X-Stack, Traleika Glacier (XSTG)

Three-year project results and products, including Intel, Rice, UCSD, PNNL, UIUC, Reservoir Labs, & Eqware

Shekhar Borkar, Josh Fryman

&

The TG Team

This work funded in part by US Government Contracts
No.# 7078611, 7216501 and B608115.

Outline

Overview of extreme-scale programs

Architecture & software stack

Results of Tools and Applications

Community Responses

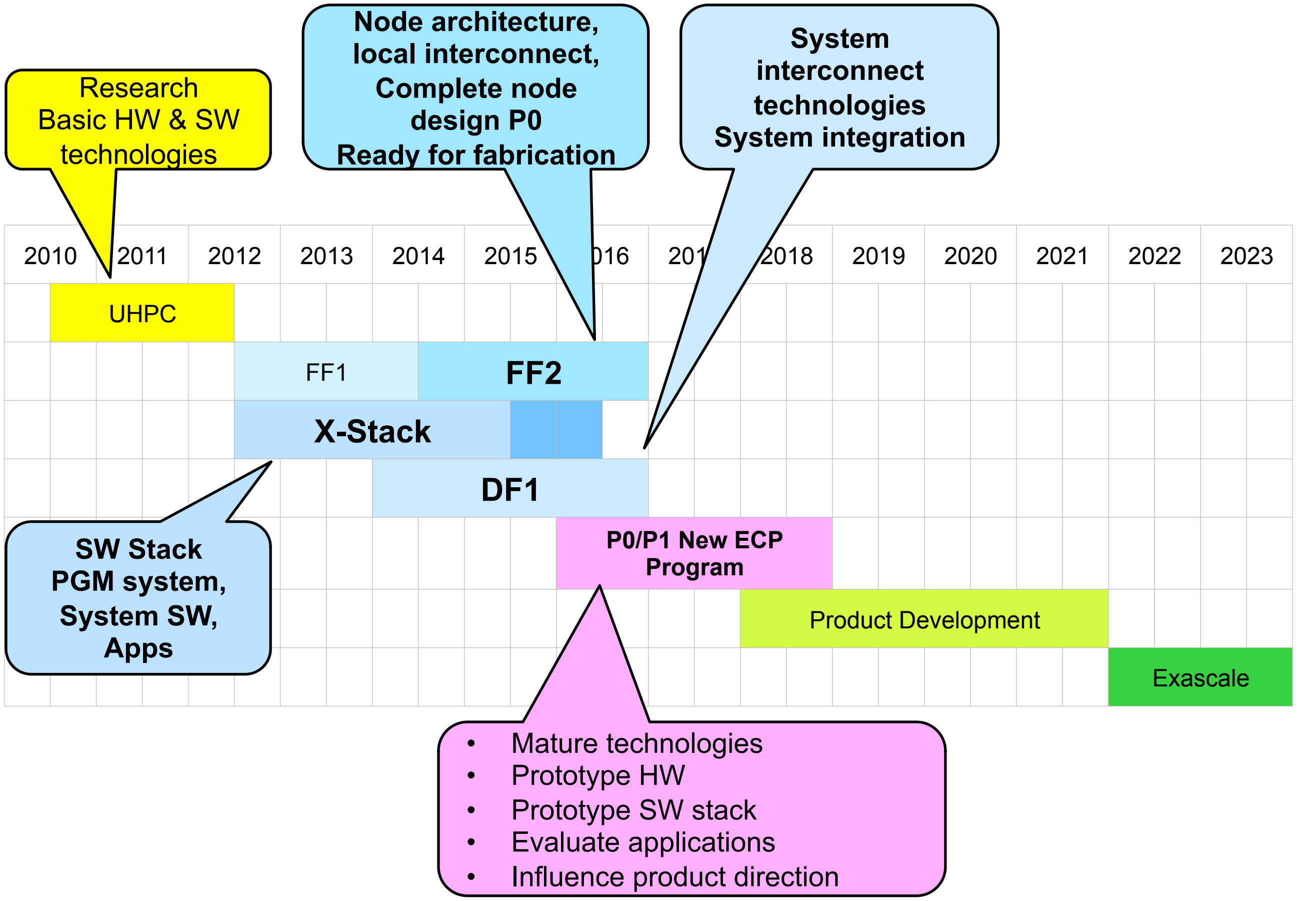
Report Card

Future Work

Top Extreme-scale Challenges

1. **System Power & Energy (Exascale in 20 MW)**
2. **New, efficient, memory subsystem (BW, latency, capacity, cost)**
3. **Extreme parallelism O(Billion) (productivity, legacy)**
4. **New execution model (introspection, event-driven)**
5. **Resiliency to provide system reliability (>> 6 days)**
6. **$ Cost and affordability by enhancing system-efficiency**

Extreme-Scale Programs for Ubiquity



Research
Basic HW & SW
technologies

Node architecture,
local interconnect,
Complete node
design P0
Ready for fabrication

System
interconnect
technologies
System integration

2010 2011 2012 2013 2014 2015 016 201 2018 2019 2020 2021 2022 2023

UHPC

FF1

FF2

X-Stack

DF1

P0/P1 New ECP
Program

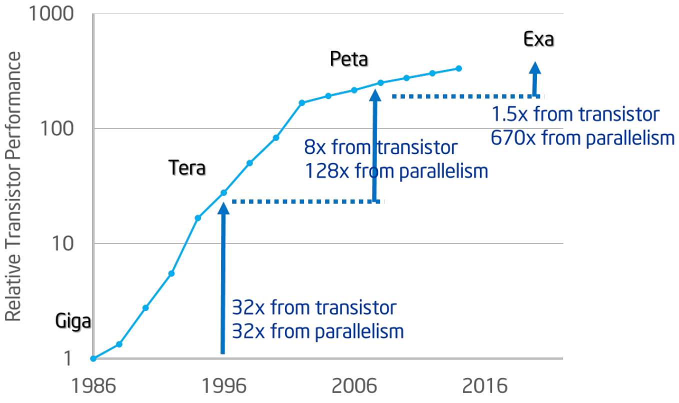
Product Development

Exascale

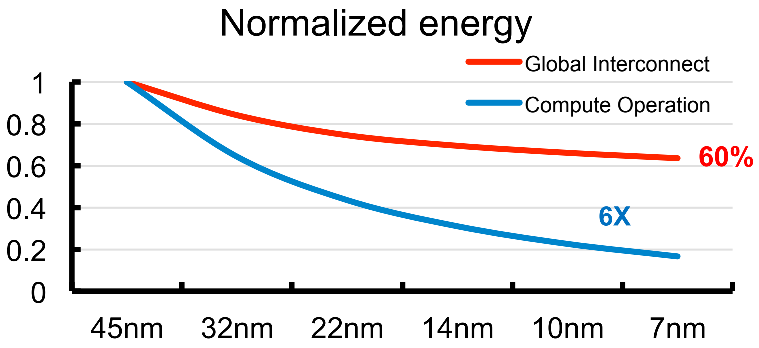
SW Stack
PGM system,
System SW,
Apps

- Mature technologies
- Prototype HW
- Prototype SW stack
- Evaluate applications
- Influence product direction

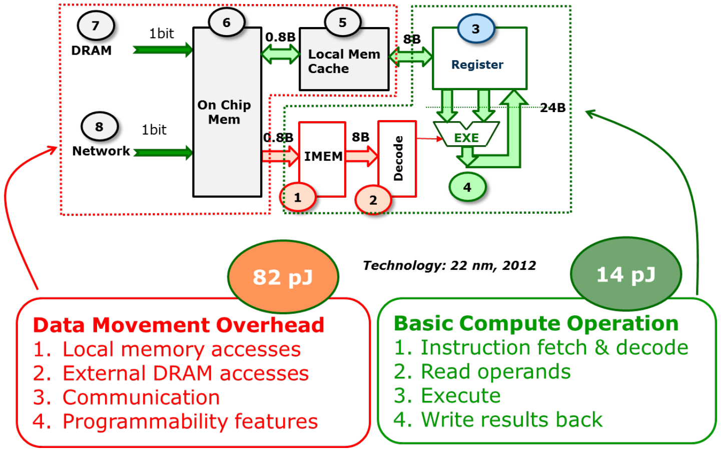
Compute vs Data Movement



Performance from parallelism


Normalized energy

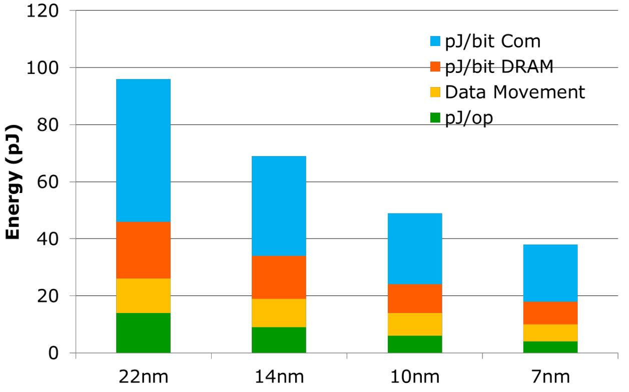
Basic compute loop



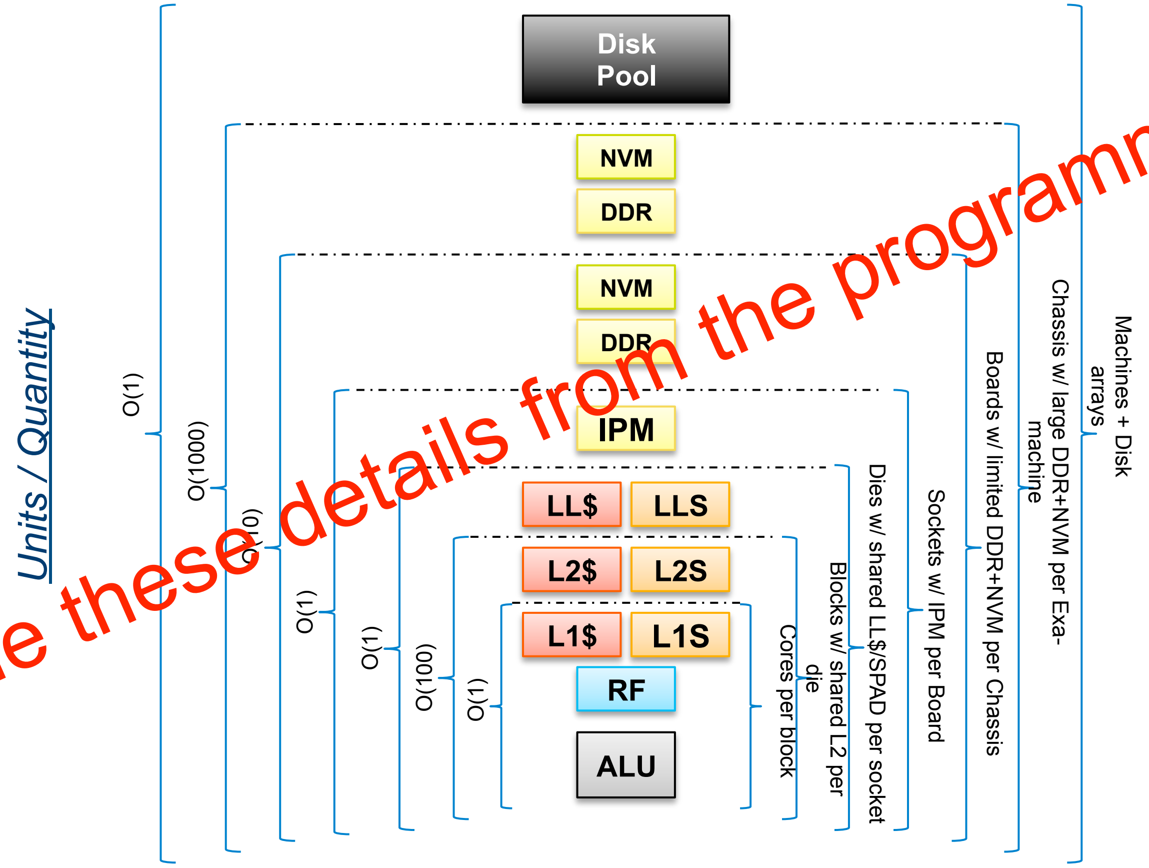
Technology: 22 nm, 2012

Data Movement Overhead
1. Local memory accesses
2. External DRAM accesses
3. Communication
4. Programmability features

Basic Compute Operation
1. Instruction fetch & decode
2. Read operands
3. Execute
4. Write results back



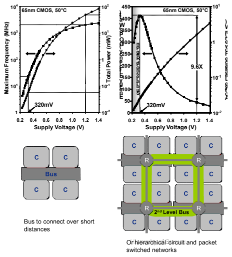
Memory Hierarchy



Hide these details from the programmer….

Addressing Challenges

1. Power & Energy



Bus to connect over short distances

Or hierarchical circuit and packet switched networks

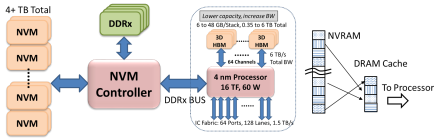
Vdd scaling

NTV operation

Fine grain power management

Hierarchical, heterogeneous IC fabric

2. Memory subsystem

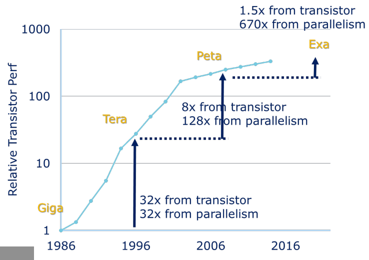

4+ TB Total
NVM
NVM
NVM
NVM
DDRx
NVM Controller
DDRx BUS
Lower capacity, increase BW
6 to 48 GB/Stack, 0.35 to 6 TB Total
3D HBM
3D HBM
64 Channels
6 TB/s Total BW
4 nm Processor 16 TF, 60 W
IC Fabric: 64 Ports, 128 Lanes, 1.5 TB/s
NVRAM
DRAM Cache
To Processor

Disproportionately large local memories

Incorporate NVM for cost

Hierarchy, global addressability

3. Programmability



O(B) threads

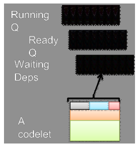
Data locality

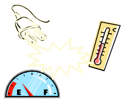
Separation of concerns

Legacy compatibility

Productivity

4. Execution model





Asynchronous

Event-driven tasks (tiny threads)

Dynamic scheduling of threads

Introspective resource management

(Self-aware through observations)

Dynamic system optimization

5. Resiliency

Error detection
Fault isolation
Fault confinement
Reconfiguration
Recovery & Adapt

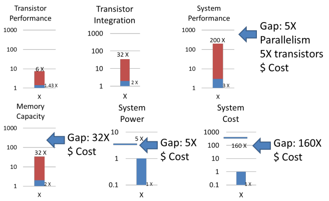
First, understand faults

Detection in HW with sensors

Reactive and proactive measures

6. Cost & affordability

1000X performance in 10 years Apply Dennard scaling theory



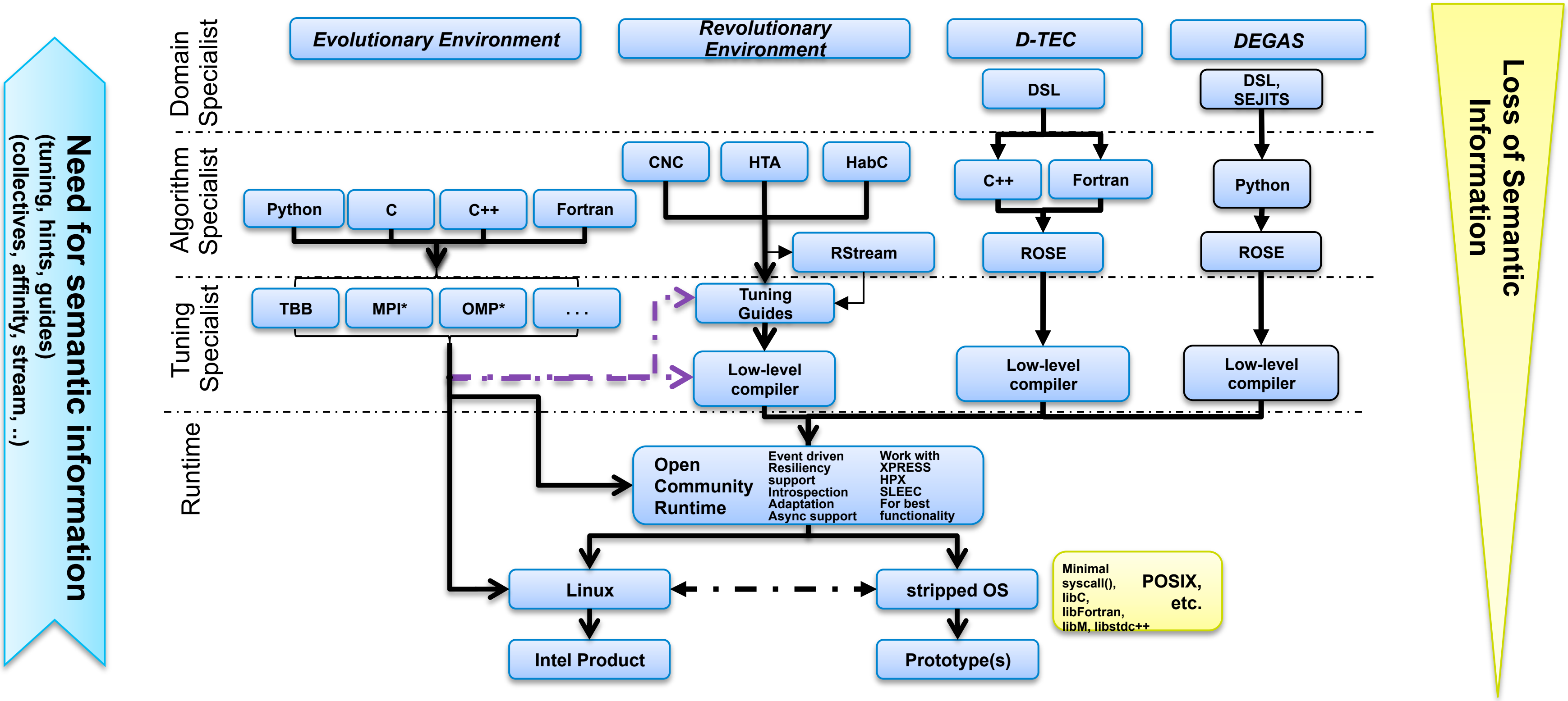
Improve Perf/Transistor

New, efficient architecture

Data locality

Insert NVM in the hierarchy

Software Ecosystem: Support today, find tomorrow



Major Points

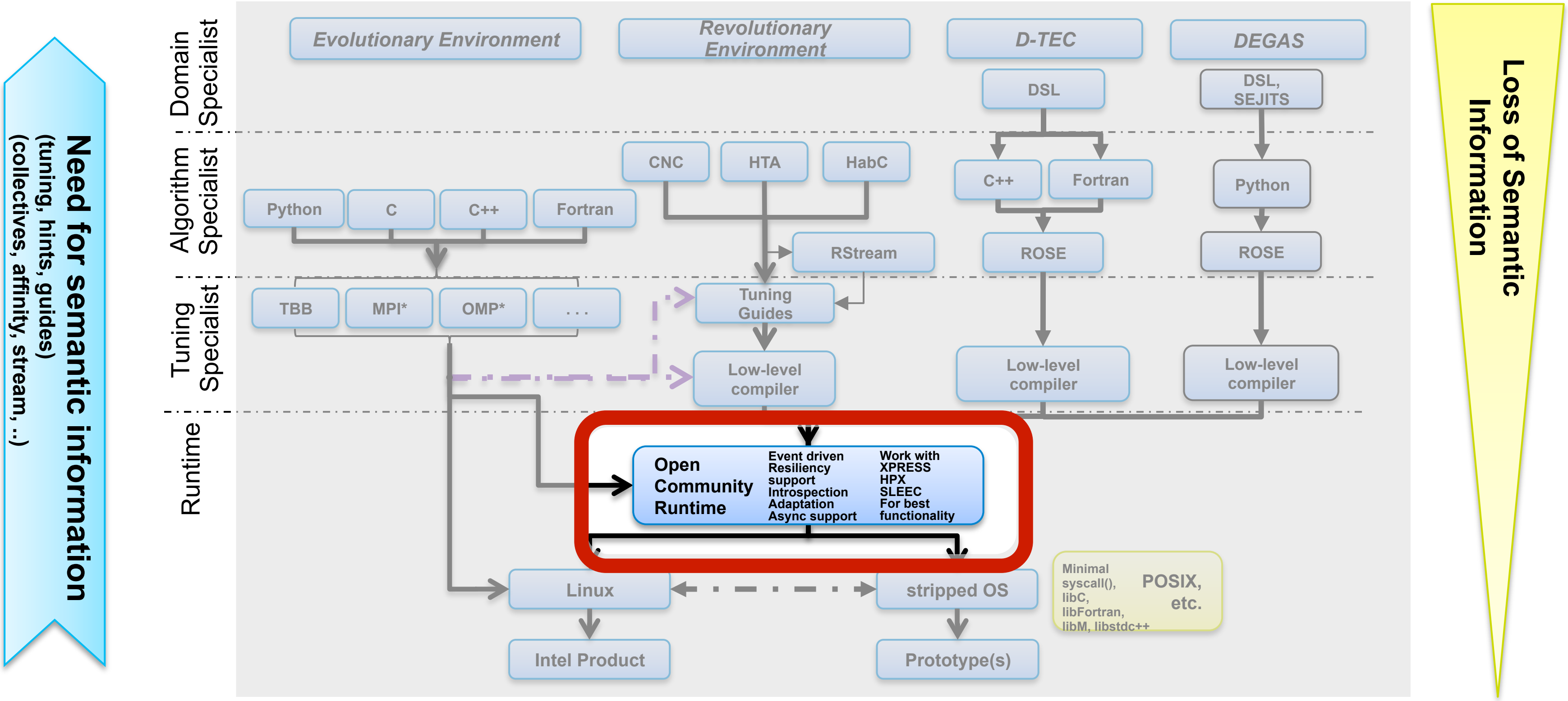
- All work done under XSTG is available publicly, open-source BSD licensed
  - http://xstack.exascale-tech.com
  - Exceptions:
    - PNNL has not yet released their OCR implementation at this time
    - R-Stream is closed-source with government use rights
- XSTG project has created significant reference implementations
  - Formal specification for task-based execution model
  - Reference implementation that even in immature form exceeds expectations
  - Productivity tools, compilers, profilers, visualizers, re-factored applications, libraries of kernels and examples, simulators, and tutorials
  - **Application workshops every 6 months with the community**

XSTG has transitioned from Research to Technology Maturation

Embargoed Data

- The following sections contain results that have not yet been published
- Please embargo sharing and distribution of this content until publication efforts have concluded by each team
- All results are identified by team – contact each for further details

Task-Execution Runtime (Intel, Rice)



Need for semantic information
(tuning, hints, guides)
(collectives, affinity, stream, ..)

Domain Specialist

Evolutionary Environment

Revolutionary Environment

D-TEC

DSL

DEGAS

DSL, SEJITS

Algorithm Specialist

CNC

HTA

HabC

Python

C

C++

Fortran

C++

Fortran

Python

RStream

ROSE

ROSE

Tuning Specialist

TBB

MPI*

OMP*

. . .

Tuning Guides

Low-level compiler

Low-level compiler

Low-level compiler

Runtime

Open Community Runtime

Event driven
Resiliency support
Introspection
Adaptation
Async support

Work with
XPRESS
HPX
SLEEC
For best functionality

Linux

stripped OS

Minimal syscall(), libC, libFortran, libM, libstdc++

POSIX, etc.

Intel Product

Prototype(s)

Loss of Semantic Information

Open Community Runtime (OCR)

Multi-party collaboration

- Intel, Rice, UIUC, UCSD, PNNL, Reservoir Labs

- Provide effective abstraction for diverse hardware (hetero-ISA ready)
- Typify future task-based execution models
- Handle large-scale parallelism efficiently and dynamically
- Provide user-perspective application-transparent resiliency
- Maintain a separation of concerns (application/scheduling/resources)
- Open source (encourage collaboration) http://xstack.exascale-tech.com
  - *OCR is X-Stack Traleika Glacier project’s implementation for this revolutionary run-time prototype*
  - *FFWD-2 is extending OCR with legacy support, re-factored applications, and re-factoring guides, templates, and tools*

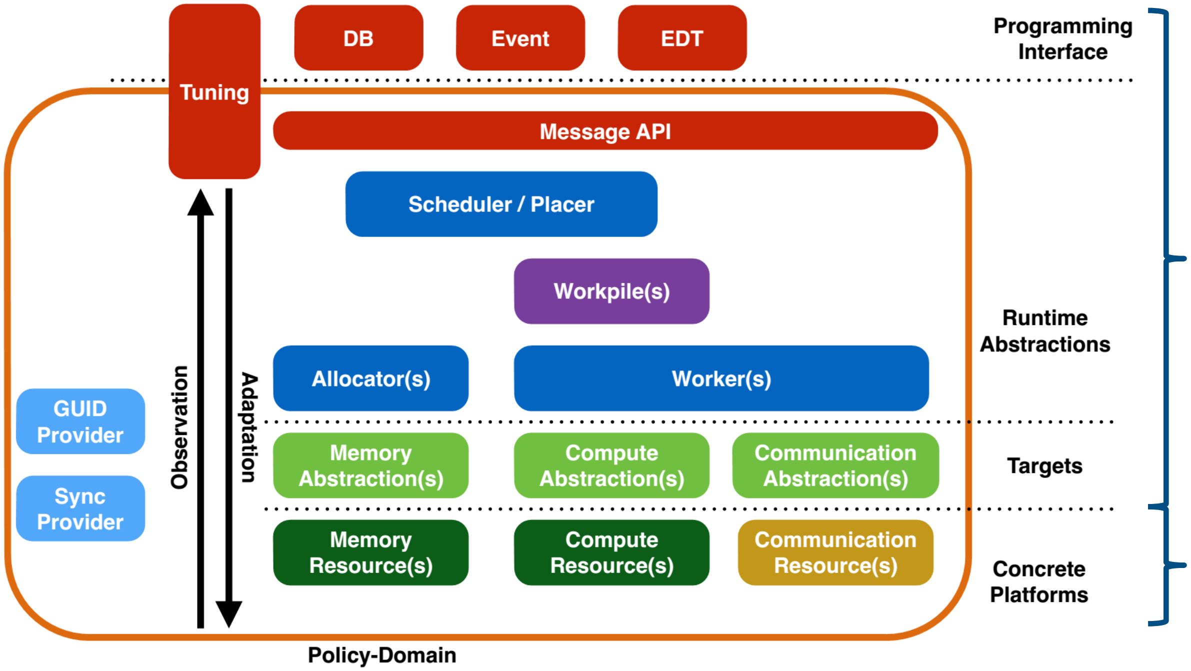
Runtime Design Principles of OCR

Modular

Extensible

Adaptable

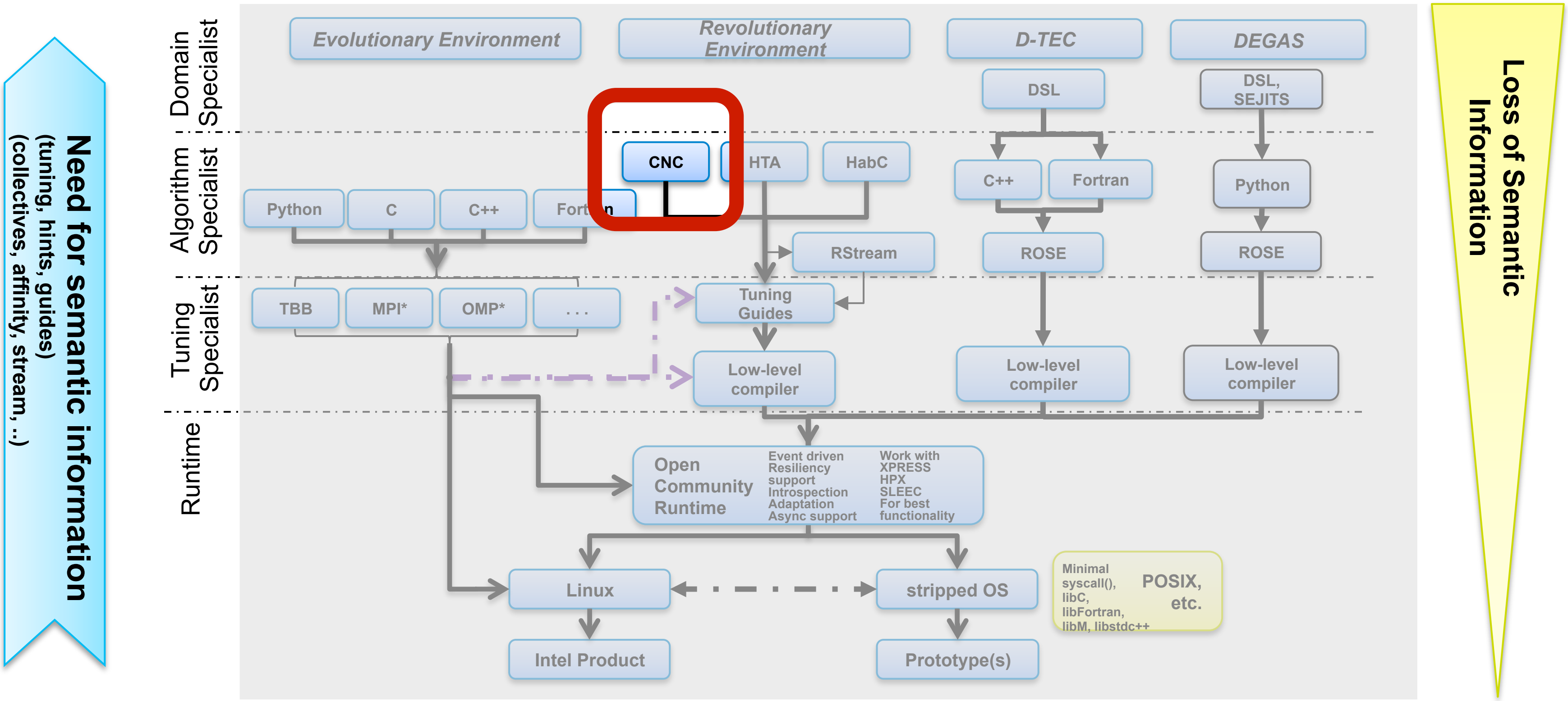
Tunable



Generic, Adaptive

Platform Specific

Productivity: Concurrent Collections (Rice, Intel)



Need for semantic information
(tuning, hints, guides)
(collectives, affinity, stream, ..)

Domain Specialist

Algorithm Specialist

Tuning Specialist

Runtime

Evolutionary Environment

Revolutionary Environment

D-TEC

DEGAS

DSL

DSL, SEJITS

CNC

HTA

HabC

C++

Fortran

Python

Python

C

C++

Fortran

RStream

ROSE

ROSE

TBB

MPI*

OMP*

. . .

Tuning Guides

Low-level compiler

Low-level compiler

Low-level compiler

Open Community Runtime

Event driven
Resiliency support
Introspection
Adaptation
Async support

Work with XPRESS
HPX
SLEEC
For best functionality

Linux

stripped OS

Minimal syscall(), libC, libFortran, libM, libstdc++

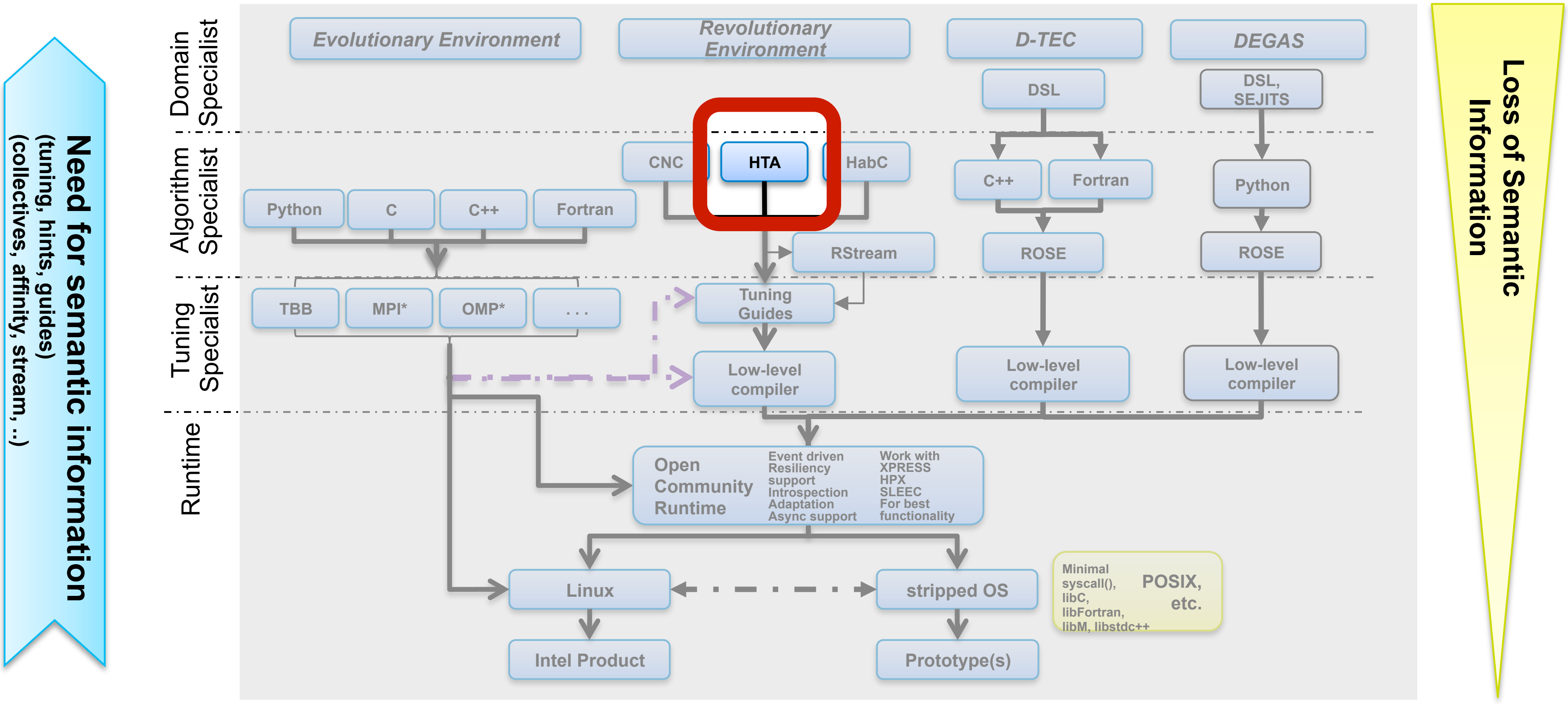
POSIX, etc.

Intel Product

Prototype(s)

Loss of Semantic Information

Productivity: Hierarchically Tiled Arrays (UIUC)



Need for semantic information
(tuning, hints, guides)
(collectives, affinity, stream, ..)

Domain Specialist
Algorithm Specialist
Tuning Specialist
Runtime

Evolutionary Environment
Revolutionary Environment
D-TEC
DEGAS

DSL
DSL, SEJITS
CNC
HTA
HabC
C++
Fortran
Python
Python
C
C++
Fortran
RStream
ROSE
ROSE
TBB
MPI*
OMP*
. . .
Tuning Guides
Low-level compiler
Low-level compiler
Low-level compiler

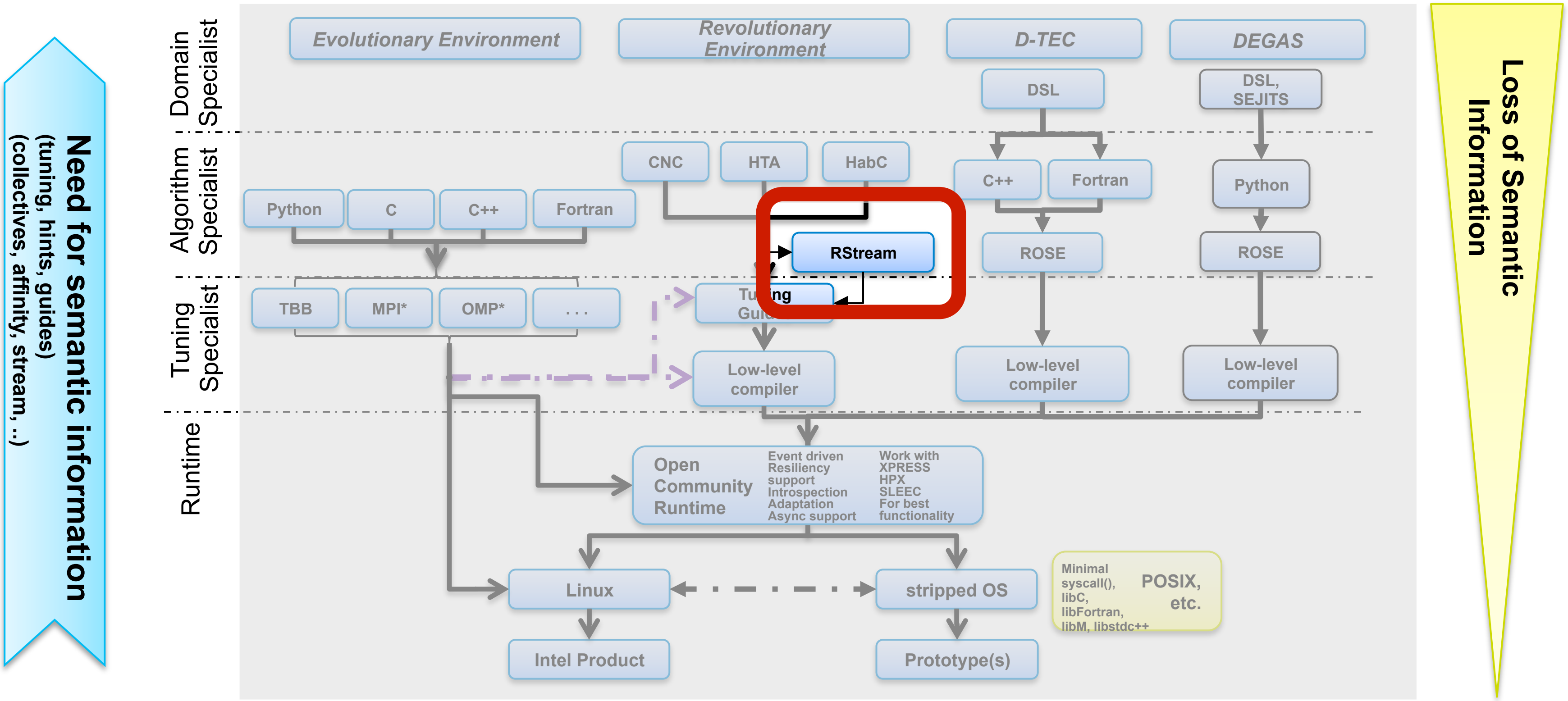
Open Community Runtime
Event driven
Resiliency support
Introspection
Adaptation
Async support
Work with
XPRESS
HPX
SLEEC
For best functionality

Linux
stripped OS
Intel Product
Prototype(s)

Minimal syscall(), libC, libFortran, libM, libstdc++
POSIX, etc.

Loss of Semantic Information

Productivity: R-Stream (Reservoir Labs)

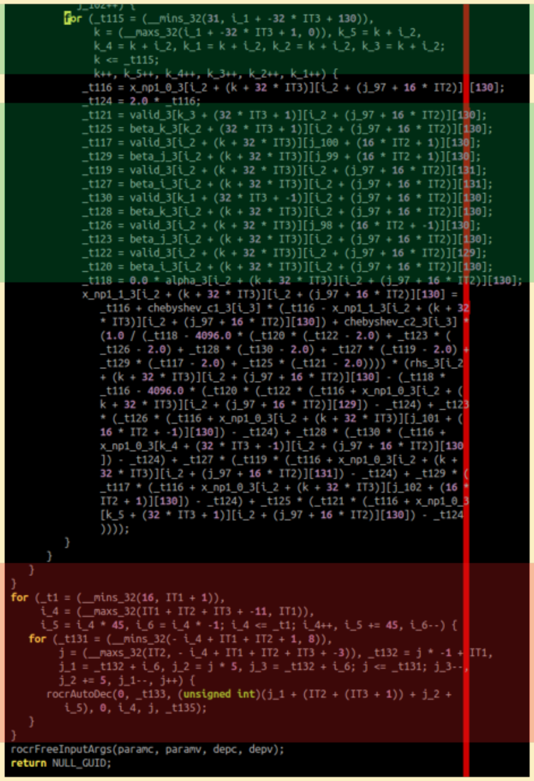


R-Stream: Polymorphic Optimizing Translator

```
for(k=1;k<NPAD-1;k++){
    for(j=1;j<NPAD-1;j++){
        for(i=1;i<NPAD-1;i++){
            x_np1_1[k][j][i] = x_np1_0[k][j][i]
                               c1*(x_np1_0[k][j][i]
                               c2*Dinv_ijk()* (rhs[
}}}}}
```

Input C code

- Automatic parallelization and locality optimization
- Automatic explicit communications generation and explicit datablock management
- Automatic formation of OCR EDT programs (and other runtimes)
- Dynamic creation and management of dependences



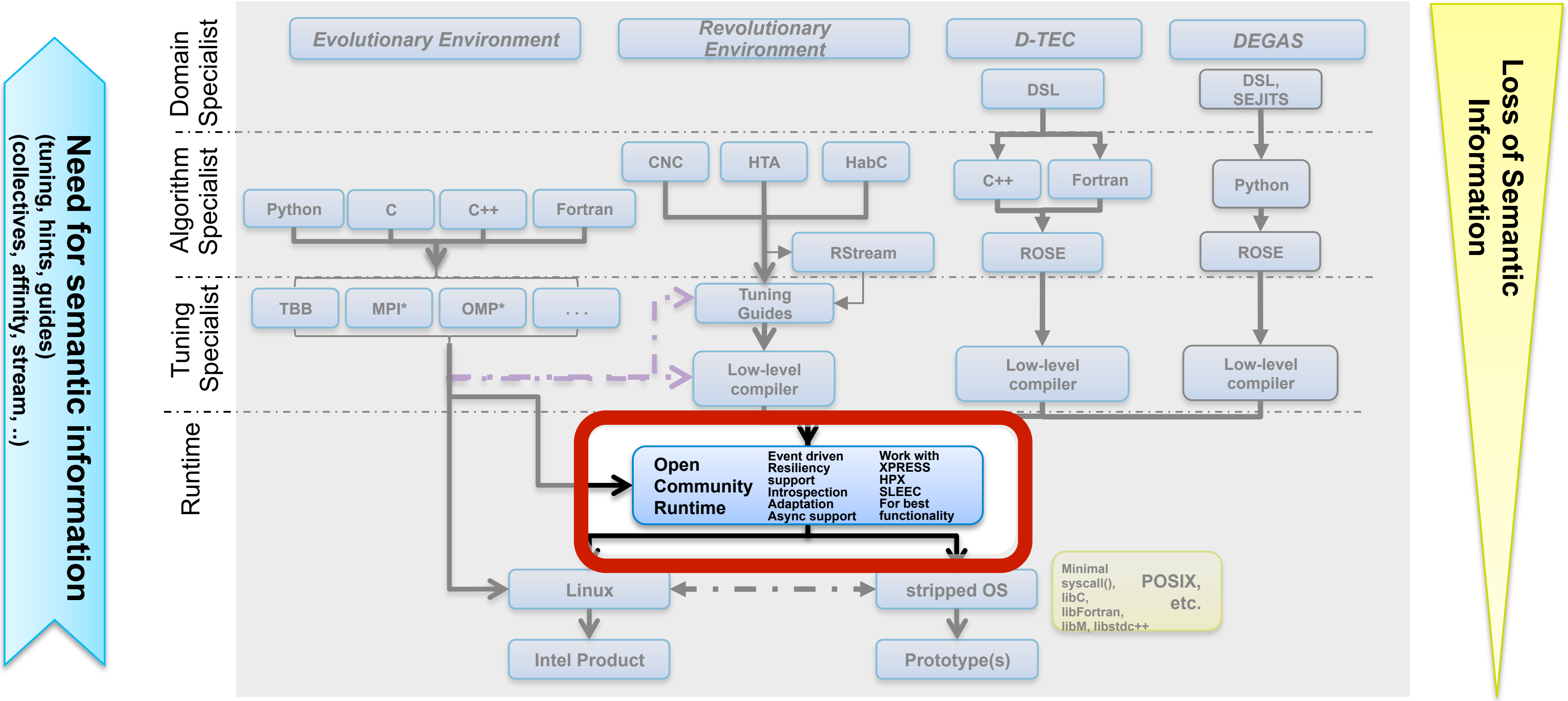
```
              j_102++) {
          for (_t115 = (__mins_32(31, i_1 + -32 * IT3 + 130)),
               k = (__maxs_32(i_1 + -32 * IT3 + 1, 0)), k_5 = k + i_2,
               k_4 = k + i_2, k_1 = k + i_2, k_2 = k + i_2, k_3 = k + i_2;
               k <= _t115;
               k++, k_5++, k_4++, k_3++, k_2++, k_1++) {
            _t116 = x_np1_0_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)] [130];
            _t124 = 2.0 * _t116;
            _t121 = valid_3[k_3 + (32 * IT3 + 1)][i_2 + (j_97 + 16 * IT2)][130];
            _t125 = beta_k_3[k_2 + (32 * IT3 + 1)][i_2 + (j_97 + 16 * IT2)][130];
            _t117 = valid_3[i_2 + (k + 32 * IT3)][j_100 + (16 * IT2 + 1)][130];
            _t129 = beta_j_3[i_2 + (k + 32 * IT3)][j_99 + (16 * IT2 + 1)][130];
            _t119 = valid_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][131];
            _t127 = beta_i_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][131];
            _t130 = valid_3[k_1 + (32 * IT3 + -1)][i_2 + (j_97 + 16 * IT2)][130];
            _t128 = beta_k_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][130];
            _t126 = valid_3[i_2 + (k + 32 * IT3)][j_98 + (16 * IT2 + -1)][130];
            _t123 = beta_j_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][130];
            _t122 = valid_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][129];
            _t120 = beta_i_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][130];
            _t118 = 0.0 * alpha_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][130];
            x_np1_1_3[i_2 + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][130] =
              _t116 + chebyshev_c1_3[i_3] * (_t116 - x_np1_1_3[i_2 + (k + 32
              * IT3)][i_2 + (j_97 + 16 * IT2)][130]) + chebyshev_c2_3[i_3] *
              (1.0 / (_t118 - 4096.0 * (_t120 * (_t122 - 2.0) + _t123 * (
              _t126 - 2.0) + _t128 * (_t130 - 2.0) + _t127 * (_t119 - 2.0) +
              _t129 * (_t117 - 2.0) + _t125 * (_t121 - 2.0)))) * (rhs_3[i_2
              + (k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][130] - (_t118 *
              _t116 - 4096.0 * (_t120 * (_t122 * (_t116 + x_np1_0_3[i_2 + (
              k + 32 * IT3)][i_2 + (j_97 + 16 * IT2)][129]) - _t124) + _t123
              * (_t126 * (_t116 + x_np1_0_3[i_2 + (k + 32 * IT3)][j_101 + (
              16 * IT2 + -1)][130]) - _t124) + _t128 * (_t130 * (_t116 +
              x_np1_0_3[k_4 + (32 * IT3 + -1)][i_2 + (j_97 + 16 * IT2)][130
              ]) - _t124) + _t127 * (_t119 * (_t116 + x_np1_0_3[i_2 + (k +
              32 * IT3)][i_2 + (j_97 + 16 * IT2)][131]) - _t124) + _t129 * (
              _t117 * (_t116 + x_np1_0_3[i_2 + (k + 32 * IT3)][j_102 + (16 *
              IT2 + 1)][130]) - _t124) + _t125 * (_t121 * (_t116 + x_np1_0_3
              [k_5 + (32 * IT3 + 1)][i_2 + (j_97 + 16 * IT2)][130]) - _t124
              ))));
          }
        }
    }
}
for (_t1 = (__mins_32(16, IT1 + 1)),
     i_4 = (__maxs_32(IT1 + IT2 + IT3 + -11, IT1)),
     i_5 = i_4 * 45, i_6 = i_4 * -1; i_4 <= _t1; i_4++, i_5 += 45, i_6--) {
   for (_t131 = (__mins_32(- i_4 + IT1 + IT2 + 1, 8)),
         j = (__maxs_32(IT2, - i_4 + IT1 + IT2 + IT3 + -3)), _t132 = j * -1 + IT1,
         j_1 = _t132 + i_6, j_2 = j * 5, j_3 = _t132 + i_6; j <= _t131; j_3--,
         j_2 += 5, j_1--, j++) {
      rocrAutoDec(0, _t133, (unsigned int)(j_1 + (IT2 + (IT3 + 1)) + j_2 +
         i_5), 0, i_4, j, _t135);
   }
}
rocrFreeInputArgs(paramc, paramv, depc, depv);
return NULL_GUID;
```

Non-trivial loop boundaries and array access functions in the core of computation block of each EDT

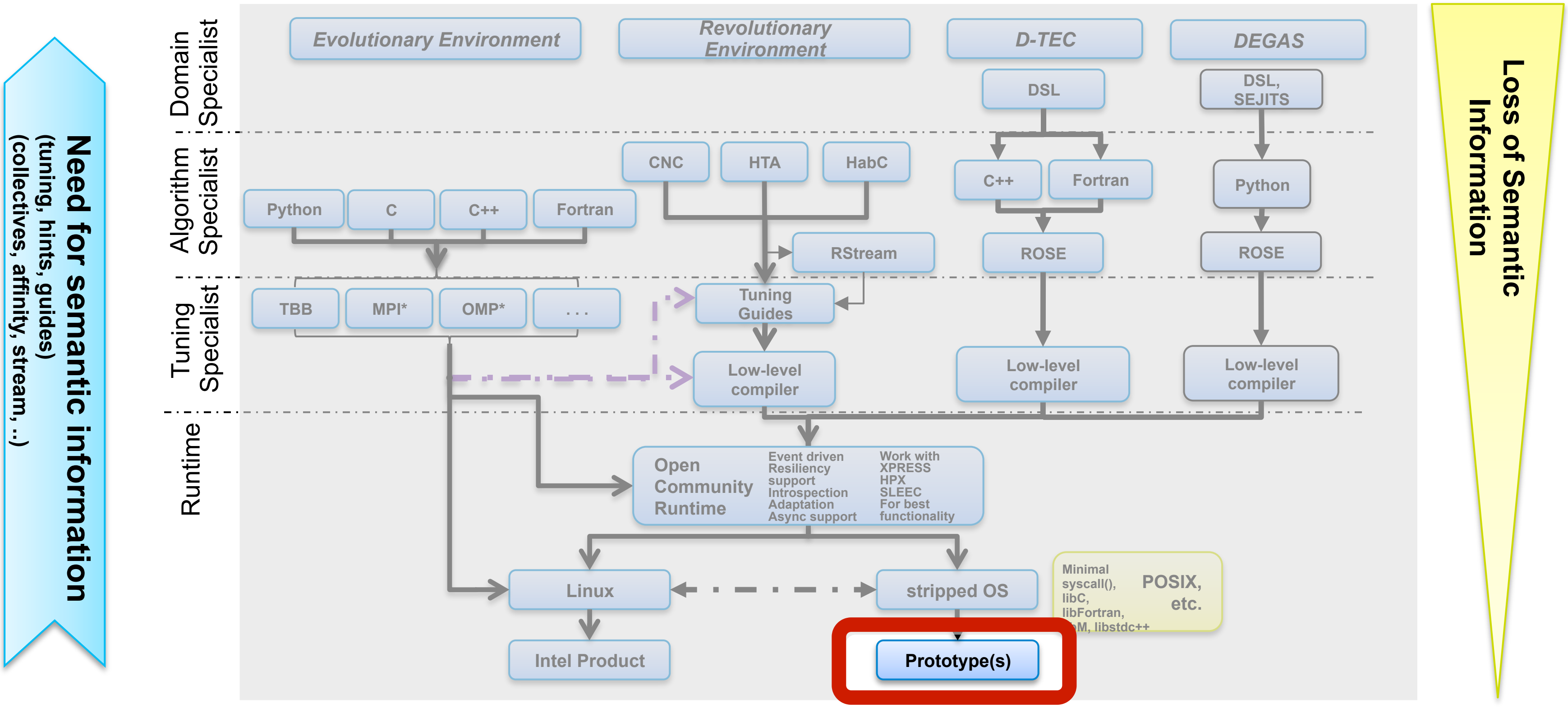
Optimized R-Stream--OCR code embeds non-trivial dependence management for dynamic creation of EDTs and datablocks

R-Stream outputted OCR code snippet

Task-Execution Runtime (PNNL)



Software Ecosystem: Support today, find tomorrow



Need for semantic information
(tuning, hints, guides)
(collectives, affinity, stream, ..)

Domain Specialist

Algorithm Specialist

Tuning Specialist

Runtime

Evolutionary Environment

Revolutionary Environment

D-TEC

DEGAS

DSL

DSL, SEJITS

CNC

HTA

HabC

Python

C

C++

Fortran

C++

Fortran

Python

RStream

ROSE

ROSE

TBB

MPI*

OMP*

. . .

Tuning Guides

Low-level compiler

Low-level compiler

Low-level compiler

Open Community Runtime

Event driven
Resiliency support
Introspection
Adaptation
Async support

Work with
XPRESS
HPX
SLEEC
For best functionality

Linux

stripped OS

Minimal syscall(), libC, libFortran, libM, libstdc++

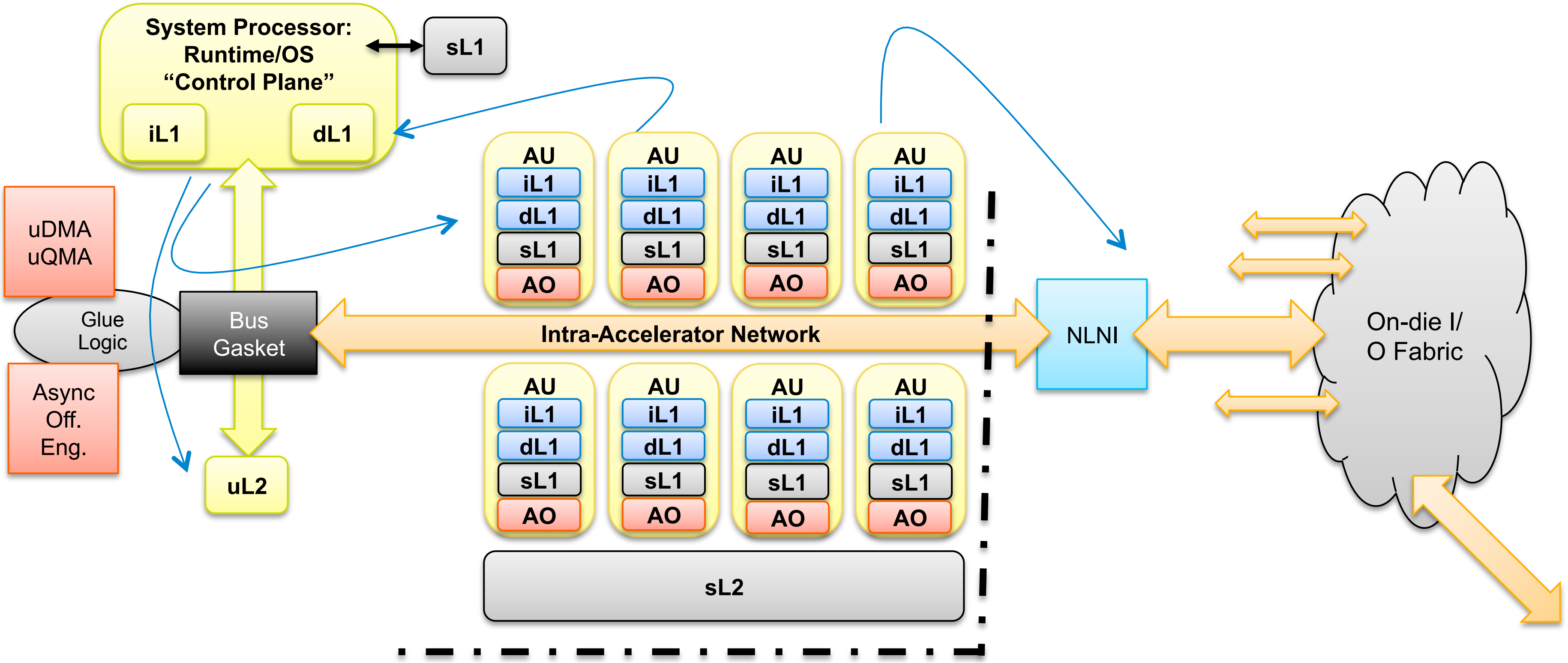
POSIX, etc.

Intel Product

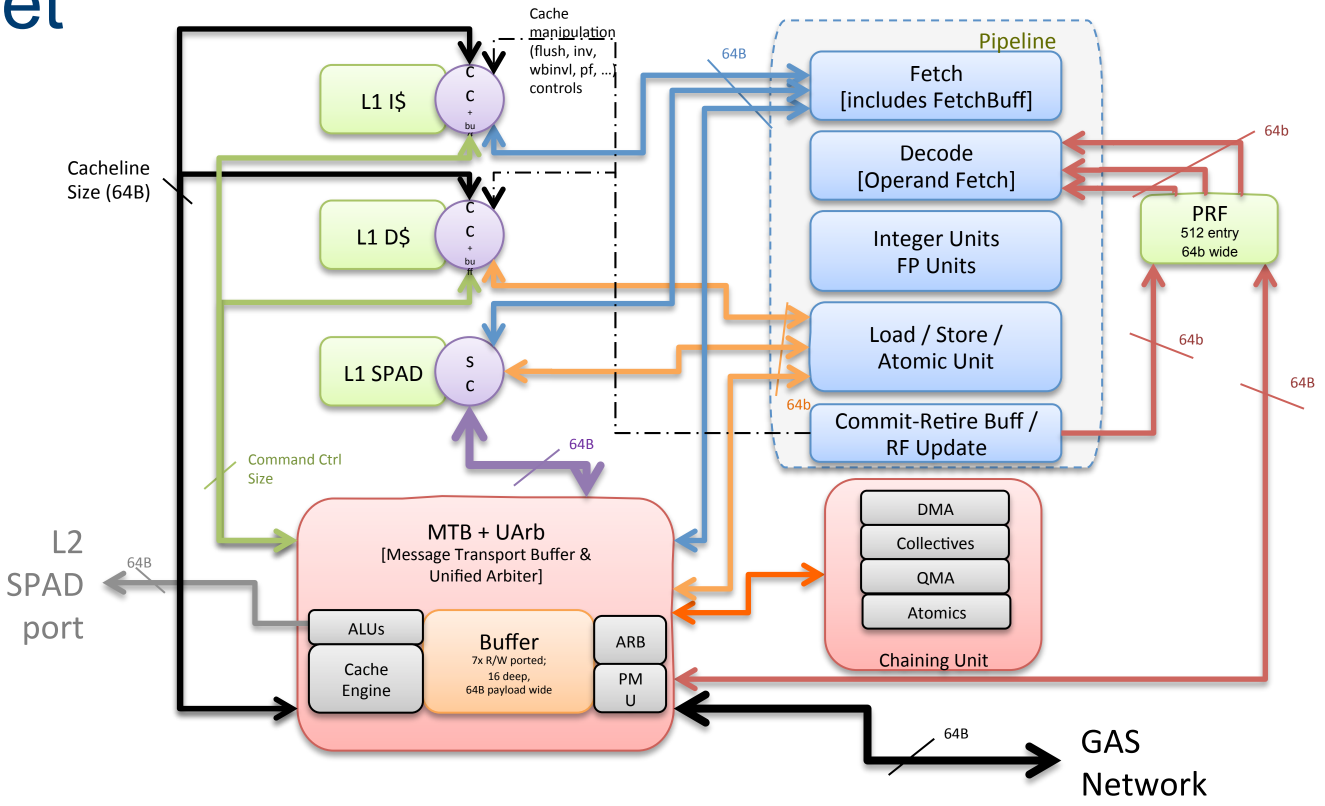
Prototype(s)

Loss of Semantic Information

OCR and SoC Methodology: Co-Design



Execution Engine (XE) architecture – tunable to target



DOE Community Responses and Thoughts (1/2)

Member	DOE Lab	Comments
Dave Richards	LLNL	I'm **pleased by the progress of the OCR project.** MPI-Lite, a full specification, and the expanded feature set including affinity hints are important additions that will expand the set of problems OCR can handle. My biggest concern with OCR is that we have yet to see a believable story regarding how high-level programming models can interface with OCR and capture the full benefits of asynchronous tasks and relocatable data blocks.
Pat McCormick	LANL	The Legion programming model from LANL will be moving on top of OCR to assess performance and viability. If successful, **Legion will use OCR as one of the primary back-end runtimes** to provide task-based execution model support.
Robert Clay	SNL-L	I'm hugely supportive of the approach — specification-based, open design and development. This is what the community needs to build interchangeable 'components' in the task-parallel RTS space. **We (Sandia) plan to remain actively engaged with the OCR team(s), and consider OCR an extremely interesting candidate** for a core part of our future task-parallel RTS in context of a DHARMA build out. While OCR doesn't seem ready for prime time use today, I believe with continued community engagement and support it will develop into a highly effective capability which will significantly enhance its adoption w/in the broader community.

DOE Community Responses and Thoughts (2/2)

Member	DOE Lab	Comments
Ian Karlin	LLNL	The OCR research has two advantages over the other mainly university led asynchronous multi-task (AMT) runtimes. **OCR development is tied to hardware exploration, which allows for hardware features that are beneficial** to AMT runtimes to be co-designed with the programming model. In addition, it is working to become an open community product as opposed to a tool owned by one research group. That said OCR still suffers from the main drawback of most AMT models, as it has yet to be proven that adopting an AMT model is significantly beneficial for enough codes over a well designed non-bulk synchronous MPI application.
Jim Belak	LLNL	I am very pleased by the overall progress and direction of the XSTG project. **I am also excited about the direction and vision that this project is exploring**. With several application codes now expressed in the model, analysis has revealed the need to introduce concepts, such as hints, to enable performant applications using the model. I look forward to seeing large, at-scale applications running to determine where and when this approach is most appropriate for our problems

Progress Report Card

Technology	Rating	Comments
CnC	Good	CnC specification and reference are open-sourced by Intel; Rice is now driving changes and enhancements
HTA	Stopped	HTA works, but is relatively immature in the ability to express constructs and realize performance on OCR. Further work at UIUC is needed on the basic design.
Habanero	Good	Habanero C support for OCR works well. The lack of a centralized standard or reference implementation for Habanero hampers ability to support OCR with all Habanero features and versions.
R-Stream	Good	R-Stream is very effective at automatically generating high-performance OCR code from high-level C loop nests, that exploit the full feature set of OCR and allow users to get the full benefit of the EDT model vs. classical execution models.
OCR (Intel, Rice)	Good	Functionally complete with a precise open specification and reference implementation. Advanced features on resiliency, hints, and related technologies are in progress.
OCR (PNNL)	Good	A version derived from the open reference implementation with a focus on performance first. Source code is not available.
FSim	Good	Very fast, highly parallel, large-scale architecture simulator works and matches the RTL design model at a functional level. Supports bare-metal as well as OCR applications.
Tools (debug, profile)	Limited	First-pass tools are working, but are not going to scale as-is to the expected size of future machines without significant additional work. Collaboration with DOE groups working on similar tools is just starting, and will remain in progress. Starting this was delayed until the last year of the XSTG project, due to the evolving OCR design and interfaces.

Future Work: 1 of 2

Technology	Comments
Maturation - Xeon	Mature code base on scheduler and allocator to bring up performance and strengthen design corners
Maturation – Xeon Phi	Study KNL-based architecture optimization opportunities for performance and introspection
Maturation – FSim	With RTL design closing on specific performance characteristics and power values, adjust FSim to reflect and optimize the tools targeting FSim and OCR on FSim accordingly

Future Work: 2 of 2

Technology	Comments
Productivity - Legion	Support DOE task-based framework Legion on top of OCR to provide a comparison baseline
Productivity - RAJA	Exploit loop lambda model of RAJA to emit OCR tasks for parallelization evaluation; couple to Rstream and similar tools for "auto-taskification" skeleton for existing codes
Enhancements - Tools	Continue to enhance and extend early debug and profiling tools with DOE teams to facilitate using task-based execution models at scale