X-Stack, Traleika Glacier (XSTG)

Three-year project results and products, including

Intel, Rice, UCSD, PNNL, UIUC, Reservoir Labs, U.Vienna, & 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

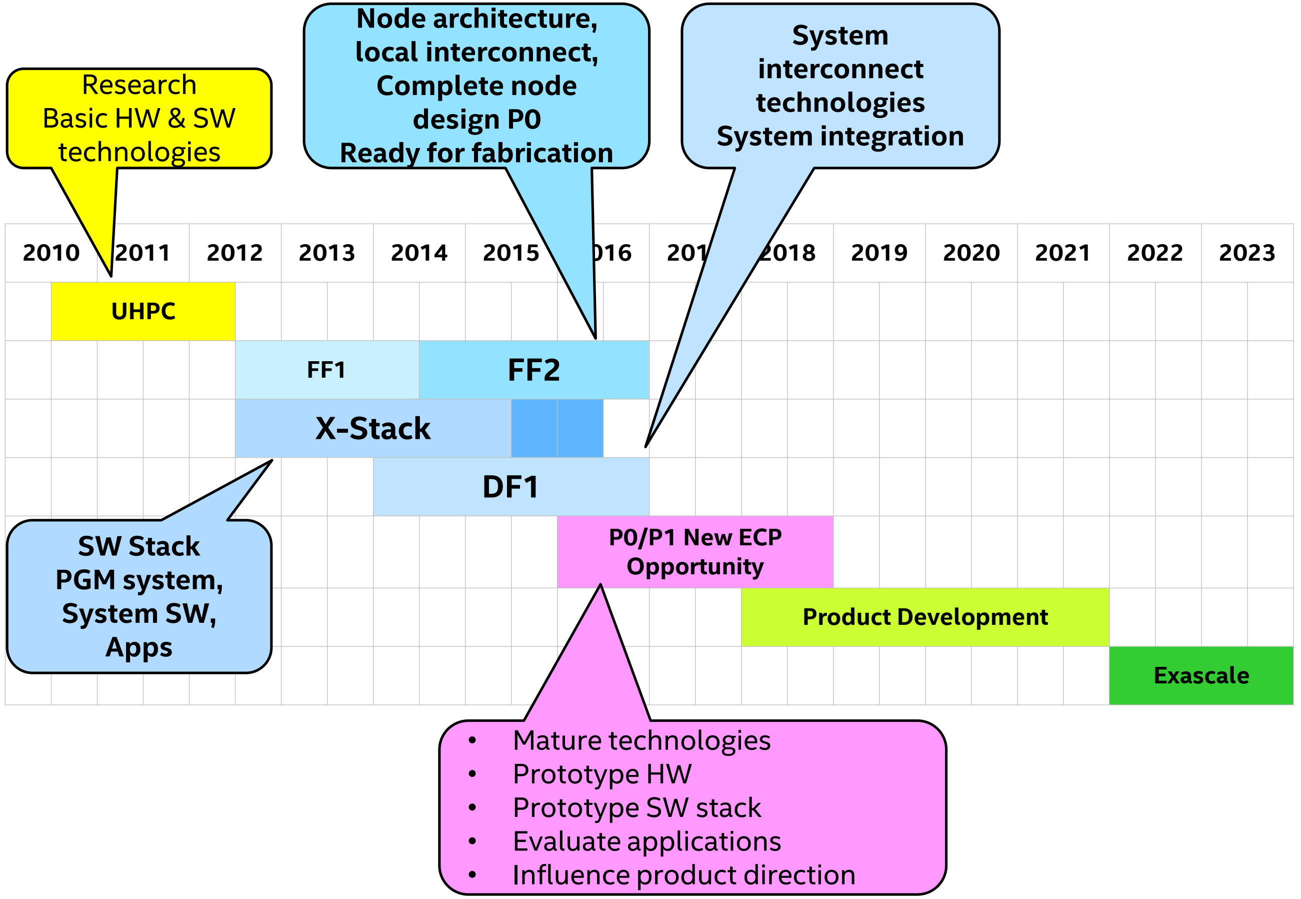
Future Work

Demo Invitation

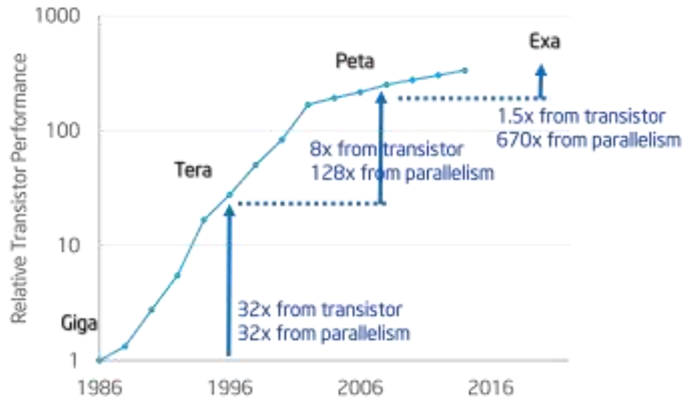
Top Extreme-scale Challenges

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**

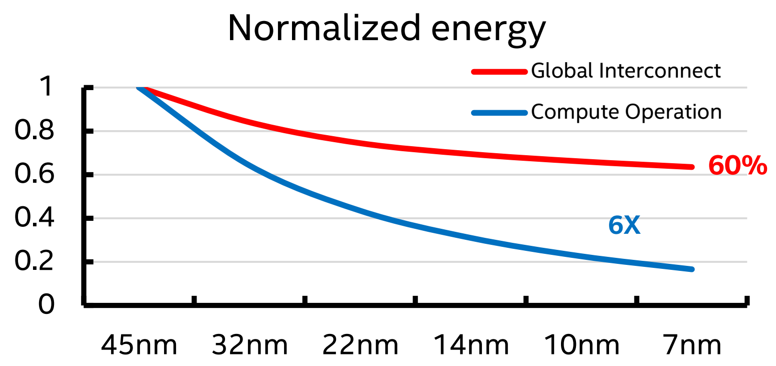
Extreme-Scale Programs for <u>Ubiquity</u>



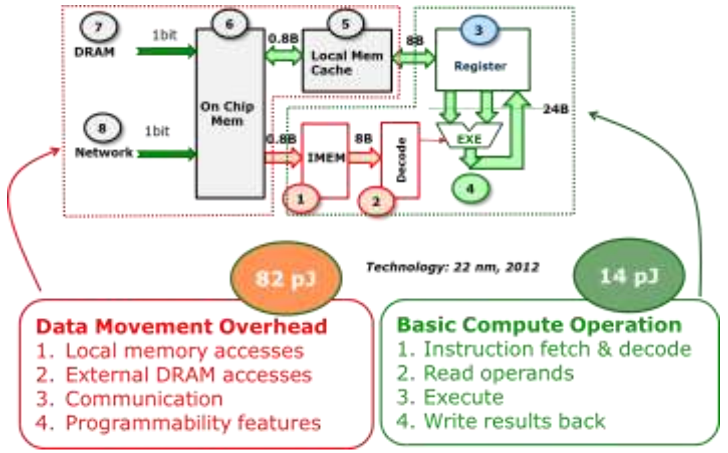
Compute vs Data Movement

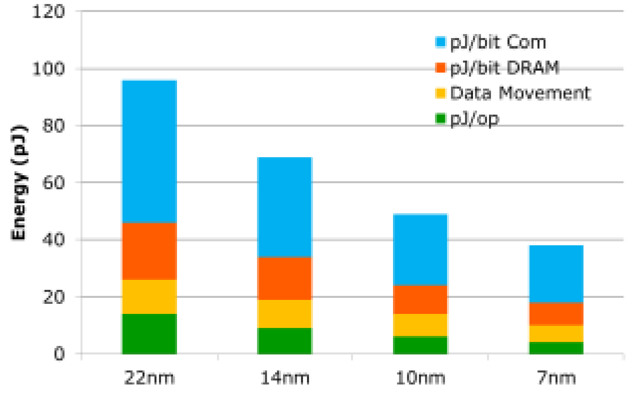


Performance from parallelism

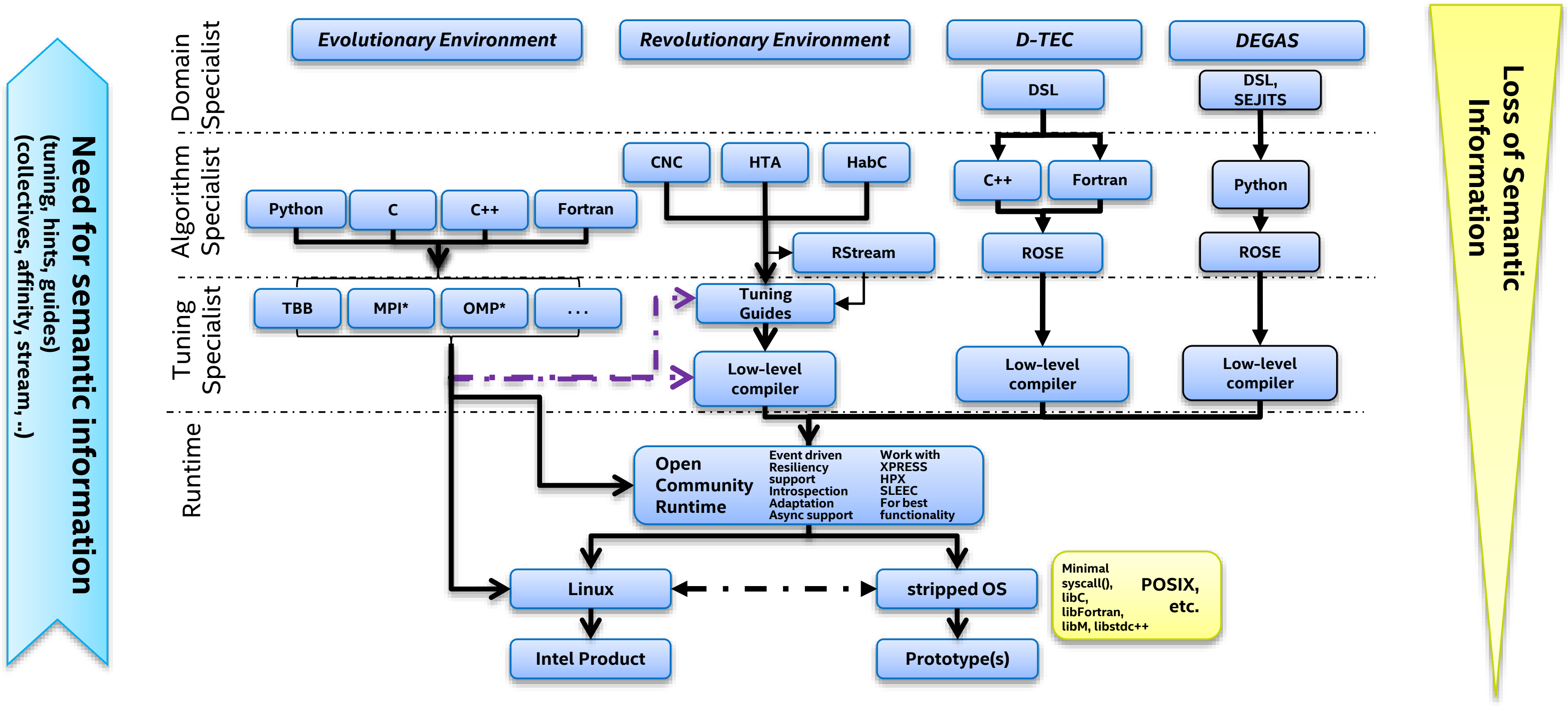


Basic compute loop





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 (publication focused)
    - R-Stream is closed-source with government use rights (full license available)
- 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

XSTG has transitioned from Research to Technology Maturation

Development websites

GIT repositories

- OCR: https://xstack.exascale-tech.com/git/public/ocr.git
- Applications: https://xstack.exascale-tech.com/git/public/apps.git
- TG: https://xstack.exascale-tech.com/git/public/tg.git

Wiki:

- https://xstack.exascale-tech.com/wiki
- https://eci.exascale-tech.com/wiki

Bug tracker: https://xstack.exascale-tech.com/redmine

Code review/contribution: https://xstack.exascale-tech.com/gerrit

Regression tests: https://xstack.exascale-tech.com/jenkins

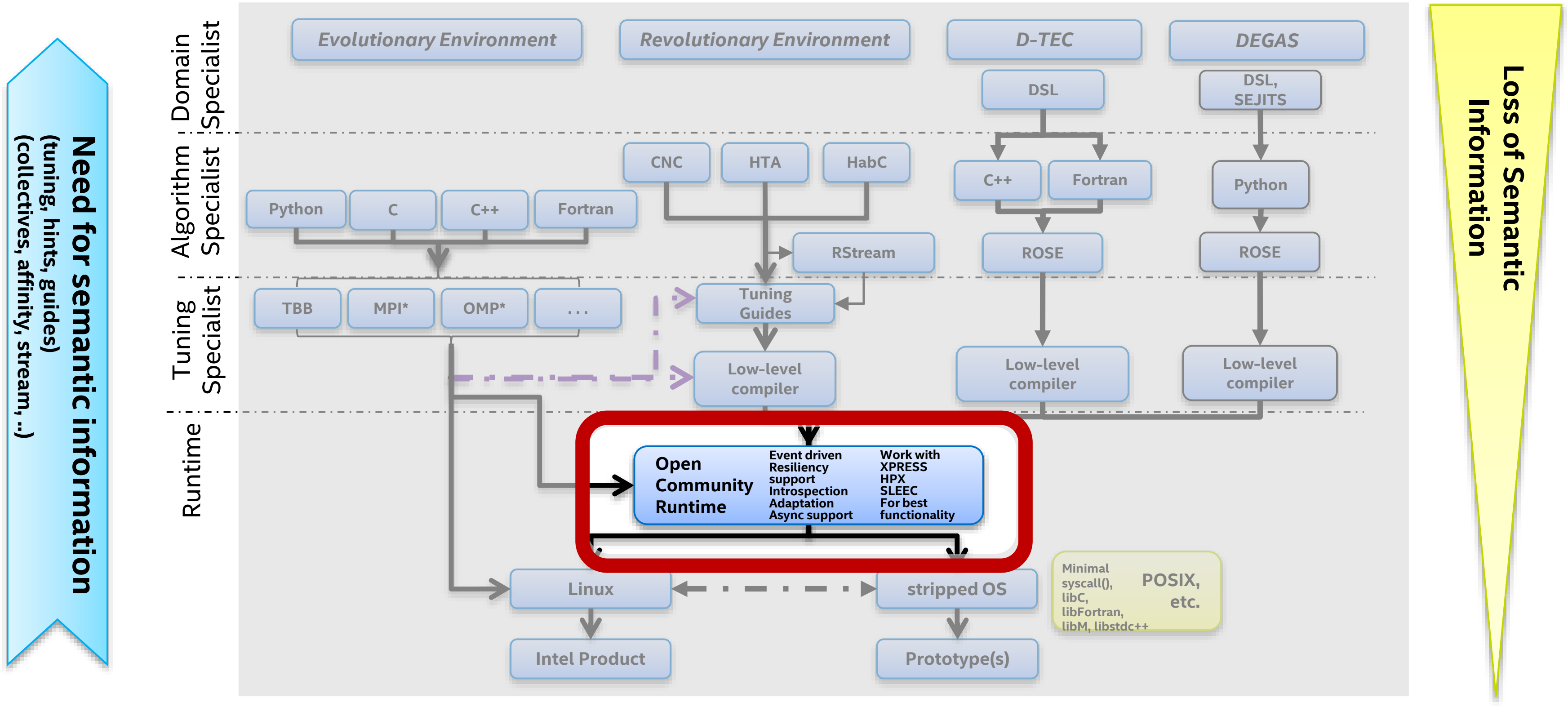
Account management: https://eci.exascale-tech.com/lam/selfService

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

- FOR THIS PUBLIC VERSION, EMBARGOED RESULTS HAVE BEEN REMOVED
    - Contact me for further details (joshua.b.fryman@intel.com)

Task-Execution Runtime (Intel, Rice)



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*

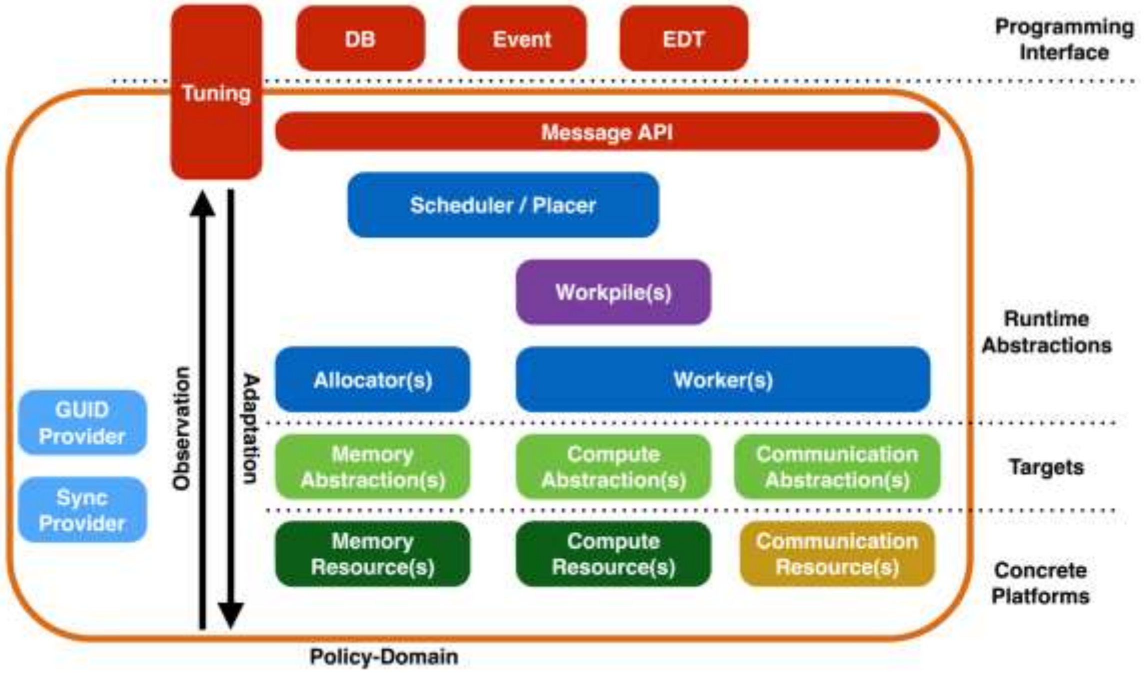
Runtime Design Principles of OCR

Modular

Extensible

Adaptable

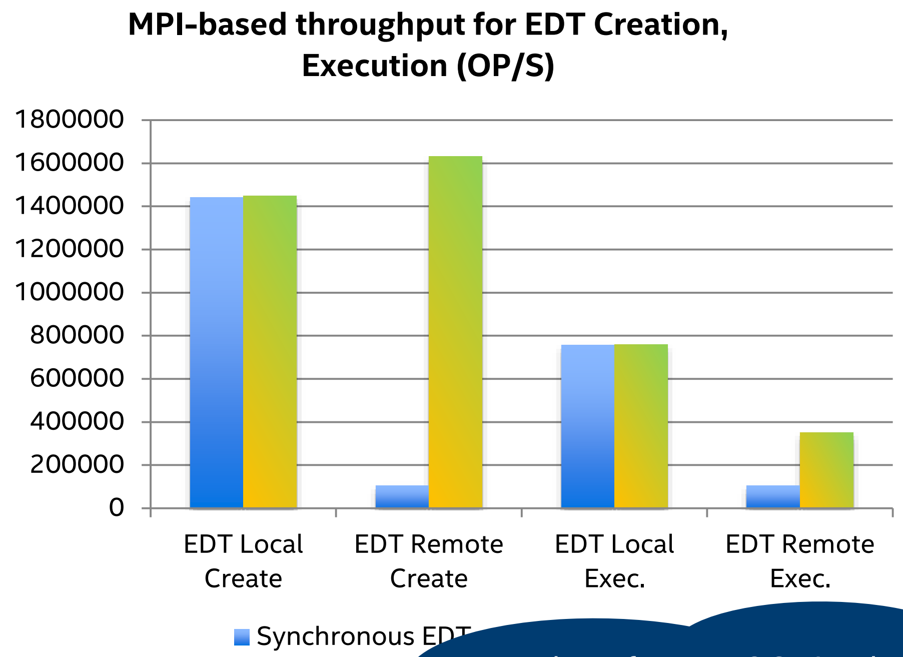
Tunable

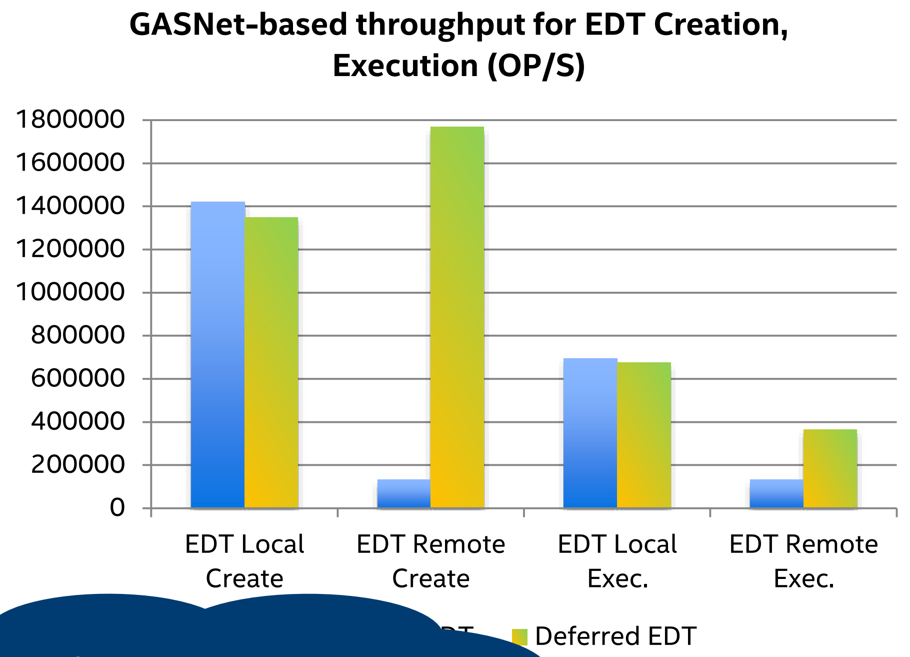


Generic, Adaptive

Platform Specific

Reference OCR: MPI vs GASnet

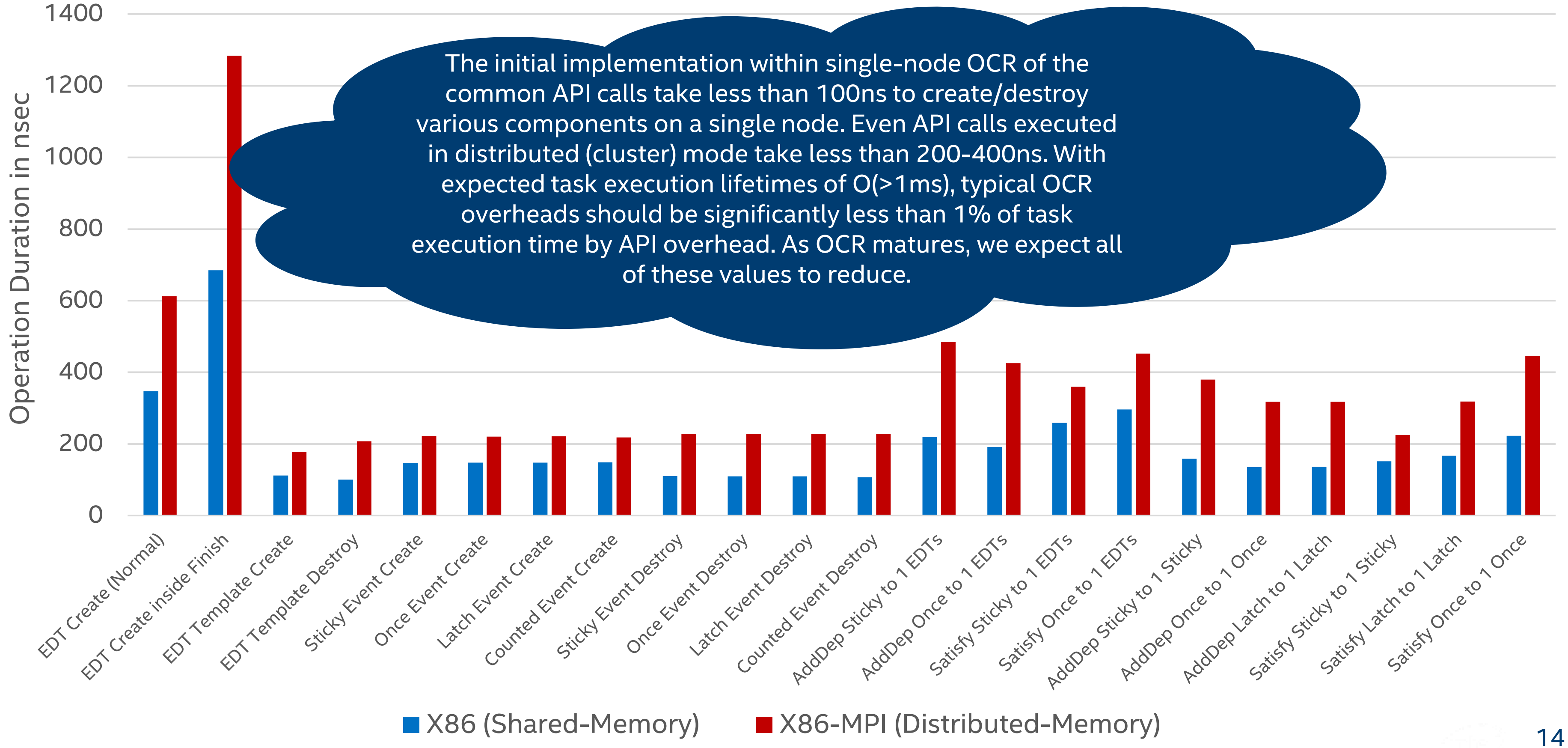




64 node … rconnect

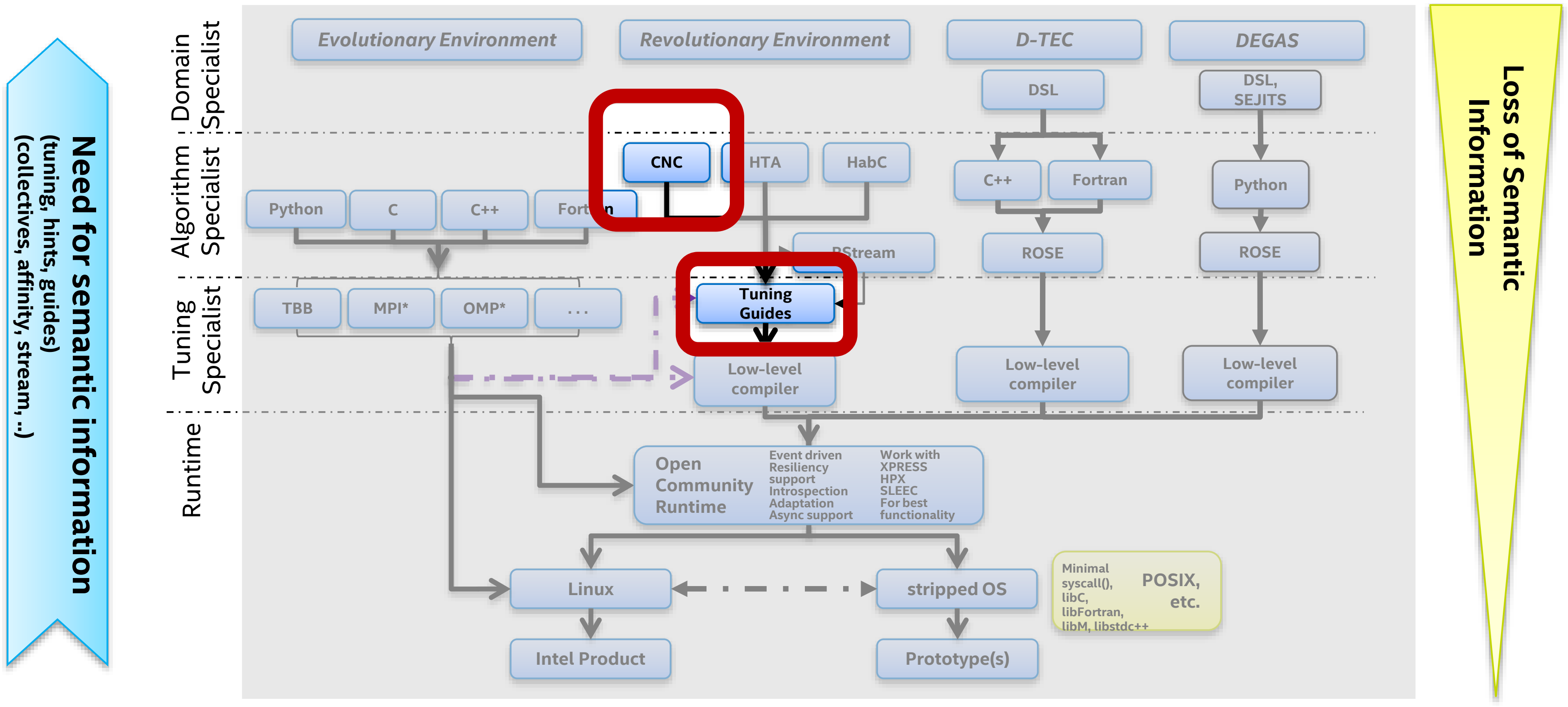
The reference OCR implementation supports running on clusters using either direct MPI calls or GASnet calls. Performance is similar between the two when measuring only the OCR API throughput.

Reference OCR: API overhead on NERSC

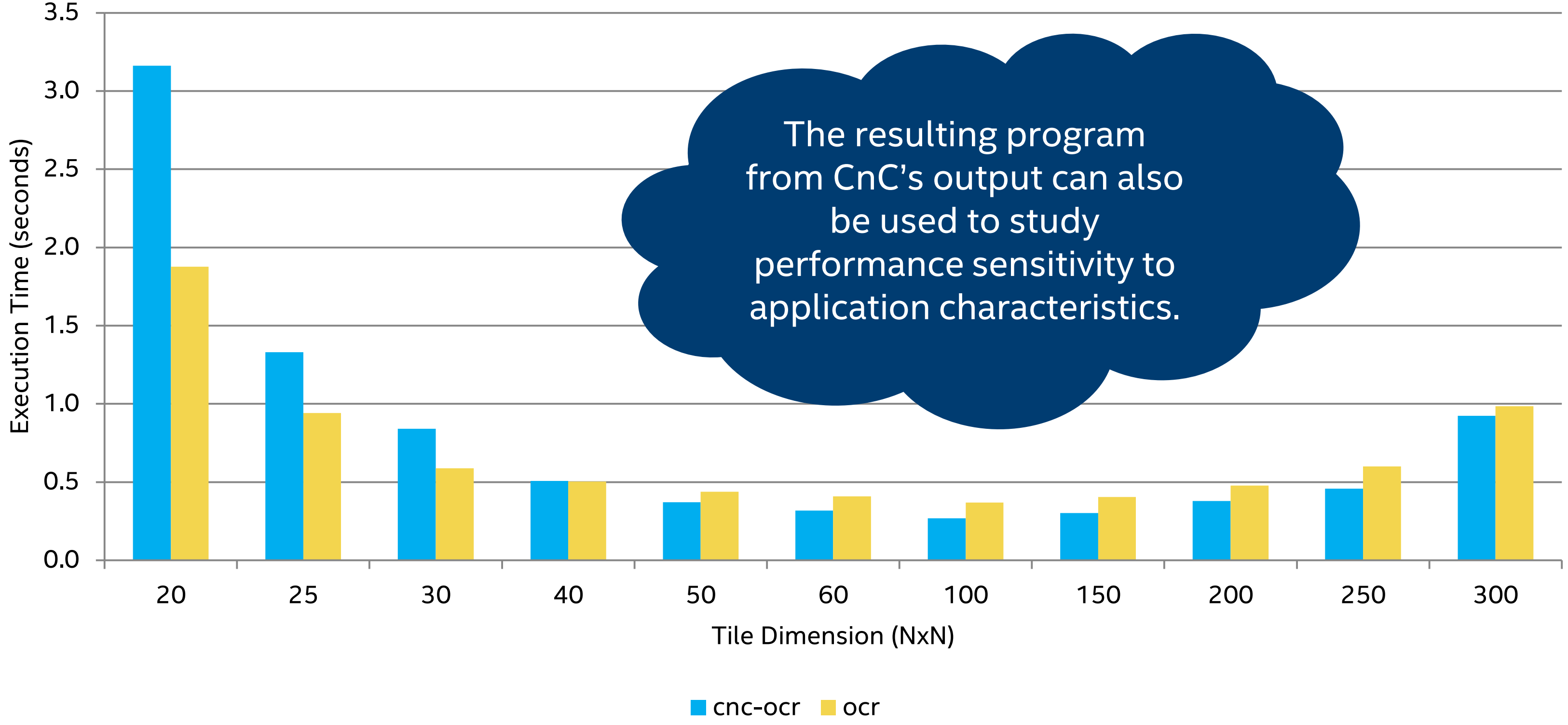


The initial implementation within single-node OCR of the common API calls take less than 100ns to create/destroy various components on a single node. Even API calls executed in distributed (cluster) mode take less than 200-400ns. With expected task execution lifetimes of O(>1ms), typical OCR overheads should be significantly less than 1% of task execution time by API overhead. As OCR matures, we expect all of these values to reduce.

Productivity: Concurrent Collections (Rice, Intel)

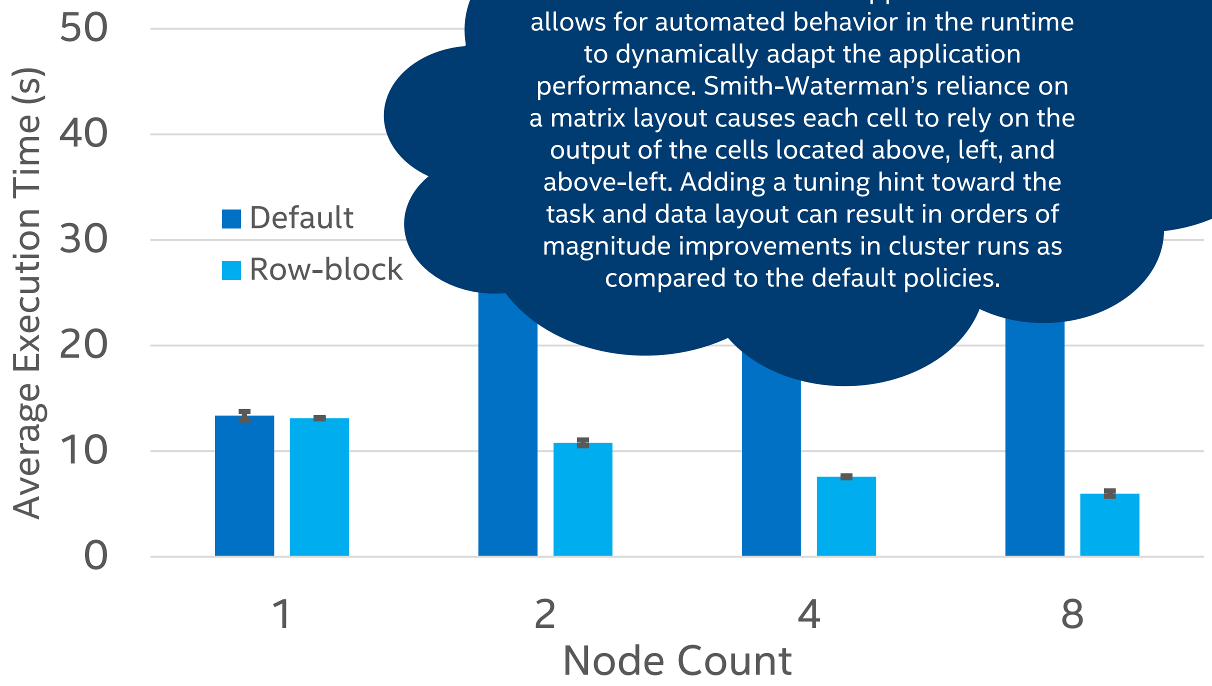


CnC Results: Sweeping tile sizes in Cholesky



The resulting program from CnC's output can also be used to study performance sensitivity to application characteristics.

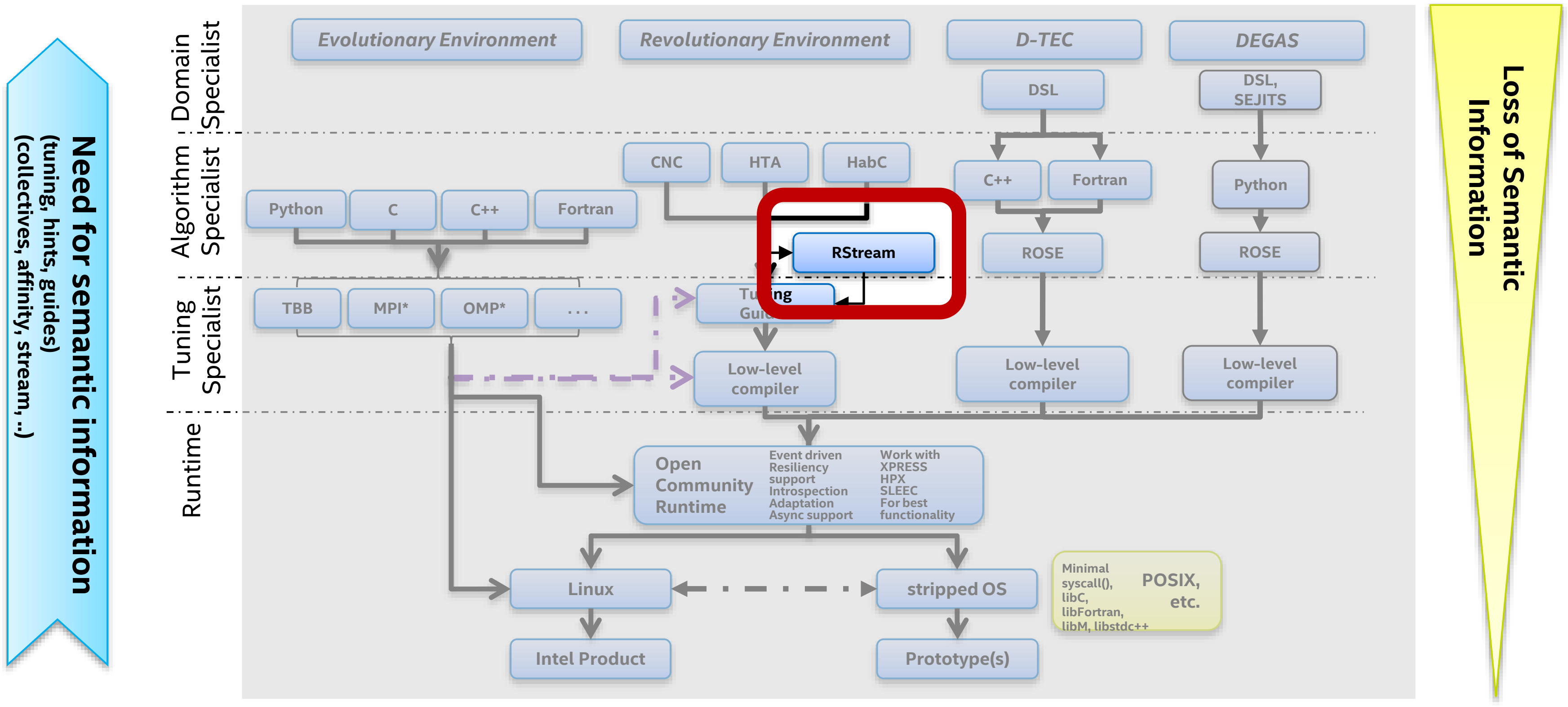
CnC Results: Tuning Smith-Waterman Sequence



The addition of "Tuning Hints" to the OCR API as well as the CnC support for OCR allows for automated behavior in the runtime to dynamically adapt the application performance. Smith-Waterman's reliance on a matrix layout causes each cell to rely on the output of the cells located above, left, and above-left. Adding a tuning hint toward the task and data layout can result in orders of magnitude improvements in cluster runs as compared to the default policies.

- Each input sequence length ~300k
- Dynamic programming optimization on ~90-billion cell matrix
- Tiles of 306x354 cells
- Total of 569x661 tiles
- Default: CnC default distribution
- Row-block: Rows in blocks of 16
- 10 runs per configuration

Productivity: R-Stream (Reservoir Labs)



R-Stream: Polymorphic Optimizing Translator

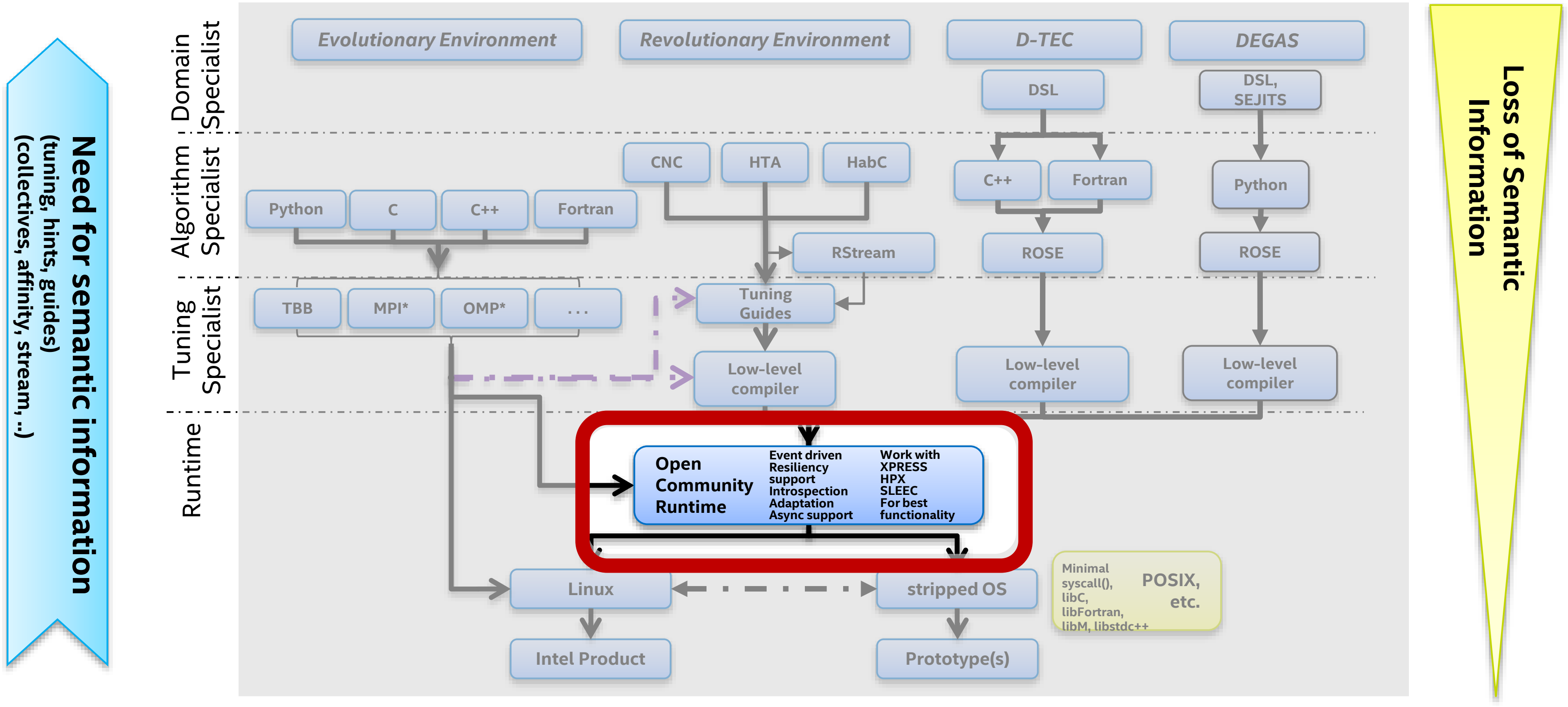
```
for(k=1;k<NPAD-1;k++){
    for(j=1;j<NPAD-1;j++){
        for(i=1;i<NPAD-1;i++){
            x_np1_1[k][j][i] = x_np1_0[k][j][i]
                               c1*(x_np1_0[k][j][i]
                               c2*Dinv_ijk()* (rhs[k
}}}}}
```

Input C code

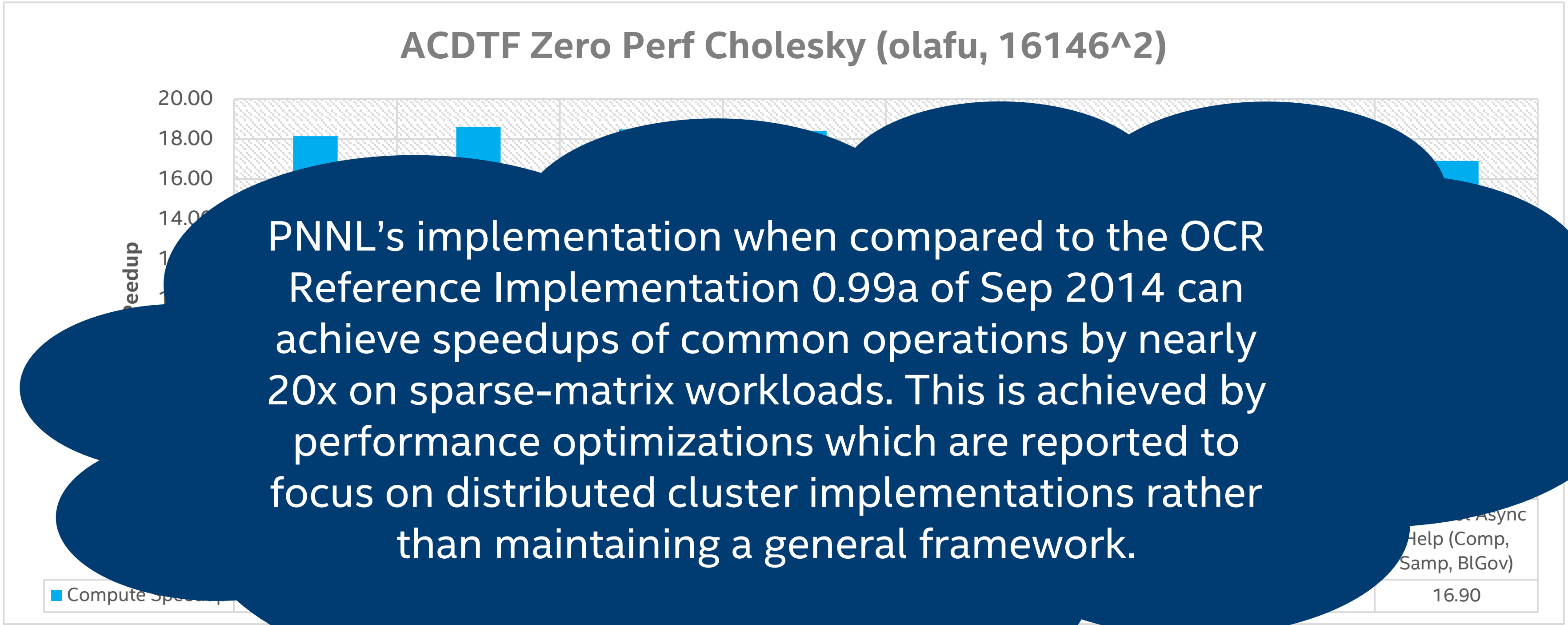
- Automatic parallelization and locality optimization
- Automatic explicit communications generation and explicit datablock management
- Automatic formation of OCR EDT programs (and other runtimes)
- Dynamic creation and management of dependences

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 variant of OCR)



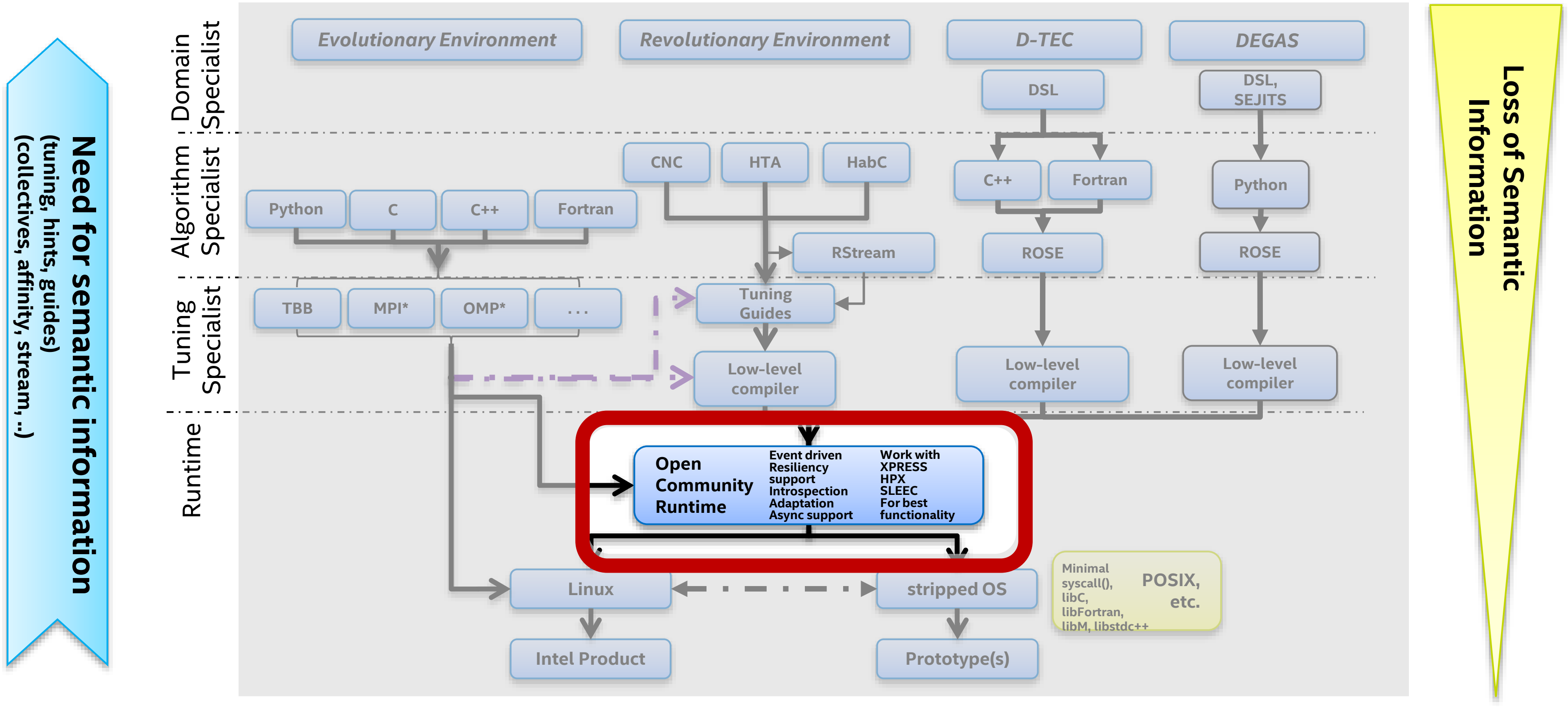
OCR Reference Implementation Derivative (PNNL)



PNNL's implementation when compared to the OCR Reference Implementation 0.99a of Sep 2014 can achieve speedups of common operations by nearly 20x on sparse-matrix workloads. This is achieved by performance optimizations which are reported to focus on distributed cluster implementations rather than maintaining a general framework.

ACDFT-OCR running on community O [illegible] nt points across its different configurations -- synchronous vs. (fast-) asynchronous, sampling vs. non-sampling, various [illegible] pression, Samp = Sampling, Gov = Governor). Furthermore, the asynchronous version has a portable implementation that works on any versi [illegible] ersion which requires pointers for the platform to be able to address any memory location. ACDTF supports a centralized core to handle requests (Cent). Block size is ~300.

Task-Execution Runtime (Vienna version of OCR)



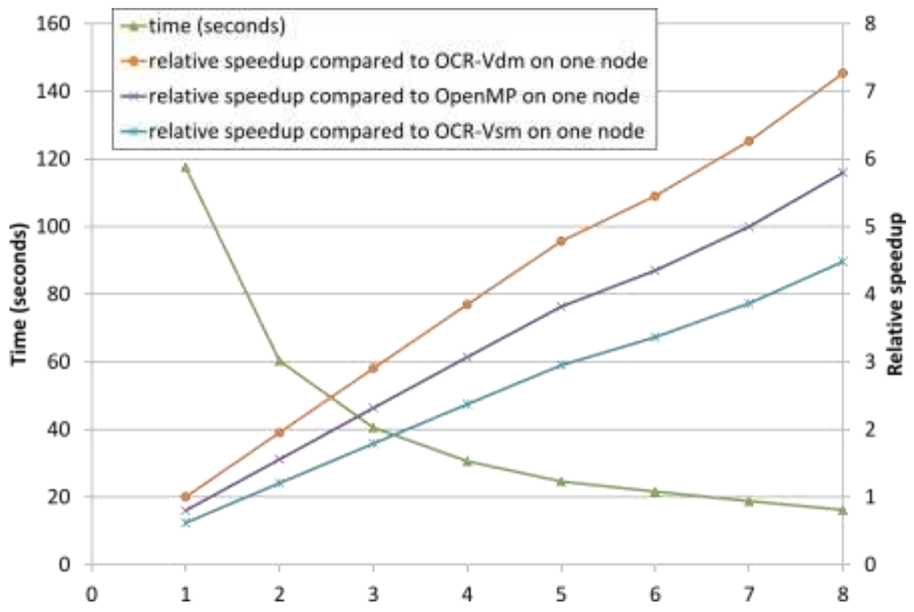
Performance (seismic application, preliminary)

CPU: dual Xeon E5-2650 (8 cores per socket)

Xeon Phi 5110P (60 cores)

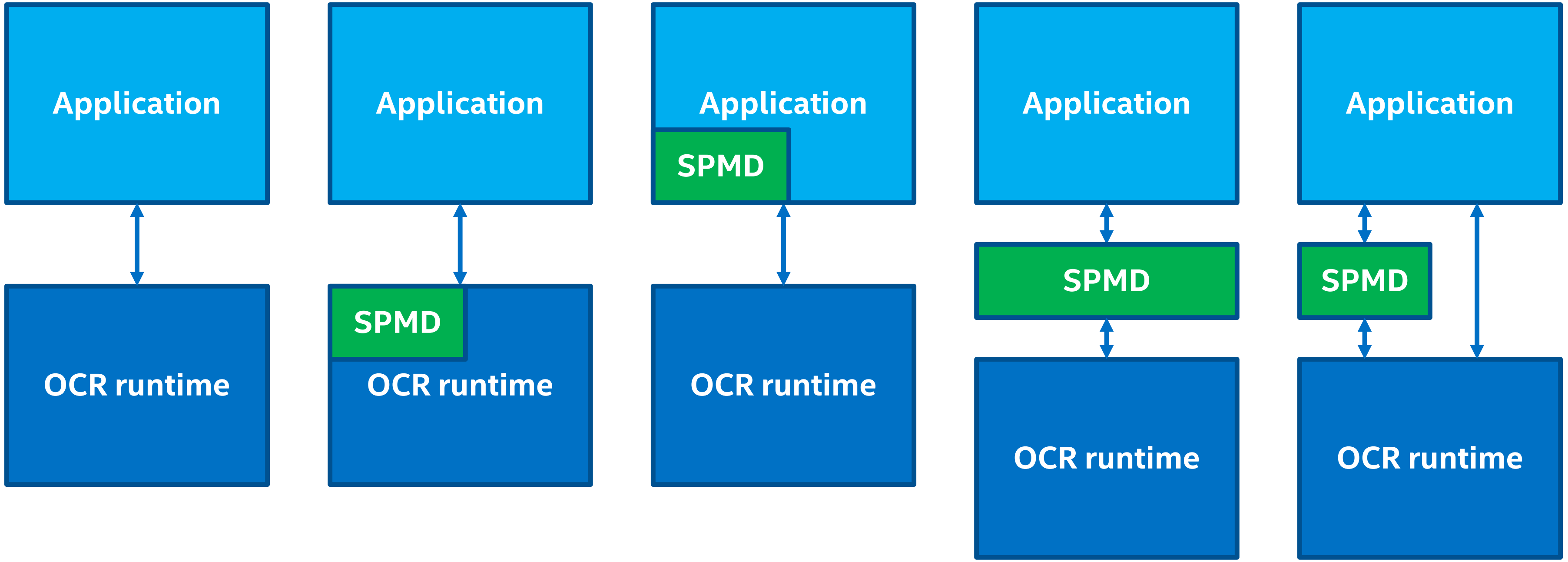
2D stencil code, computation heavy

	OCR-Vsm	OpenMP
host	72.50 s	93.75 s
KNC (Phi)	35.53 s	35.34 s



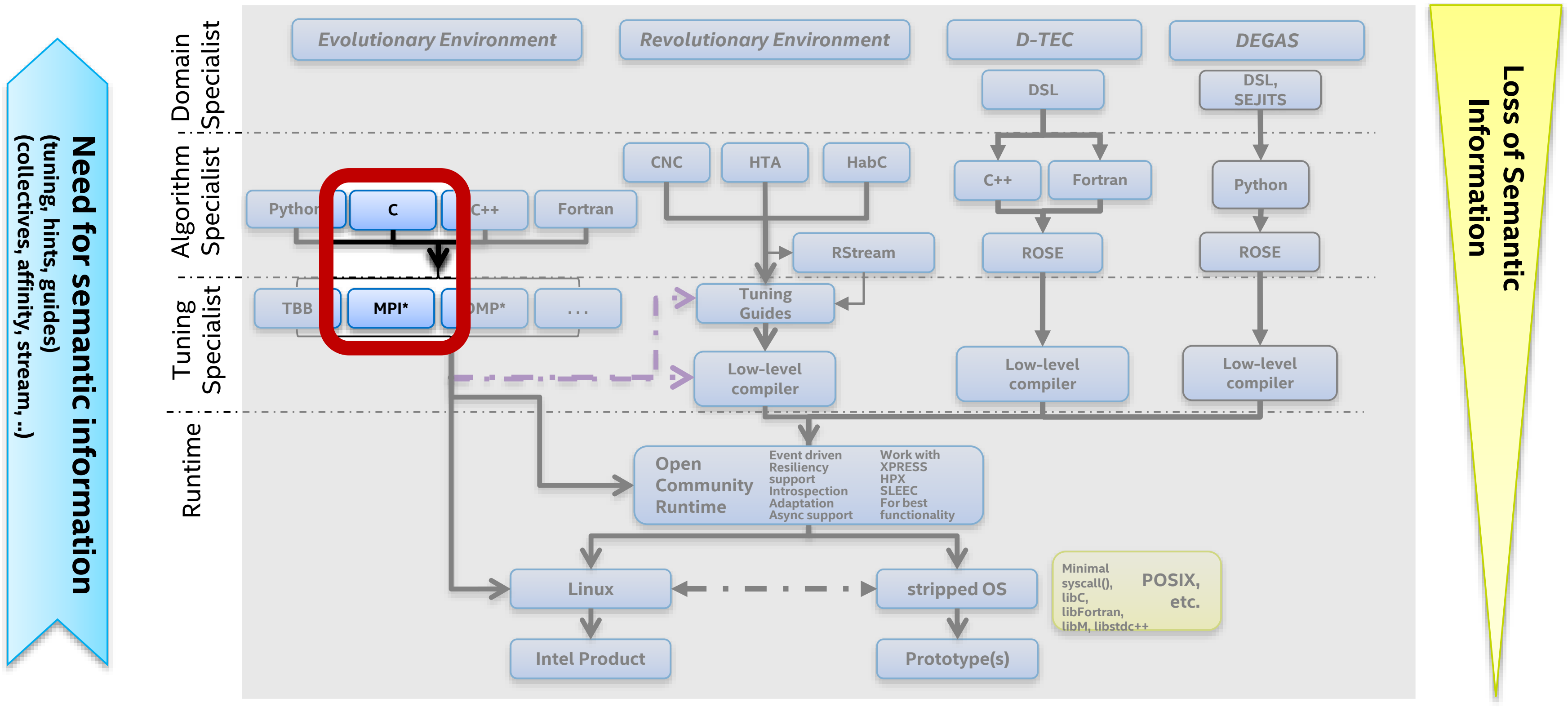
Credit: Jiri Dokulil, Univ. of Vienna

Support for SPMD-style OCR applications

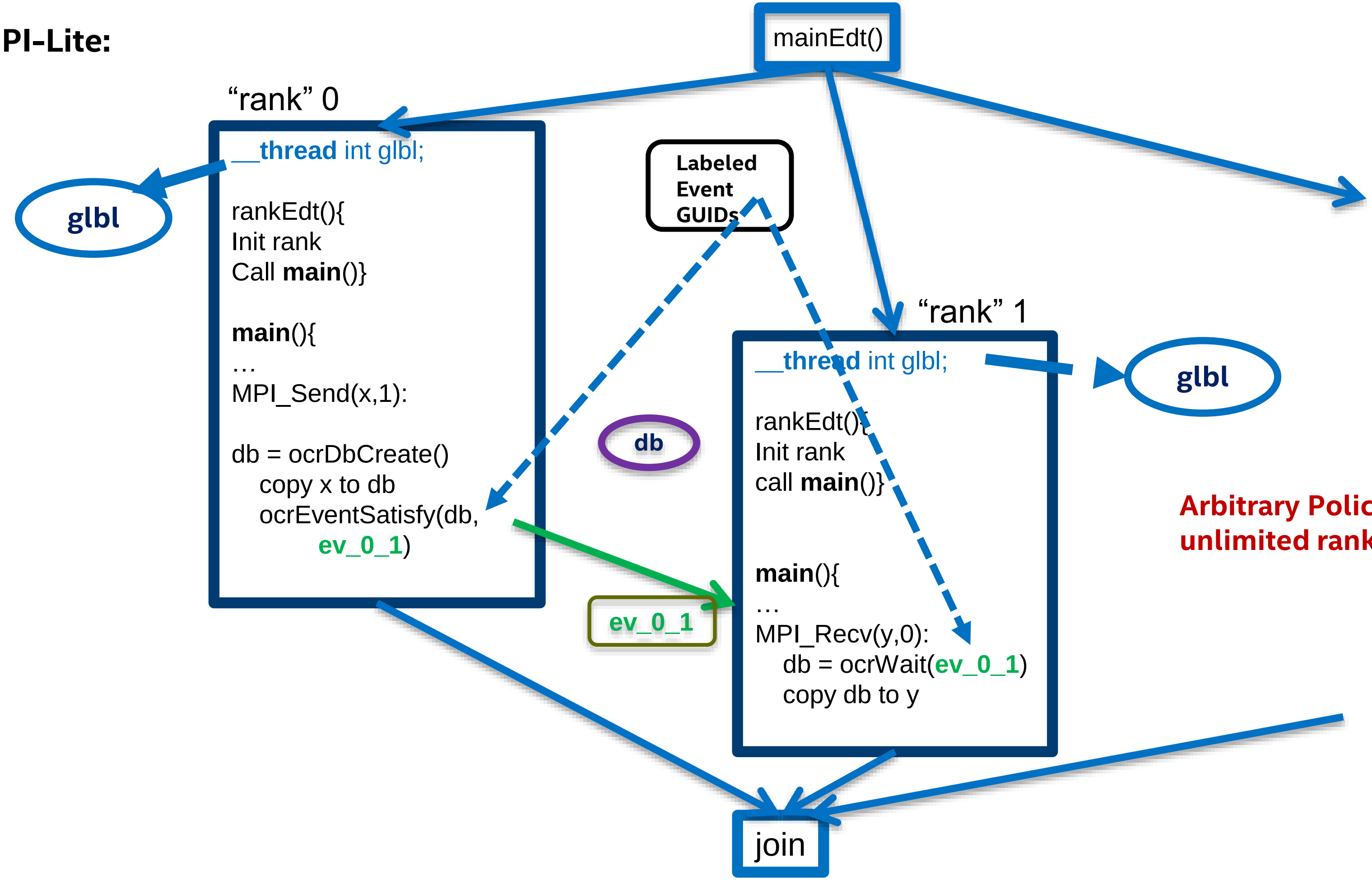


Credit: Jiri Dokulil, Univ. of Vienna

Software Ecosystem: Support today, find tomorrow



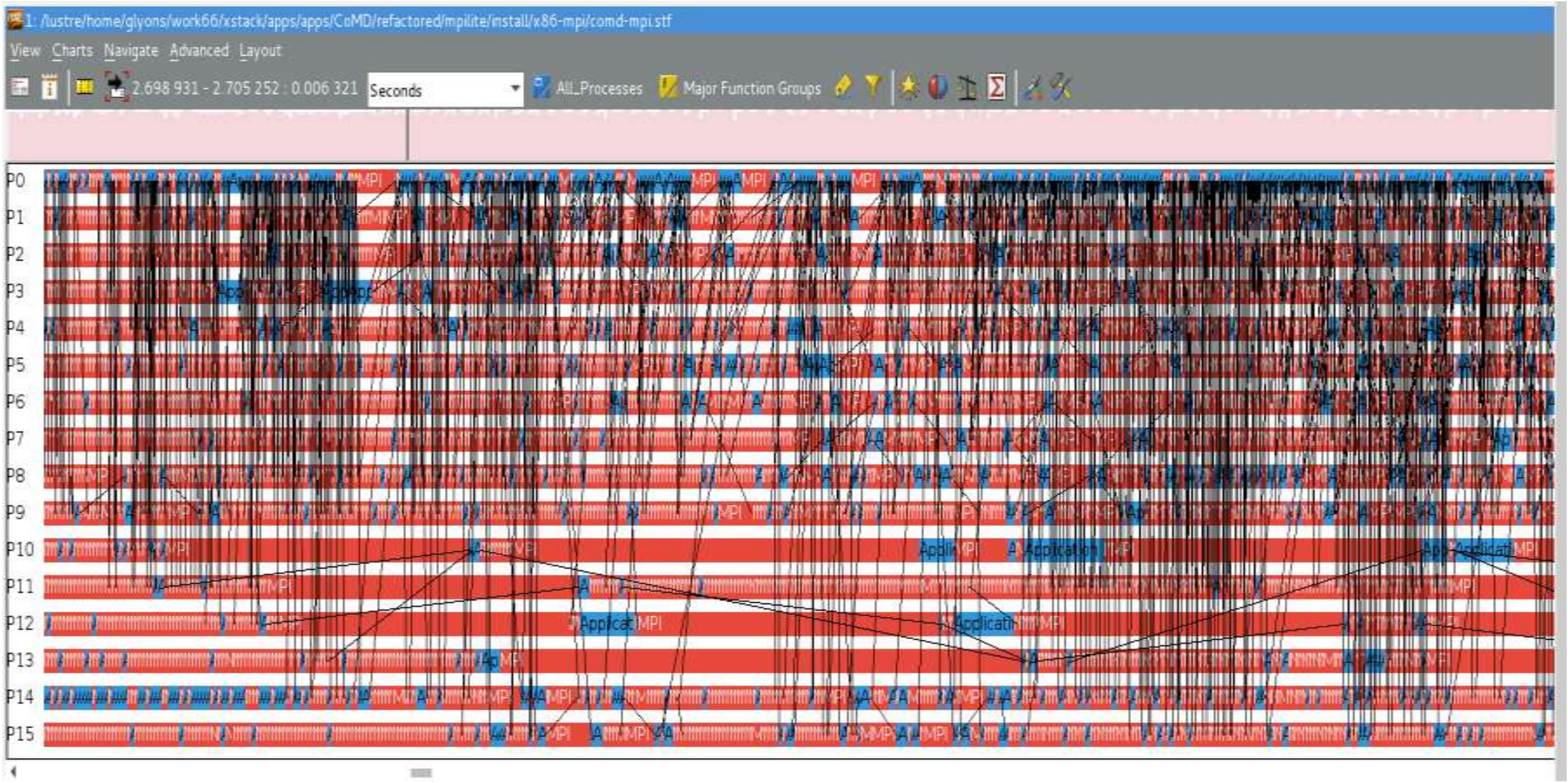
MPI-Lite:

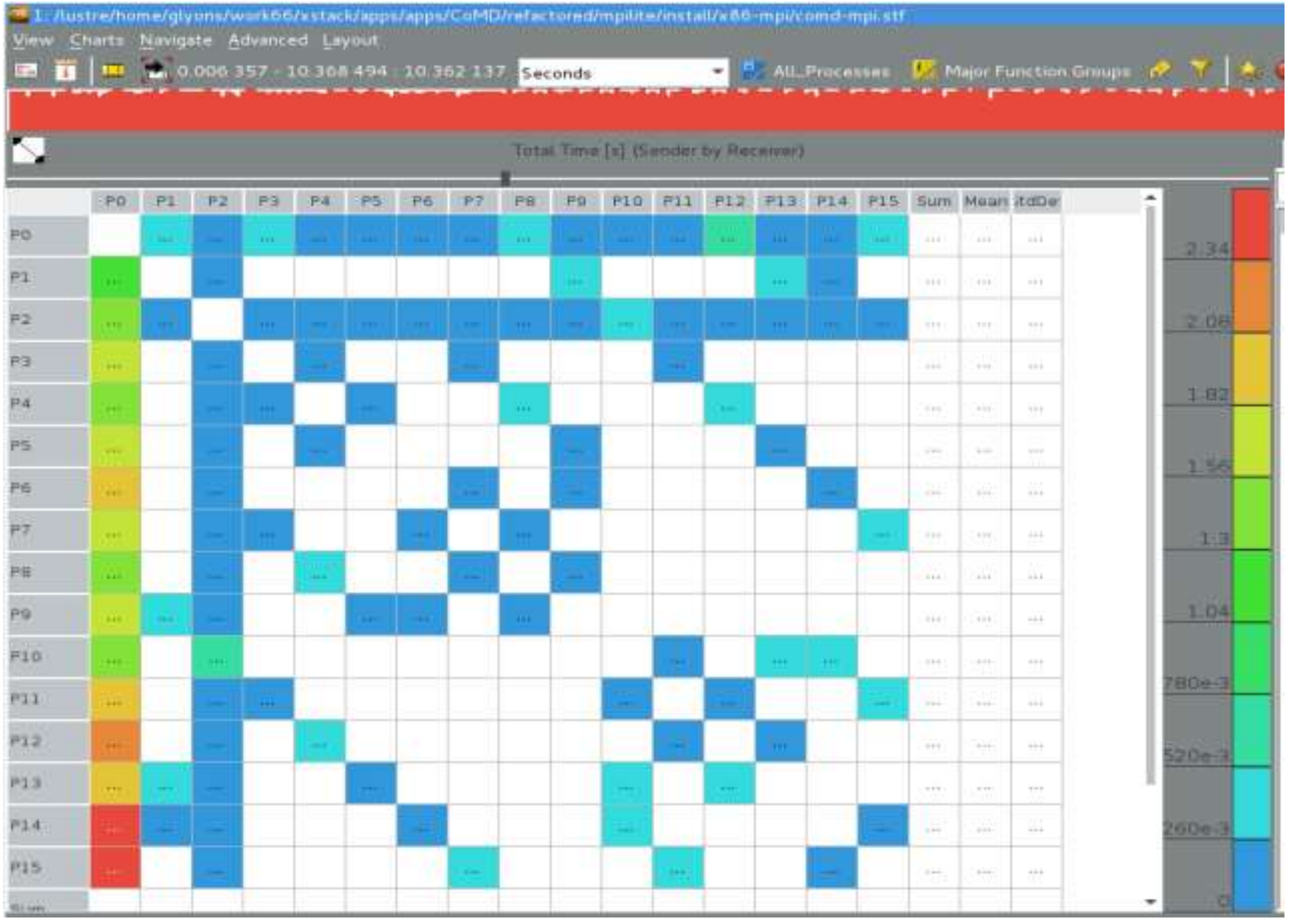


```
__thread int glbl;

rankEdt(){
Init rank
Call main()}

main(){
…
MPI_Send(x,1):

db = ocrDbCreate()
    copy x to db
    ocrEventSatisfy(db,
        ev_0_1)
```

```
__thread int glbl;

rankEdt(){
Init rank
call main()}

main(){
…
MPI_Recv(y,0):
    db = ocrWait(ev_0_1)
    copy db to y
```

Arbitrary Policy Domains; unlimited ranks


1: /lustre/home/glyons/work66/xstack/apps/apps/CoMD/refactored/mpilite/install/x86-mpi/comd-mpi.stf
View
Charts
Navigate
Advanced
Layout
2.698 931 - 2.705 252 : 0.006 321
Seconds
All_Processes
Major Function Groups
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
MPI
Application



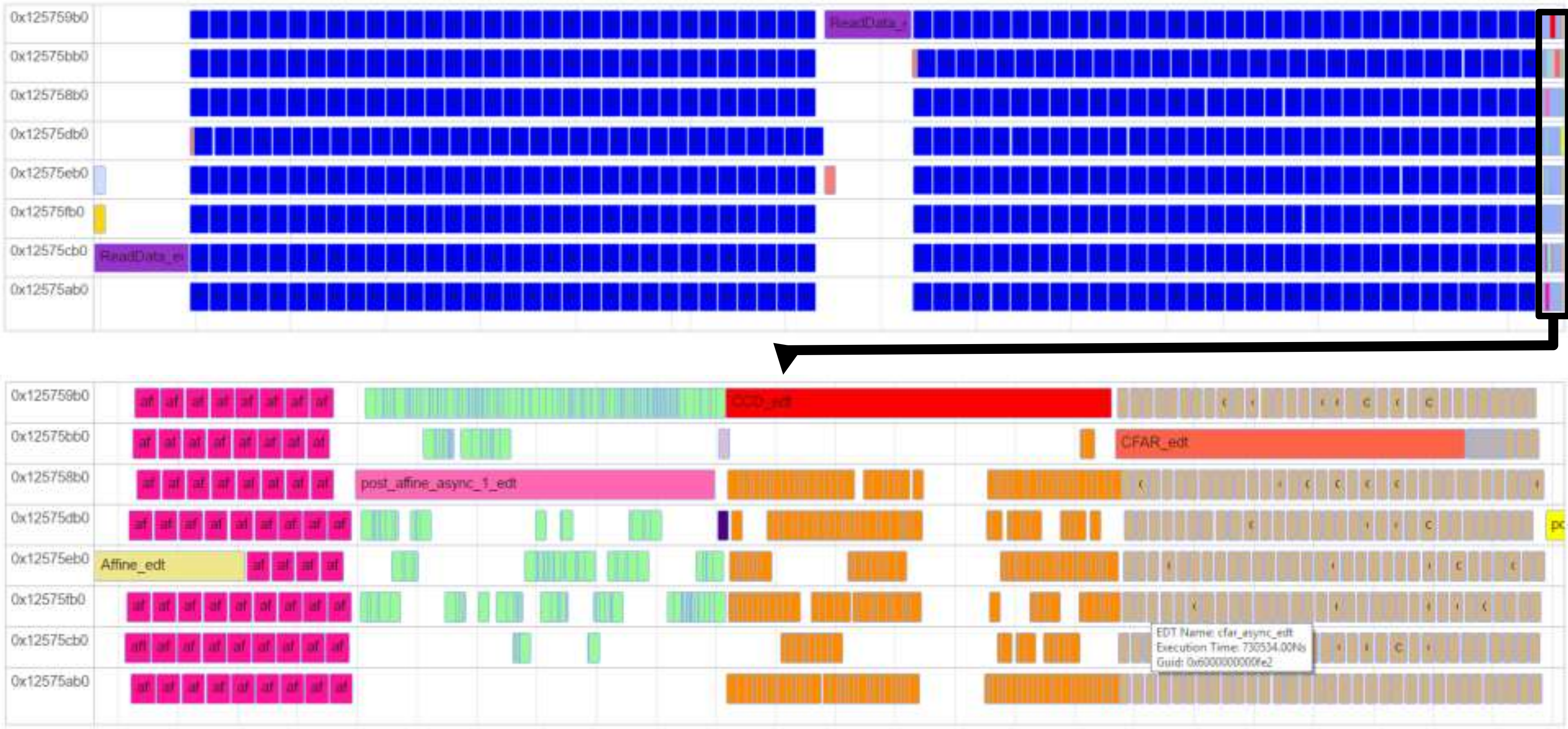
Total Time [s] (Sender by Receiver)

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	Sum	Mean	StdDe
P0		...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
P1	...		...							...				...	...		...	...	...
P2	...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
P3	...		...		...			...				...					...	...	...
P4	...		...	...		...			...				...				...	...	...
P5	...		...		...					...				...			...	...	...
P6	...		...					...		...					...		...	...	...
P7	...		...	...			...		...							...	...	...	...
P8	...		...		...			...		...							...	...	...
P9	...	...	...			...	...		...								...	...	...
P10	...		...									...		...	...		...	...	...
P11	...		...	...							...		...			...	...	...	...
P12	...		...		...							...		...			...	...	...
P13	...	...	...			...					...		...				...	...	...
P14	...	...	...				...				...					...	...	...	...
P15	...		...					...				...			...		...	...	...
Sum																			

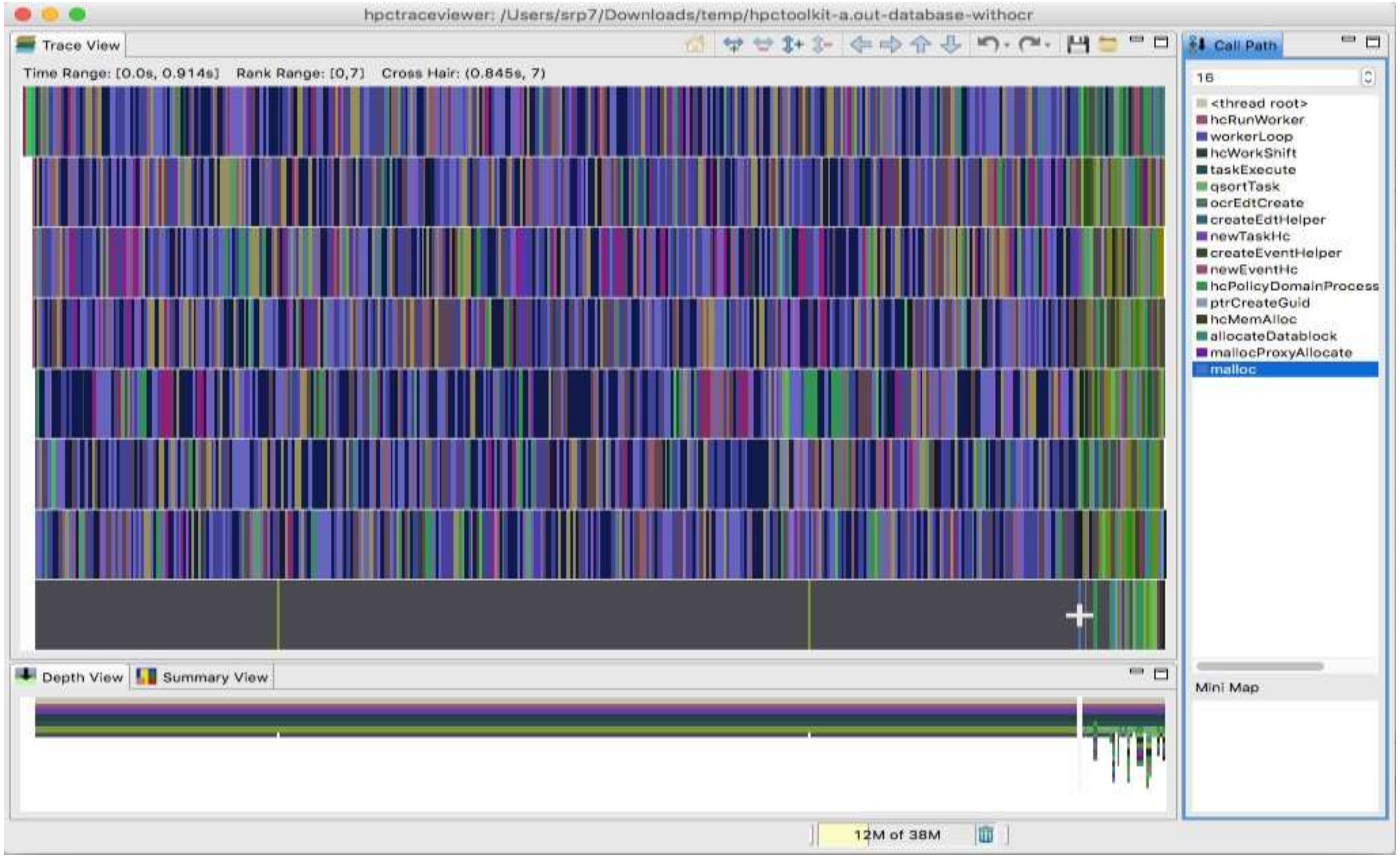
Total (CPU) execution time: 6681537942 nanoseconds

Total number of executed tasks: 1362

Workers



EDT Name: cfar_async_edt
Execution Time: 730534.00Ns
Guid: 0x6000000000fe2


hpctraceviewer: /Users/srp7/Downloads/temp/hpctoolkit-a.out-database-withocr

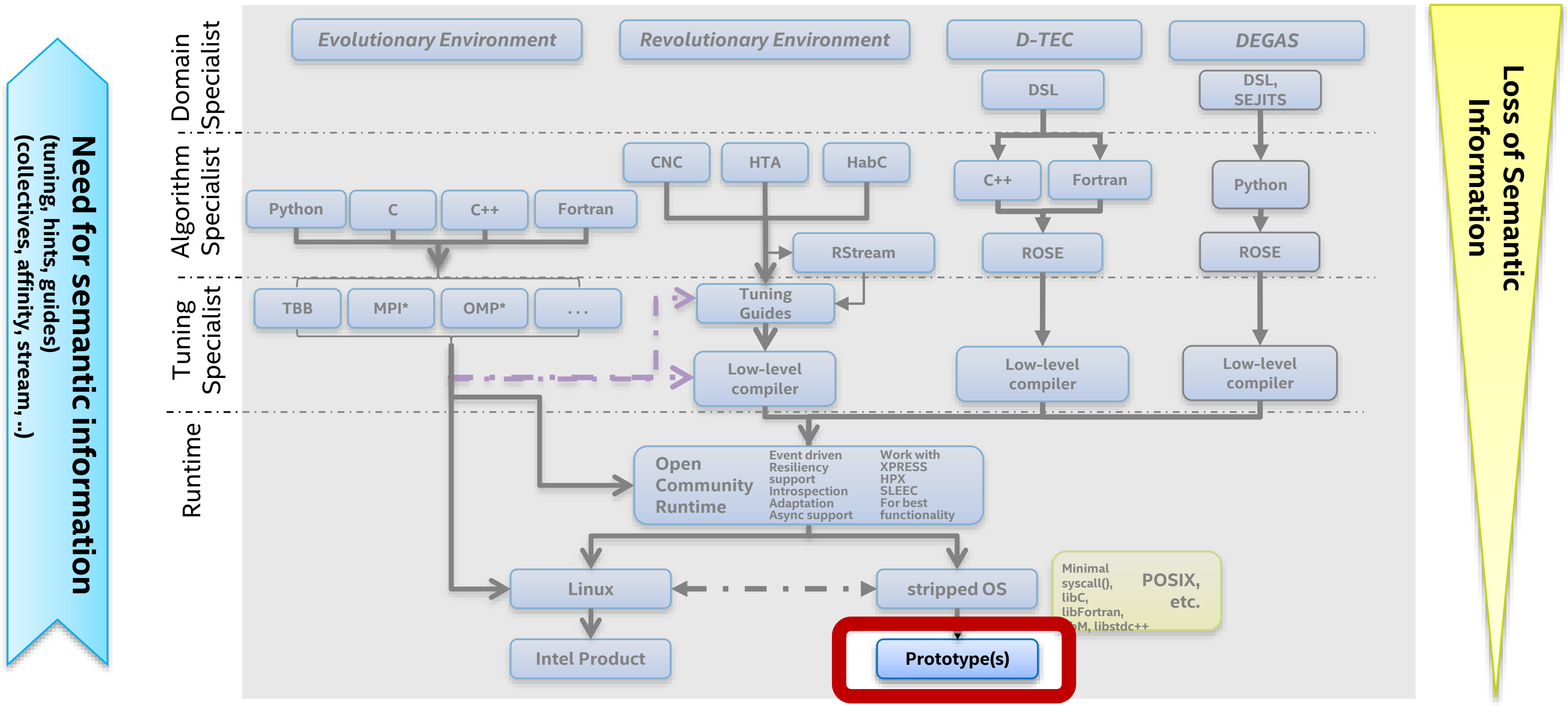
Time Range: [0.0s, 0.914s]   Rank Range: [0,7]   Cross Hair: (0.845s, 7)

16

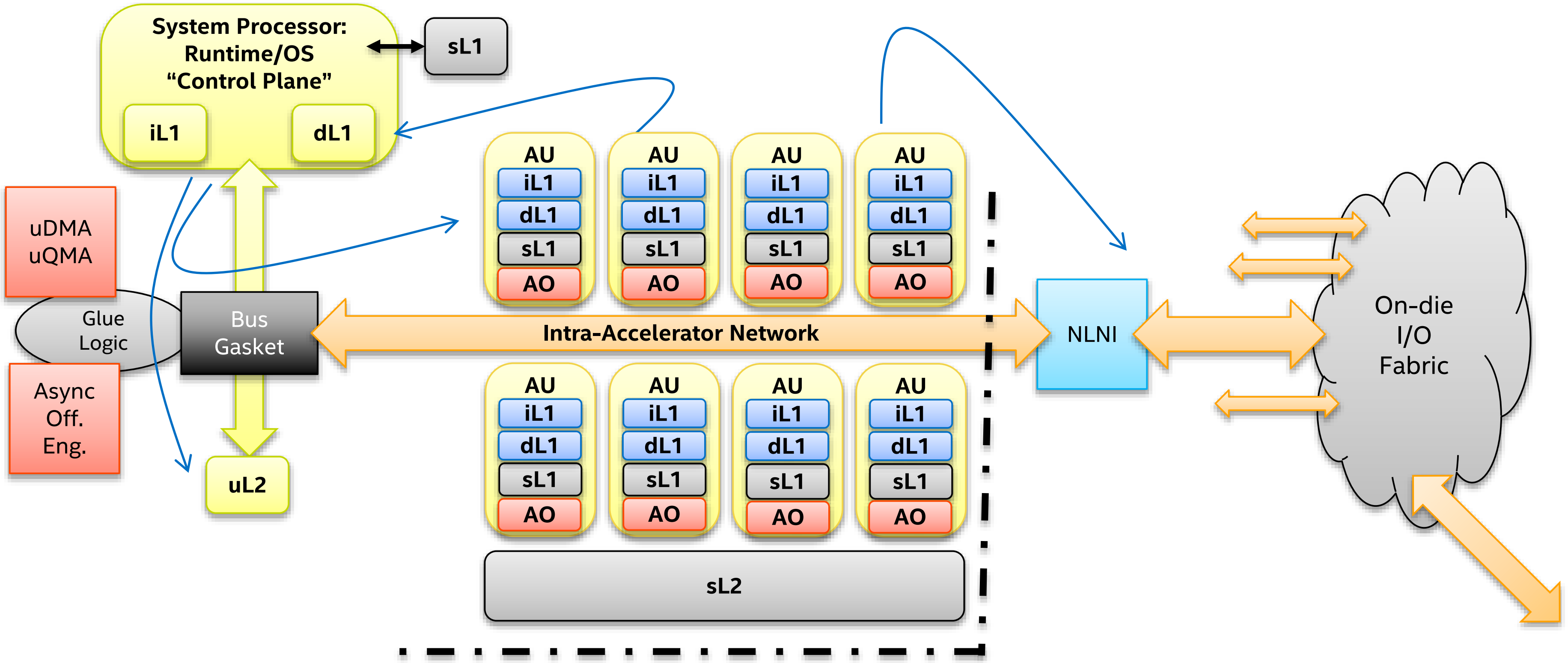
- <thread root>
- hcRunWorker
- workerLoop
- hcWorkShift
- taskExecute
- qsortTask
- ocrEdtCreate
- createEdtHelper
- newTaskHc
- createEventHelper
- newEventHc
- hcPolicyDomainProcess
- ptrCreateGuid
- hcMemAlloc
- allocateDatablock
- mallocProxyAllocate
- malloc

12M of 38M

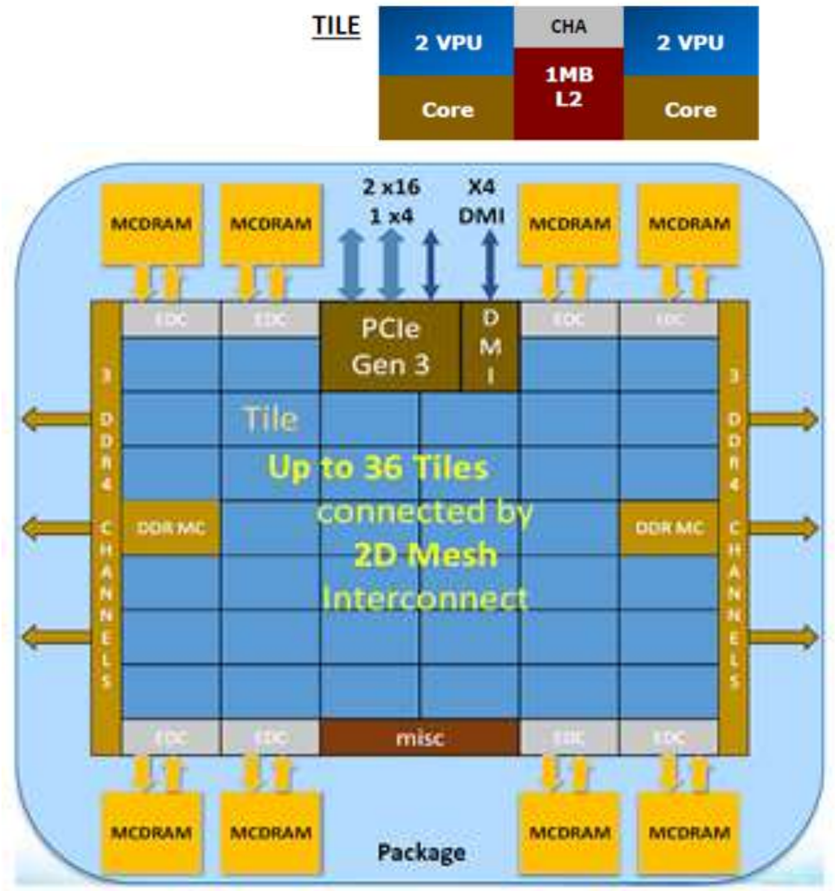
Software Ecosystem: Support today, find tomorrow



OCR and SoC Methodology: Co-Design



Knights Landing Overview



KNL presentation at Hotchips '15

Three modes

- **Self-boot** processor
- Self-boot w/ **integrated fabric**
- **Co-processor** (PCIe addon card)

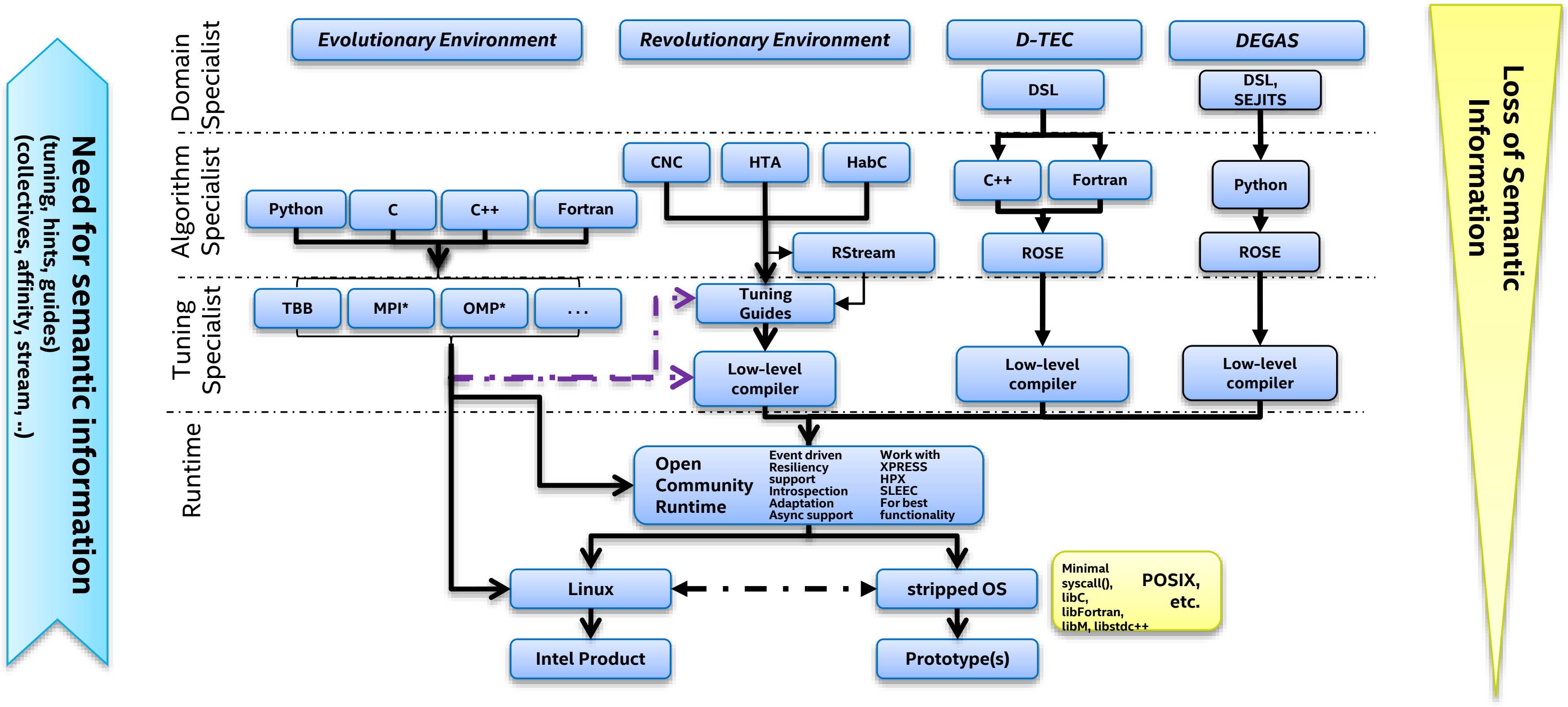
MCDRAM: three memory modes

- **Flat –** entirely addressable
- **Cache –** on DDR, direct-mapped
- **Hybrid –** part cache, part memory

Cluster modes (cc mesh interconnect)

- **All-to-all**: address uniformly hashed
- **Quadrant**: software-transparent, address hashed to directory in the same quadrant as memory
- **Sub-NUMA**: exposed as 4 NUMA nodes

Software Ecosystem: Support today, find tomorrow



Future Work: In progress now

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: Just starting or under consideration

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

Progress Report Card

Technology	Rating	Comments
CnC	**Promising**	CnC specification and reference are open-sourced by Intel; Rice is now driving changes and enhancements; concern over viability for large-scale applications remains
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	**Promising**	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. Newer HC++ has more features, but has the same underlying challenges.
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)	**Promising**	A version derived from the open reference implementation with a focus on performance first. Source code is not available.
OCR (Vienna)	**Promising**	Working on distributed version for larger capabilities; a rapid evaluation framework that has been demonstrated as easy to test new ideas; fully standard compliant; open source.
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.

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

Publications

- *Kavitha Chandrasekar, Bala Seshasayee, Ada Gavrilovska, and Karsten Schwan,* "Task characterization-driven scheduling of multiple applications in a task-based runtime", in Proceedings of the First International Workshop on Extreme Scale Programming Models and Middleware (ESPM 2015).
- *Kavitha Chandrasekar, Balasubramanian Seshasayee, Ada Gavrilovska, Karsten Schwan*, "Improving data reuse in co-located applications with progress-driven scheduling" at RESPA Workshop, co-located with SC '15.
- *Z. Budimlic, V. Cave, S. Chatterjee, R. ́Cledat, V. Sarkar, B. Seshasayee, R. Surendran, N. Vrvilo*, " Characterizing Application Execution using the Open Community Runtime", at RESPA Workshop Co-located with SC15
- *J. Dokulil, M. Sandrieser, S. Benkner*, "OCR-Vx – An Alternative Implementation of the Open Community Runtime", at RESPA Workshop Co-located with SC15
- *N. Vrvilo, R. Cledat*, "Implementing a High-level Tuning Language on the Open Community Runtime: Experience Report", at RESPA Workshop Co-located with SC15
- *M. Baskaran, B. Meister, T. Henretty, S. Tavarageri, B. Pradelle, A. Konstantinidis, R. Lethin,* "Automatic Code Generation for an Asynchronous Task-based Runtime", at RESPA Workshop Co-located with SC15
- *Nick Vrvilo*, "CnC for Tuning Hints on OCR", at CnC Workshop 2015
- *J. Dokulil, S. Benkner,* "Retargeting of the Open Community Runtime to Intel Xeon Phi", at ALCHEMY Workshop, co-located with ICCS 2015.
- *Suetterlein JD, JB Landwehr, JB Manzano Franco, and A Marquez*, "Toward A Unified HPC and Big Data Runtime." White paper and presentation at the STREAM 2015 workshop, Indianapolis, IN, October 27 - 28, 2015
- *Landwehr JB, A Marquez, JD Suetterlein, JB Manzano Franco, and C Plata*, "PNNL's Open Community Runtime - P-OCR: An asynchronous event-driven runtime that finally flies." Poster presented at DOE booth at the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 15), Austin, Texas, Nov. 15 - 20, 2015.
- *Landwehr JB, JD Suetterlein, A Marquez, JB Manzano Franco, and GR Gao*, "Application Characterization at Scale: Lessons learned from developing a distributed Open Community Runtime system for High Performance Computing." Accepted to the ACM International Conference on Computing Frontiers 2016. Como, Italy, May 16 - 18, 2016.
- *Suetterlein JD, JB Landwehr, A Marquez, JB Manzano Franco, and GR Gao,* "Asynchronous Runtimes in Action: An Introspective Framework for a Next Gen Runtime." Accepted to the IPDRM 2016: First Annual Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware co-Located with IPDPS 2016, Chicago, IL, May 23, 2016.
- *Nicolas Vasilache, Muthu Baskaran, Tom Henretty, Benoit Meister, M. Harper Langston, Sanket Tavarageri, Richard Lethin*, "A Tale of Three Runtimes", arXiv:1409.1914, September 2014.
- *Benoit Meister, Muthu Baskaran, Benoit Pradelle, Thomas Henretty, Richard Lethin*, "Efficient Compilation to Event-Driven Task Programs", arXiv:1601.05458, January 2016.

Demos – Tonight (or anytime you like)

- OCR – Intel and PNNL versions
- Applications – legacy under MPI-Lite and revolutionary (OCR native)
  - Running as large as NERSC lets us do so tonight
- Analyzers and tools – visualization, understanding, debugging
- Posters, Ad-hoc discussions
- Several key OCR & Apps team members present to deep-dive and cover Q&A
  - And we are here to collaborate – interested in all the projects
- All materials publicly available from git repositories

http://xstack.exascale-tech.com/