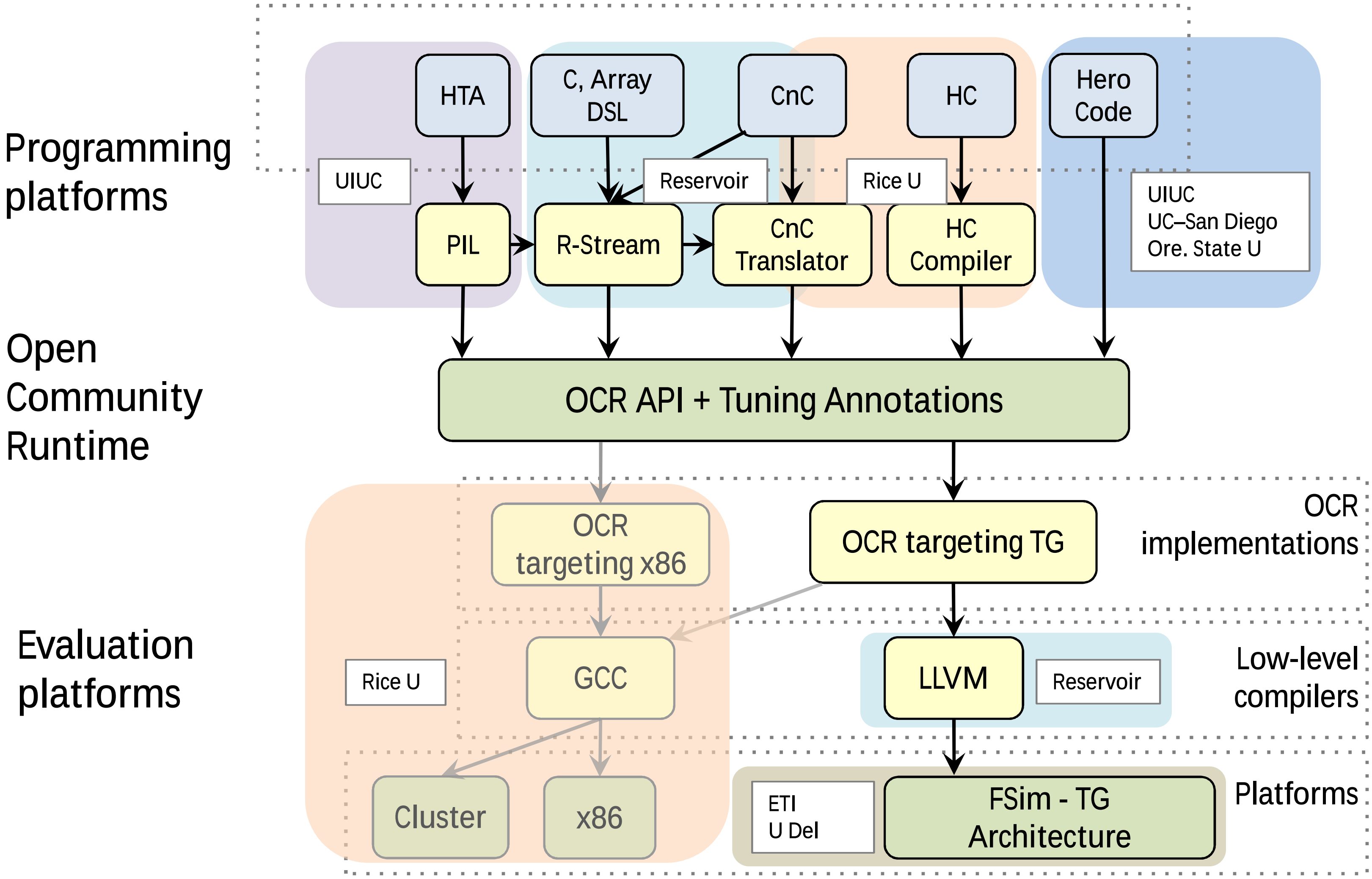


1 EF Peak, 20 MW

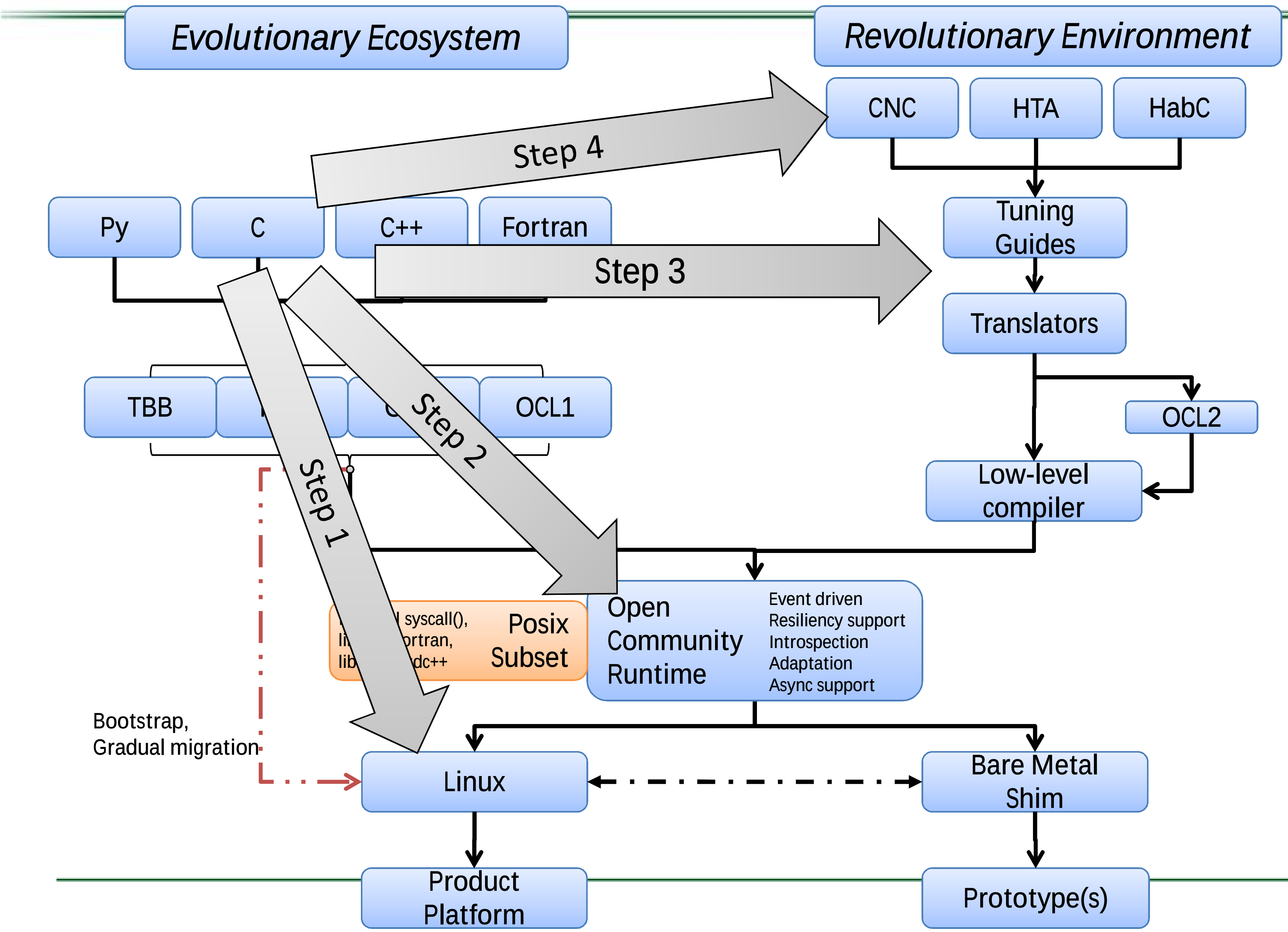
Switch Cabinet



X-Stack (TG) Software Stack



Legacy Support



Evolutionary Ecosystem

Revolutionary Environment

CNC

HTA

HabC

Step 4

Py

C

C++

Fortran

Step 3

Tuning Guides

Translators

TBB

OCL1

Step 2

OCL2

Low-level compiler

Step 1

l syscall(),
li ortran,
lib dc++

Posix Subset

Open Community Runtime

Event driven
Resiliency support
Introspection
Adaptation
Async support

Bootstrap,
Gradual migration

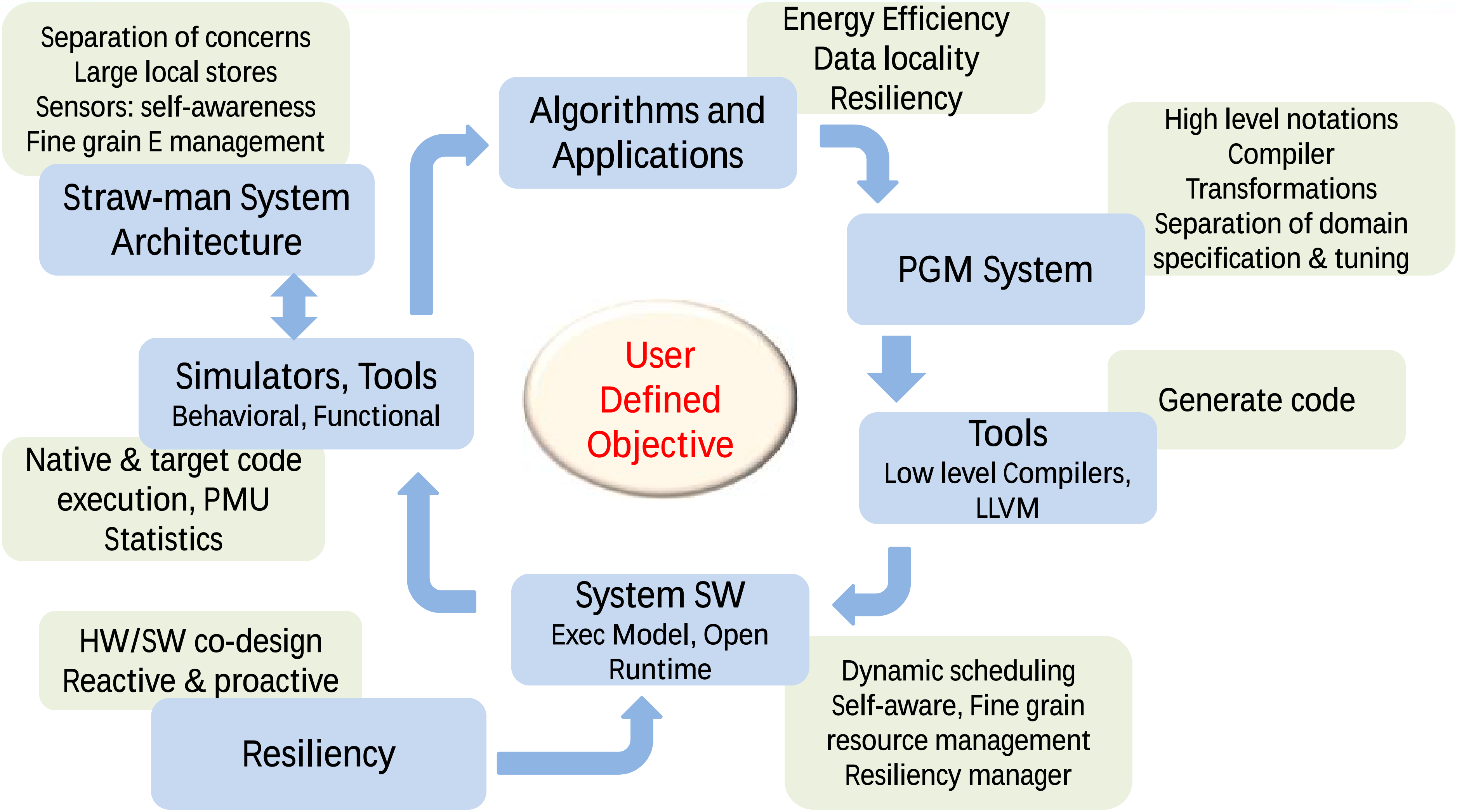
Linux

Bare Metal Shim

Product Platform

Prototype(s)

Software Components Put Together for Evaluation



Separation of concerns
Large local stores
Sensors: self-awareness
Fine grain E management

Straw-man System Architecture

Algorithms and Applications

Energy Efficiency
Data locality
Resiliency

PGM System

High level notations
Compiler
Transformations
Separation of domain specification & tuning

User Defined Objective

Simulators, Tools
Behavioral, Functional

Native & target code execution, PMU Statistics

Tools
Low level Compilers, LLVM

Generate code

System SW
Exec Model, Open Runtime

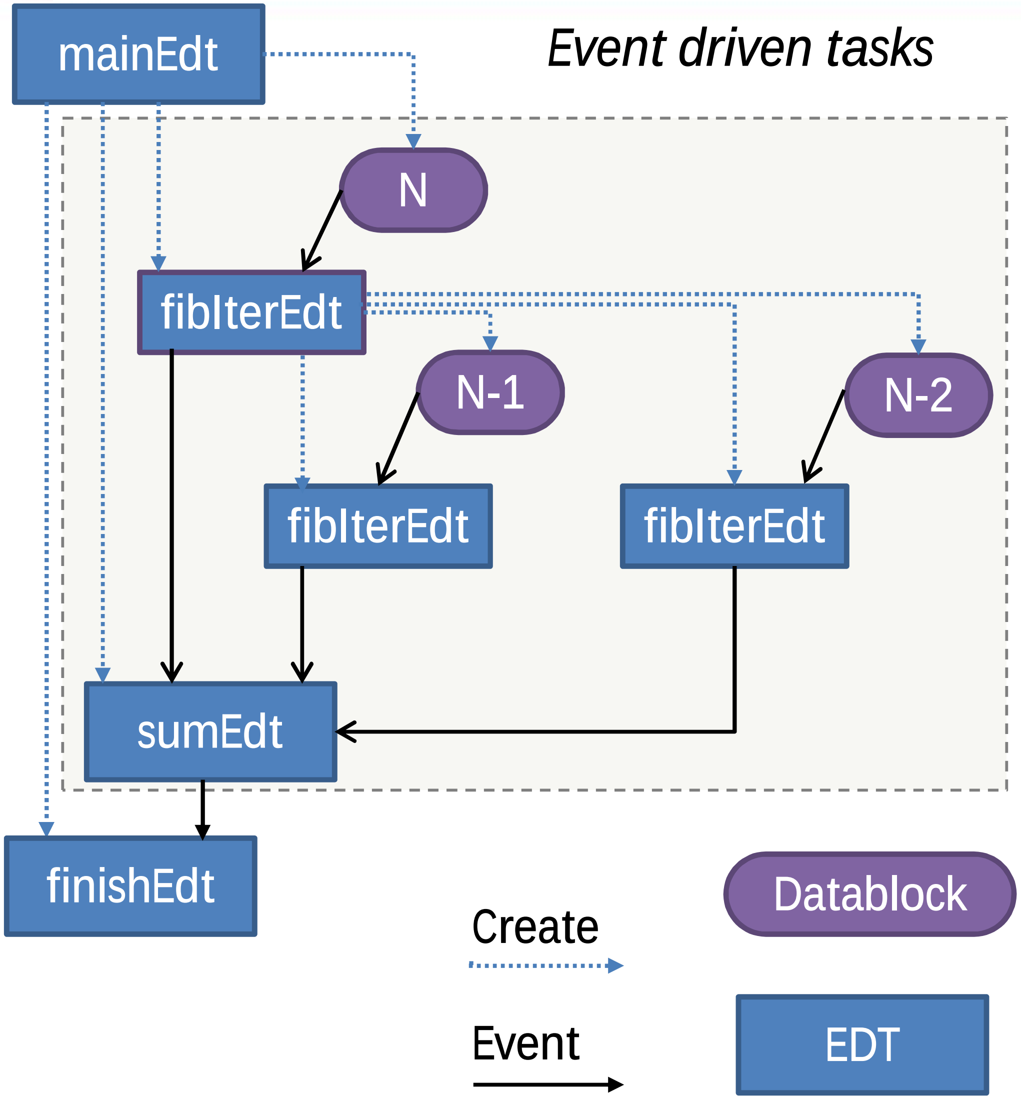
Dynamic scheduling
Self-aware, Fine grain resource management
Resiliency manager

HW/SW co-design
Reactive & proactive

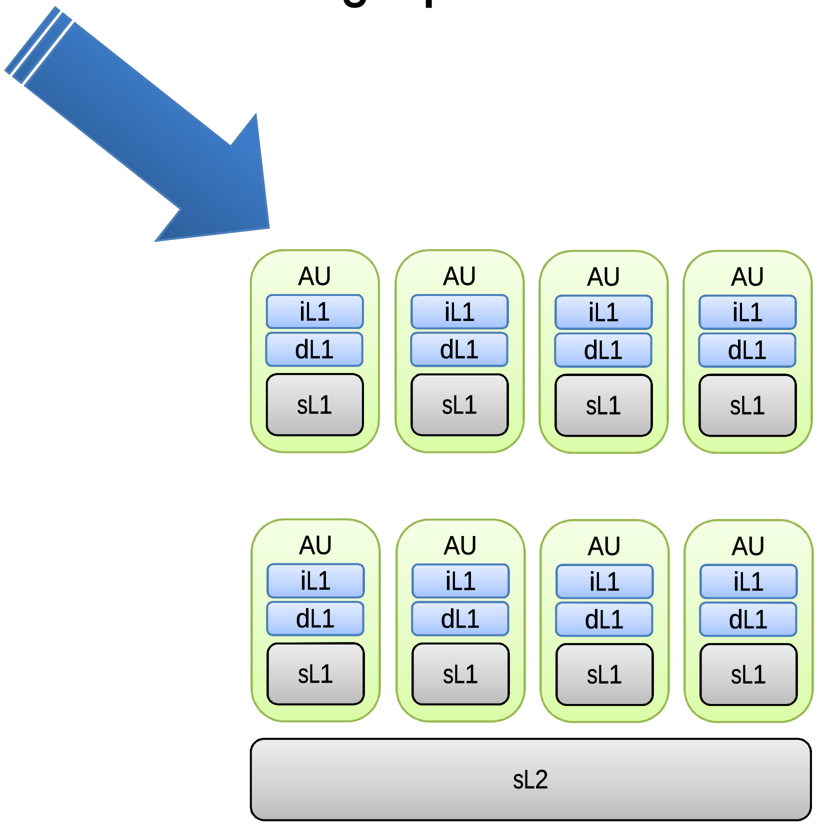
Resiliency

Dataflow-inspired Programming Model

Event driven tasks



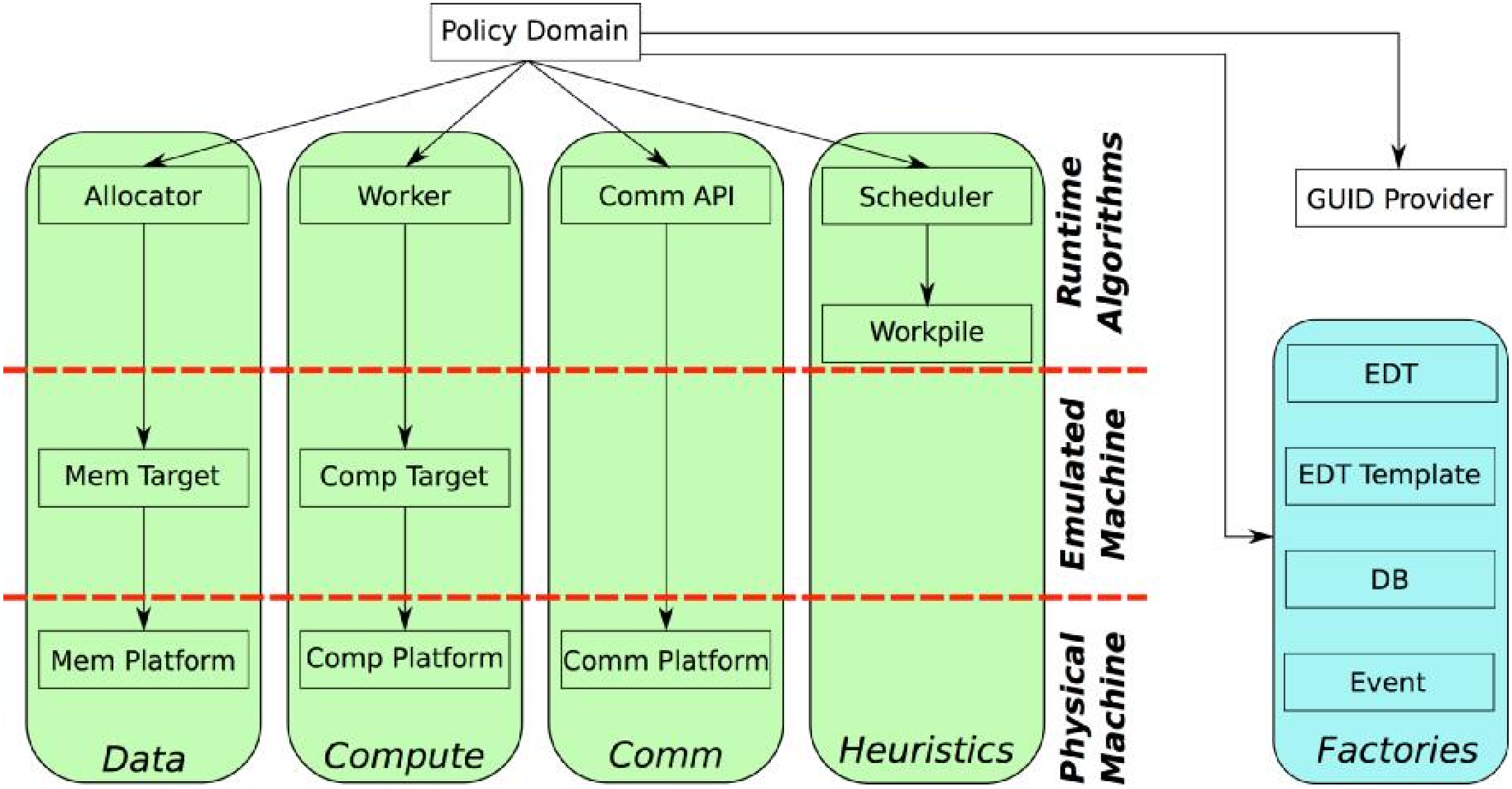
Runtime maps the constructed data-flow graph to architecture



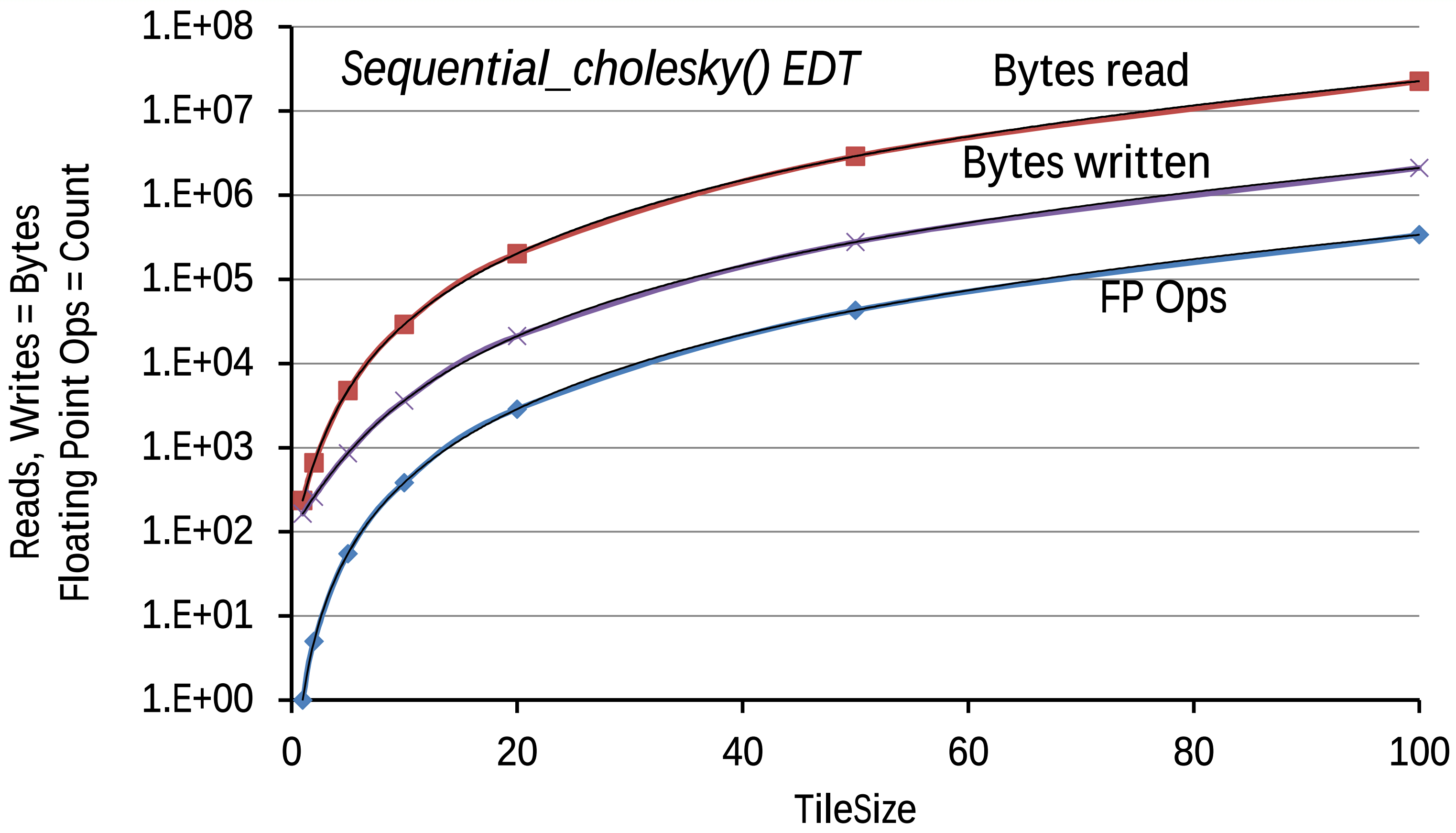
Open Community Runtime—Research Platform

- A research platform to evaluate revolutionary concepts
  - Event driven programming model
  - Introspection based resource management
  - Self-awareness
  - Resiliency
- Provides framework for future research
- Provides a reference implementation
- Provides runtime statistics for evaluation
  - Energy consumption
  - Data movement and computation

OCR modules


Policy Domain
Allocator
Worker
Comm API
Scheduler
GUID Provider
Workpile
Runtime Algorithms
EDT
Mem Target
Comp Target
Emulated Machine
EDT Template
DB
Mem Platform
Comp Platform
Comm Platform
Event
Physical Machine
Data
Compute
Comm
Heuristics
Factories

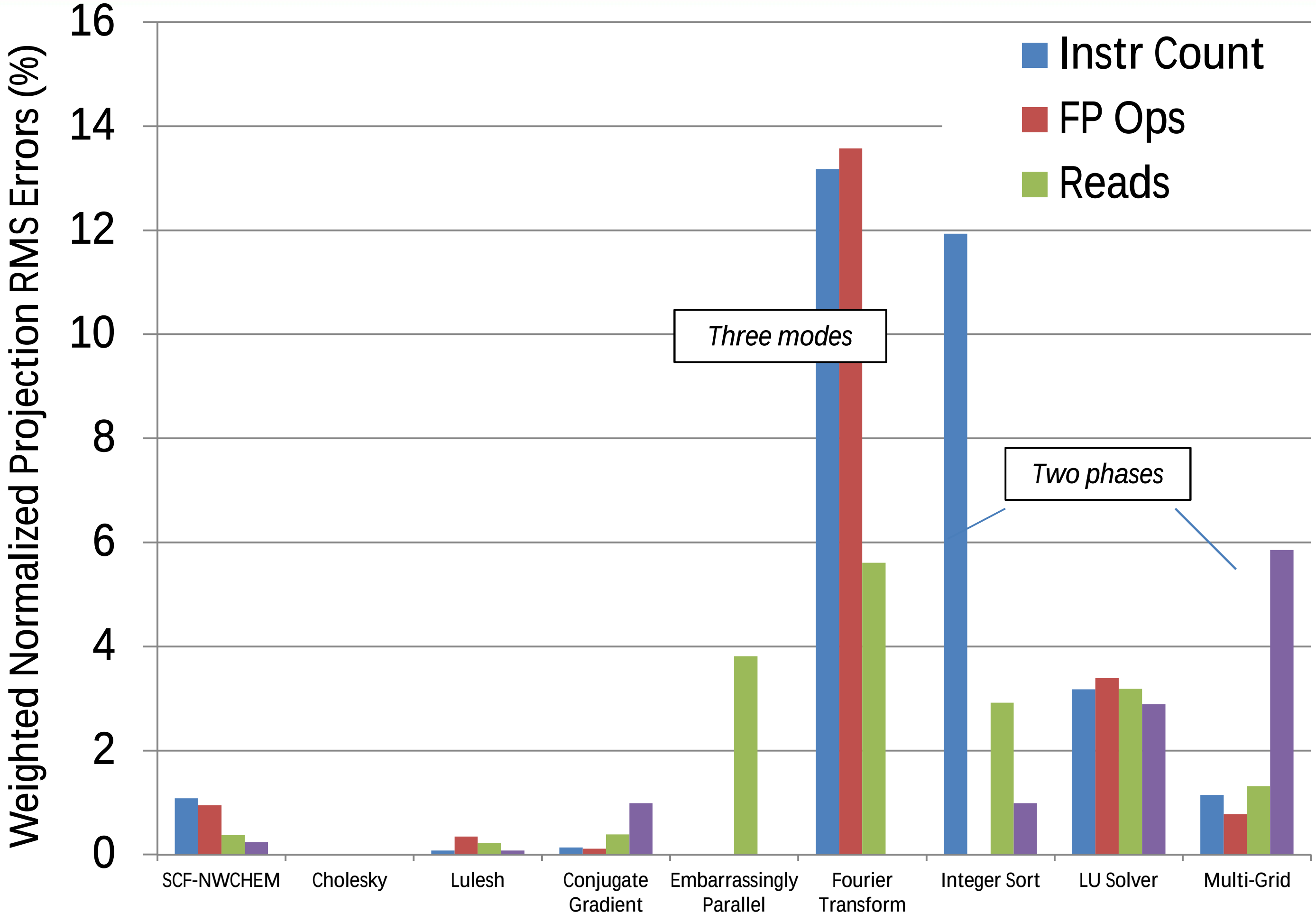
EDT Profiling using OCR



- Runtime statistics to evaluate benefits of the new approach

Introspection-based Projection

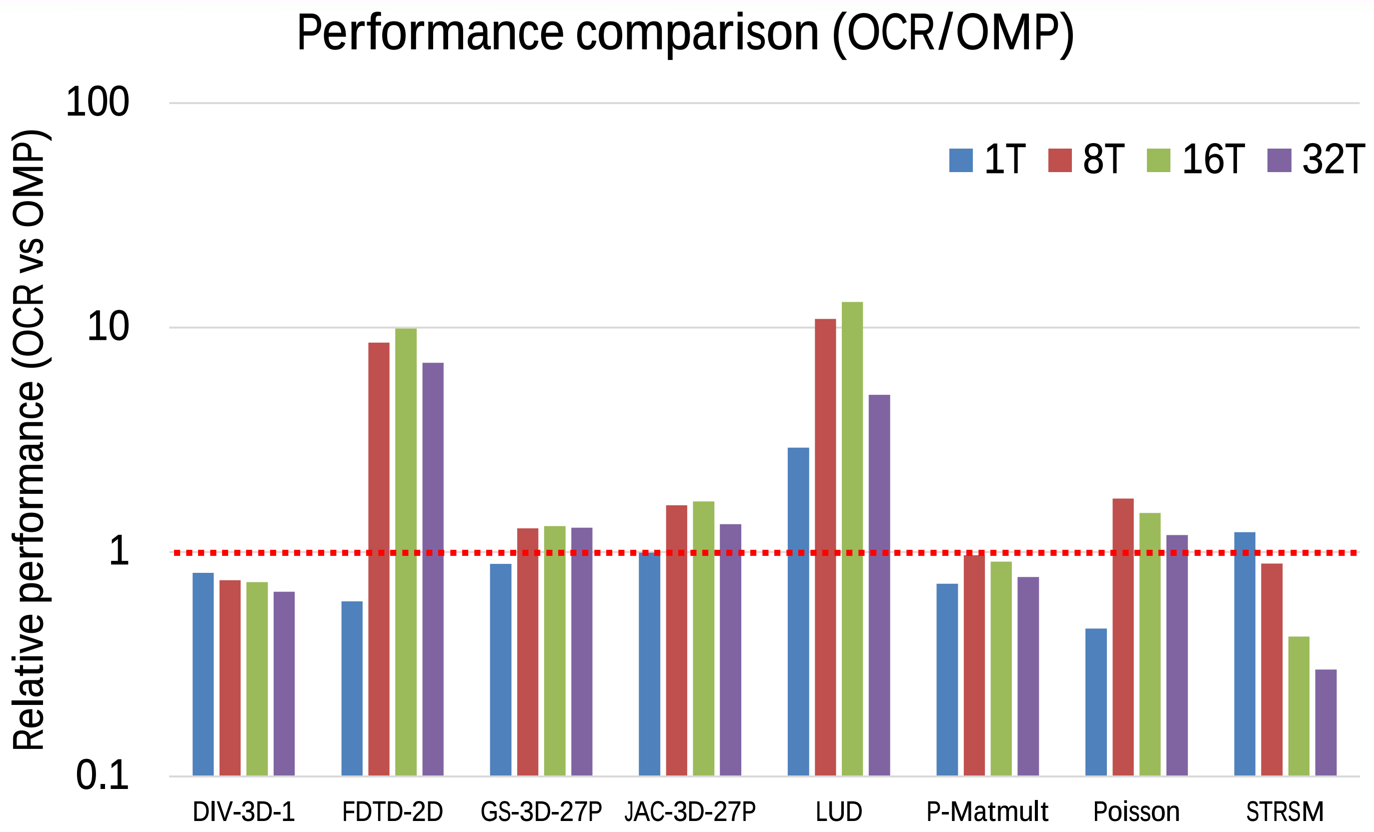
Lower is better



- Introspection based resource management looks promising

Acknowledgment: Chih-Chieh Yang, Adam Smith (NAS), Roger Golliver (LULESH), Jamie Arteaga (SCF)

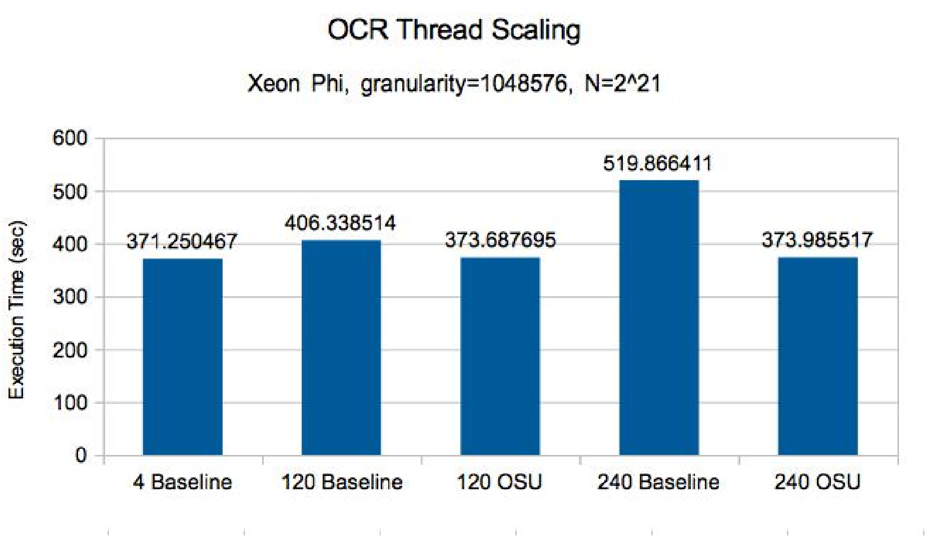
OCR on 32-Thread SNB (Reservoir)

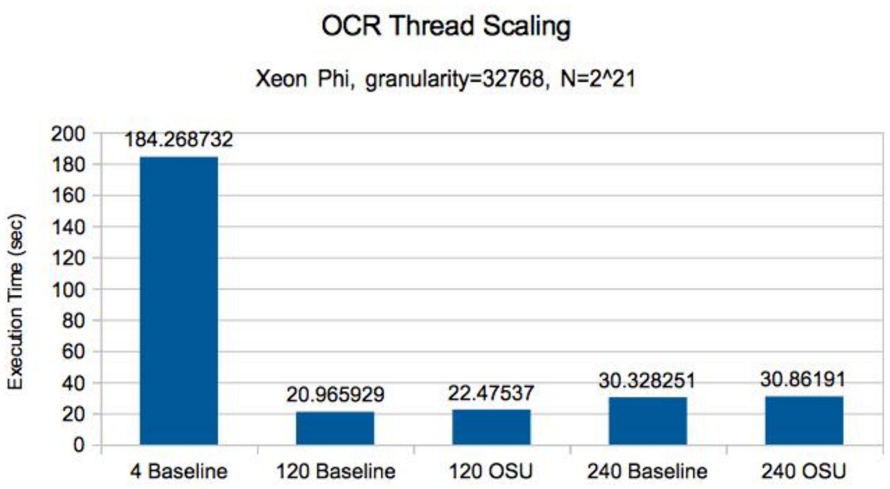


- Incomplete OCR implementation (no data blocks)
- Simple work-stealing scheduler
- Still… OCR is comparable or better than OMP in several benchmarks

(Shows a few benchmarks, for details please contact Reservoir Labs)

Thread Scaling with OCR (OR State University)





- Large data-blocks limit performance gains
- OSU scheduler (with back-off) increases performance due to fewer work-stealing attempts (less work present)

- Small data-blocks provide better speedup
- Runtime overhead seems to manifest itself

Applications Evaluated on TG Stack with OCR

- Hand-written OCR:
  - Cholesky (ETI)
  - CoMD (UCSD)
  - FFT (Oregon State)
  - HPCG (Oregon State and, separately, UCSD)
  - Lulesh 1.x (Roger Golliver)
  - SAR (Roger Golliver)
  - Stream (Oregon State)
- Written in CnC (Nick Vrvilo):
  - Smith-Waterman
  - Cholesky
- Written in HTA (UIUC):
  - Subset of NAS benchmarks (CG, FFT, Integer sort, LU decomposition, Multigrid solver) (UIUC)
- Several converted from RStream (Reservoir)

Summary

- TG SW Stack established
- Supports evolutionary & revolutionary approaches
- Open community runtime makes progress
- Results from applications look promising