Traleika Glacier (X-Stack)

https://sites.google.com/site/traleikaglacierxstack/

The Intel Team

September 18, 2012

Goals and Objectives

Goal:

Research and mature software technologies addressing major Exascale challenges and get ready to intercept by 2018-2020

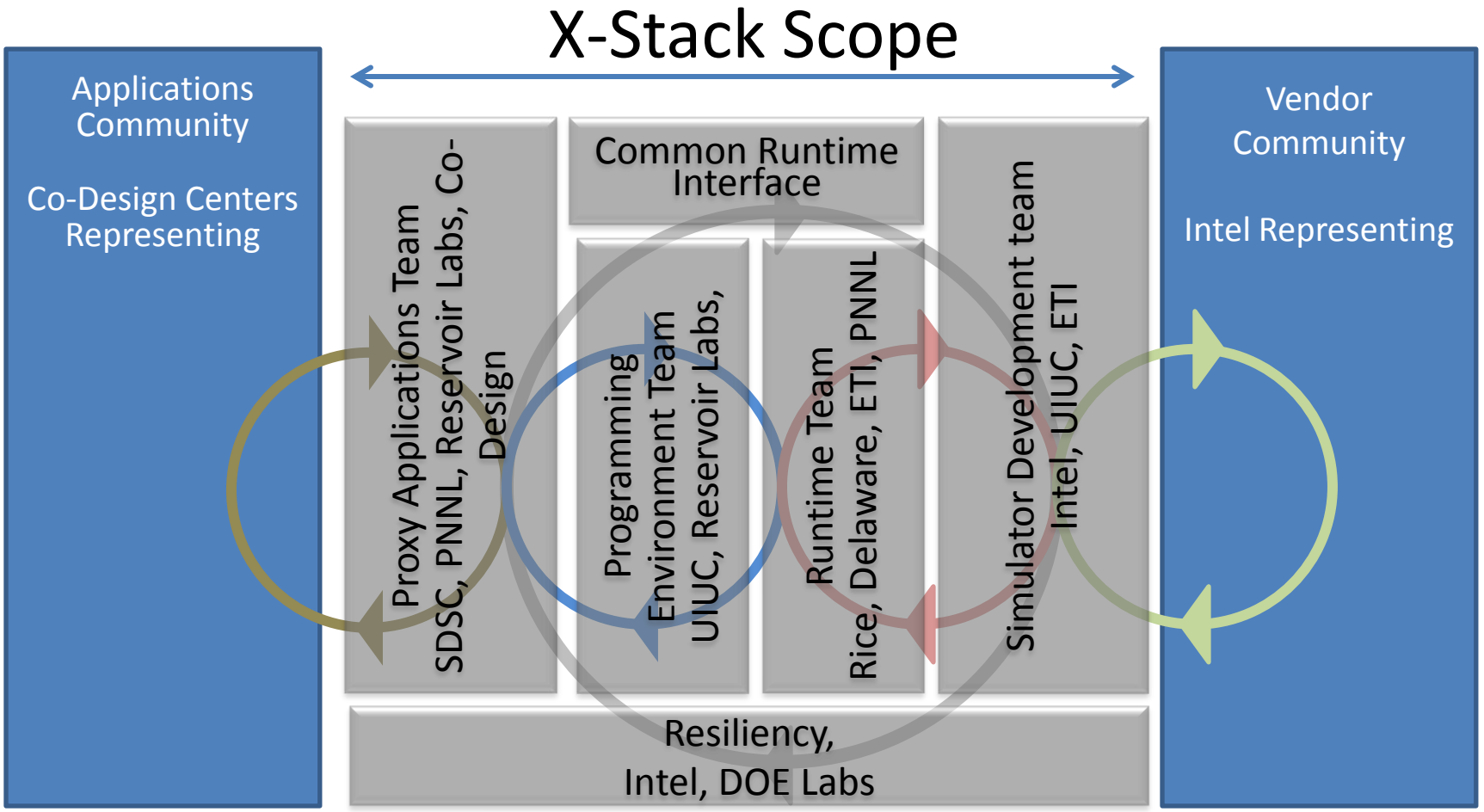
Objectives:

Energy efficiency	SW components interoperate, harmonize, exploit HW features, and optimize the system for energy efficiency
Data locality	PGM system & system SW optimize to reduce data movement
Scalability	SW components scalable, portable to $O(10^9)$—extreme parallelism
Programmability	New (Codelet) & legacy (MPI), with gentle slope for productivity
Execution model	Objective function based, dynamic, global system optimization
Self-awareness	Dynamically respond to changing conditions and demands
Resiliency	Asymptotically provide reliability of N-modular redundancy using HW/SW co-design; HW detection, SW correction

Principals

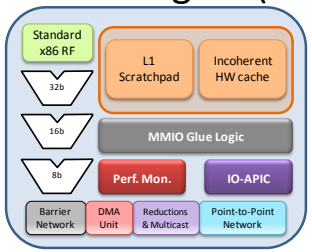
Name	Institution	Roles and Responsibilities
Shekhar Borkar	Intel	PI, Hardware guidance, HW/SW co-design, resiliency
Wilf Pinfold	Technical management	Technical program manager
Richard Lethin	Reservoir Labs	Programming system, R-Stream, tools, optimization
Rishi Khan	ET International	Simulators, execution model and runtime support
Prof. Guang Gao	University of Delaware	Execution model research
Prof. Laura Carrington	UC San Diego	Applications
Prof. Vivek Sarkar	Rice University	Programming system, runtime system
Prof. David Padua	University of Illinois	Programming system, Hierarchical Tiles Arrays (HTA)
Prof. Josep Torrellas	University of Illinois	Architecture, system architecture evaluation
John Feo	PNNL	Kernels and proxy apps for evaluation
Jackie Chen	Sandia National Lab	Co-design lead, combustion proxy app

Scope of Traleika Glacier Project



Straw-man System Architecture and Evaluation

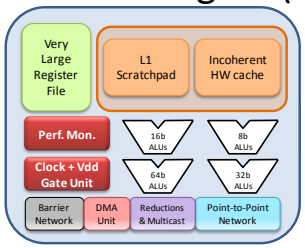
Control Engine (CE)



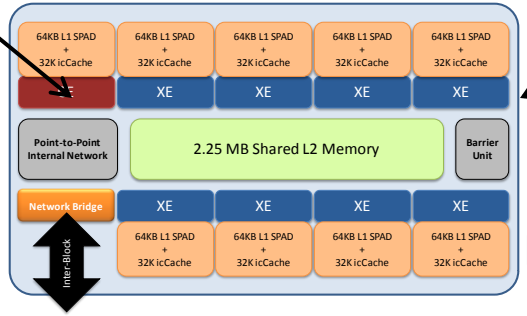
Optimized for execution model and resiliency

Optimized for apps
Large local stores
for data locality

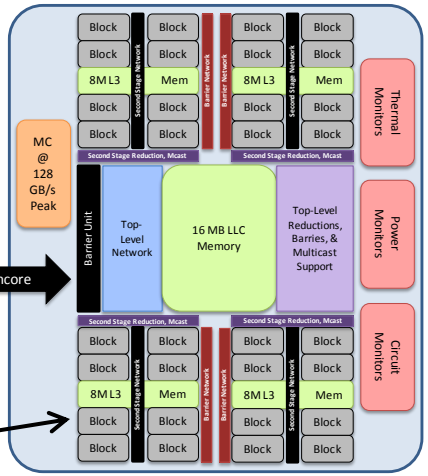
Execution Engine (XE)



Block



Processor Chip

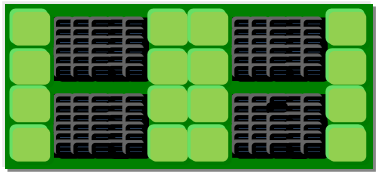


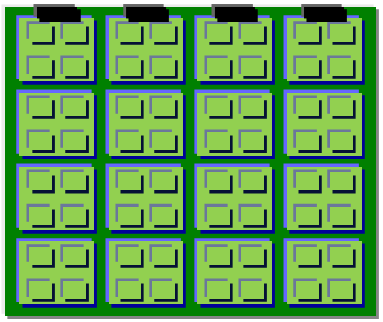
Uncore

Hierarchical, heterogeneous interconnect

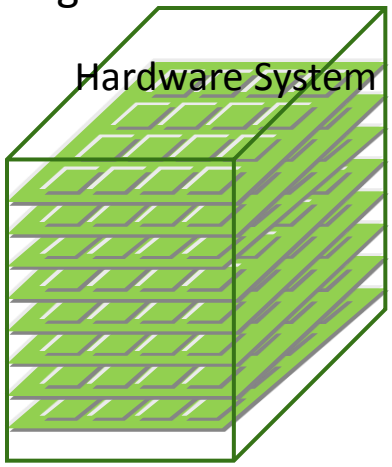
Large local stores (locality)
Sensors for self-awareness
Fine grain energy management

Processor Node with DRAM



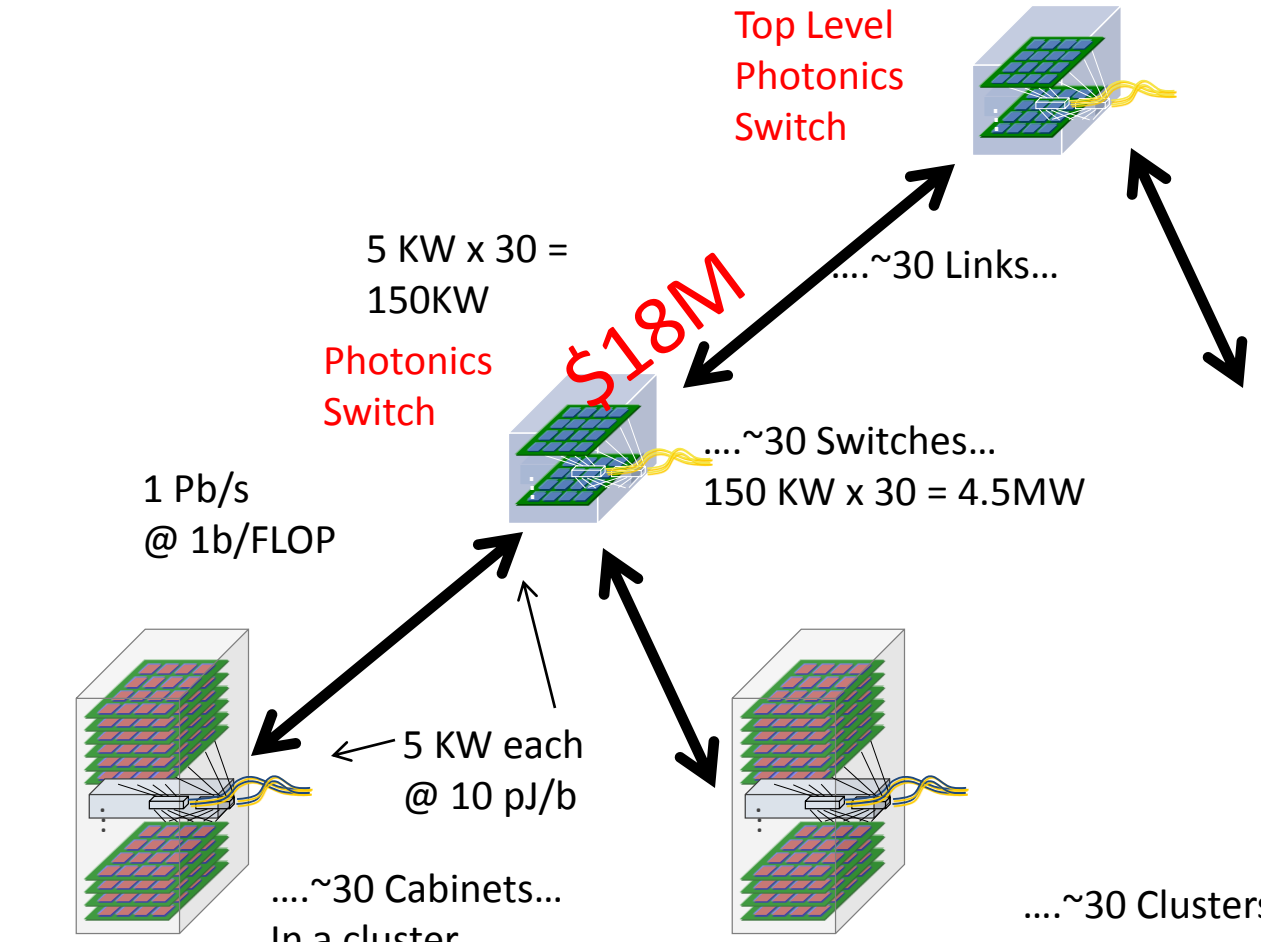


Hardware System



Architecture embraces data-locality, and tapered BW

Data-locality and BW Tapering, Why So Important?

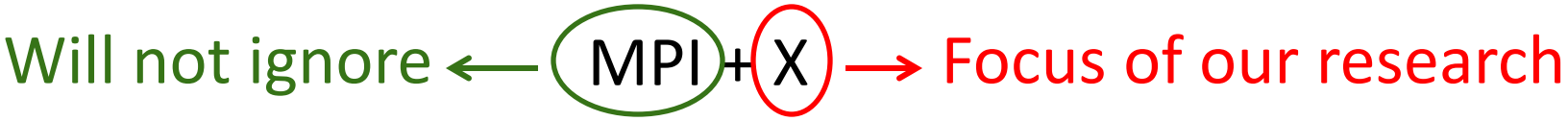


$540M

What is a reasonable tapering ratio?

What is required and what is affordable?

Programming & Execution Model



Programming model:

Separation of concerns: Domain specification & HW mapping

Express data locality with hierarchical tiling

Global, shared, non-coherent address space

Optimization and auto generation of codelets (HW specific)

Execution model:

Dataflow inspired, tiny codelets (self contained)

Dynamic, event-driven scheduling, non-blocking

Dynamic decision to move computation to data

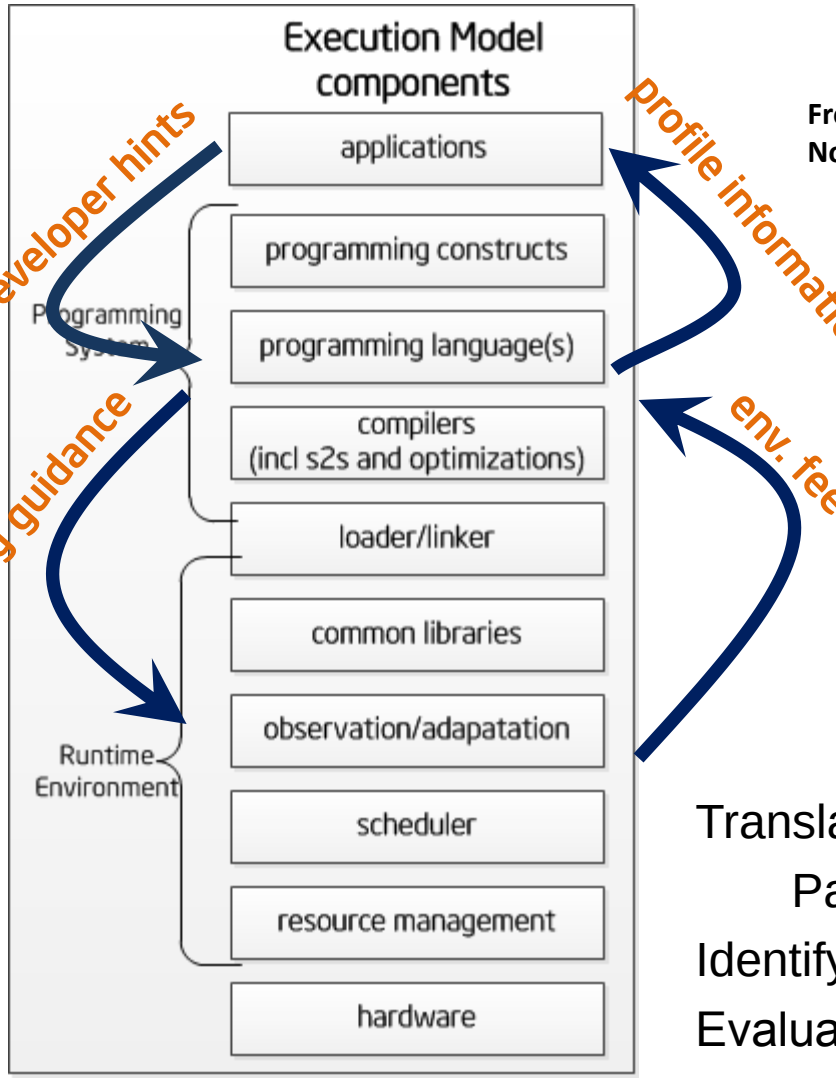
Observation based adaption (self-awareness)

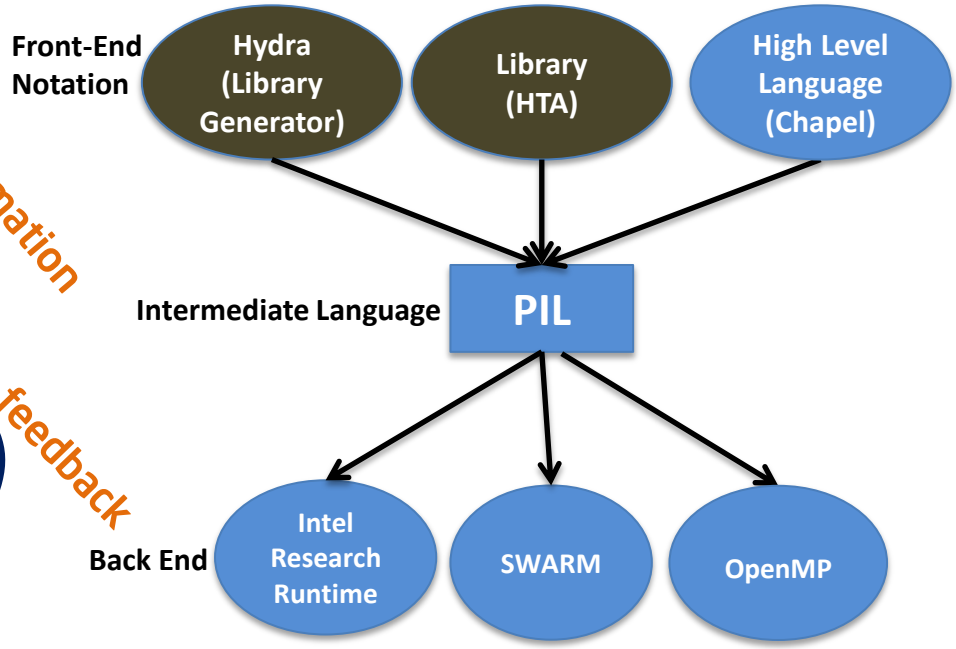
Implemented in the runtime environment

Separation of concerns:

User application, control, and resource management

Programming System Components





Translate parallel languages into codelets

Parallel Intermediate Language (PIL)

Identify new programming constructs

Evaluate: Performance, Data Locality, Energy,…

Runtime

Different runtimes target different aspects

- IRR: targeted for Intel Straw-man architecture
- SWARM: runtime for a wide range of parallel machines
- DAR^{3}TS: explore codelet PXM using portable C++
- Habanero-C: interfaces IRR, tie-in to CnC

All explore related aspects of the codelet Program Exec Model (PXM)

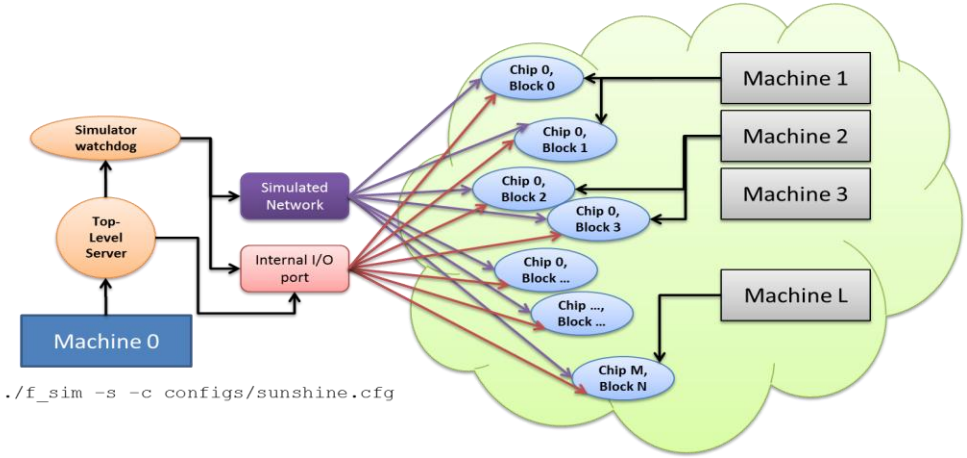
Goal: Converge towards Open Collaborative Runtime (OCR)

- Enabling technology development for codelet execution
- Model systems, foster novel runtime systems research

Greater visibility through SW stack ➔ efficient computing

- Break OS/Runtime information firewall

Some Promising Results

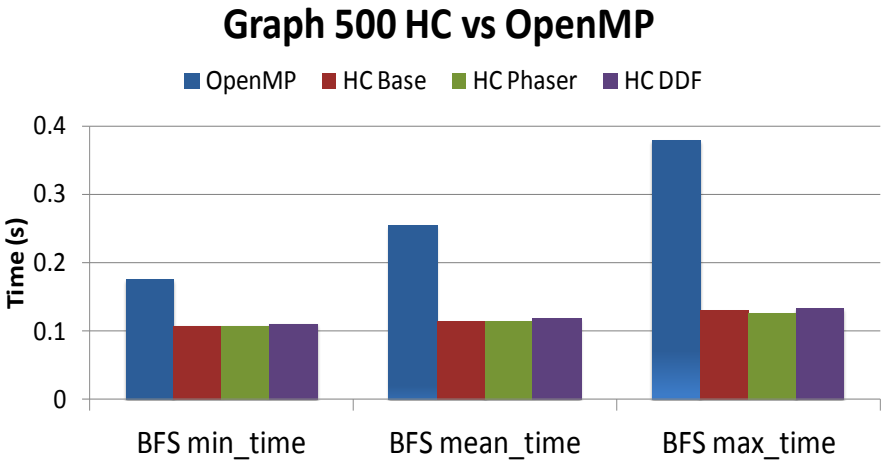


IRR

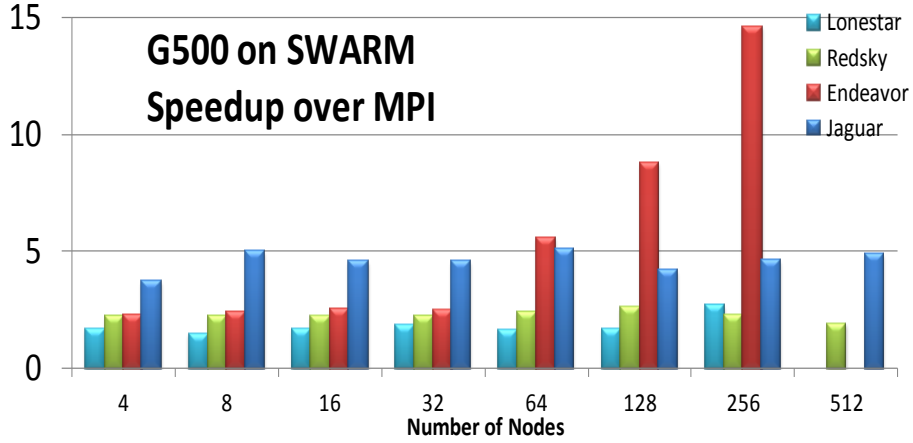
Simulated 256 XE and 32 CE cores

Runs apps and architecture evaluations

Habanero



SWARM



Runtime Research Agenda

- **Locality aware scheduling—heuristics for locality/E-efficiency**
  - Extensions to standard Habanero-C runtime
- **Adaptive boosting and idling of hardware**
  - Avoid energy expensive unsuccessful steals that perform no work
  - Turbo mode for a core executing serial code
  - Fine grain resource (including energy) management
- **Dynamic data-block movement**
  - Co-locate codelets and data
  - Move codelets to data
- **Introspection and dynamic optimization**
  - Performance counters, sensors provide real time information
  - Optimization of the system for user defined objective
  - (Go beyond energy proportional computing)

Simulators and Tools, Leverage UHPC, and Go Beyond

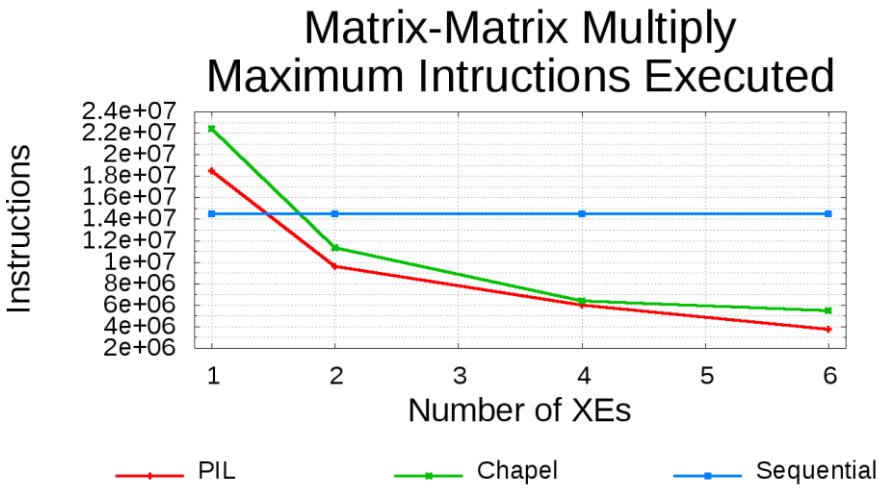
Simulator	Pros	Cons
AFL Application faking Library IRR API + Extensions	• 64-bit Linux IRR implementation • Near-native application code execution on host processor • Rapid application development • Epoch statistics as well as total statistics	• Does not model real architecture • Does not model advanced ISA features • Does not reflect expected timing of simulated system
FSim Functional Simulator	• Every hardware unit modeled, exact memory and network hierarchies including messages • Complete statistics and trace file • >10 MIPS per core speed • Massively parallel and distributed, limited only by machine pool	• Does not have a timing model • Lower speed, highly detailed

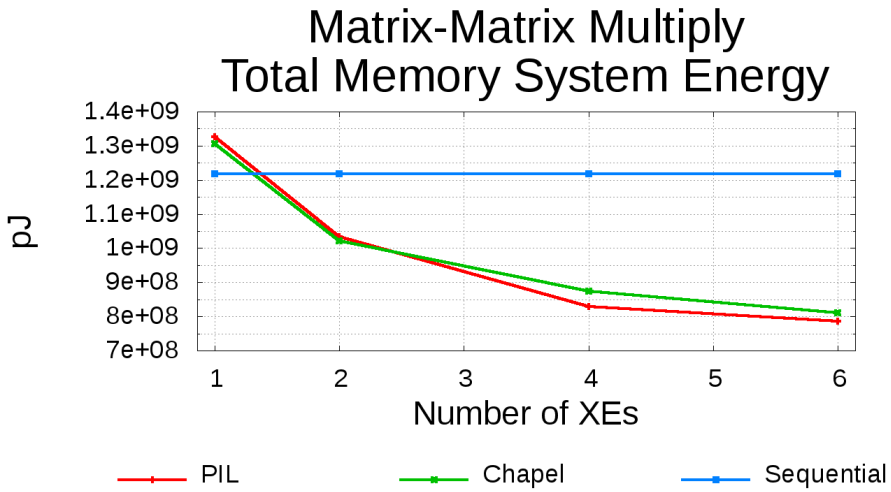
Tool	Purpose	Advantage	Weakness
Power Estimator	• Uses statistics/counters to make energy and area estimates for application behavior	• Scales from 45nm to 7nm projections • Automatic analysis of outputs from FSim runs	• Only models dynamic power, uses circuit models for leakage • Calibrated to existing commercial devices
Memory Analyzer	• Detailed models for cache and/or scratchpad hierarchies, various levels & types of coherence • Compares configurations	• Calls into Power Estimator • Enables limited AFL trace power estimation	• Does not model Instruction fetch/execution • Limited to AFL-compatible memory traces at this time

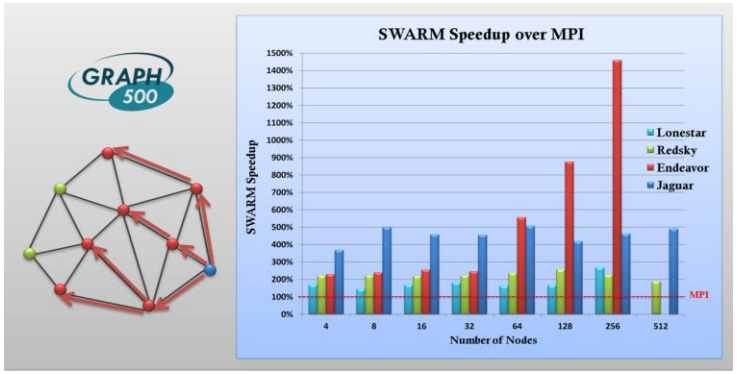
Simulators—what to expect and not

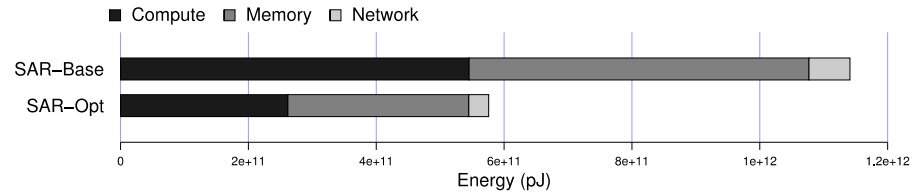
Evaluation of architecture features for PGM and EXE models	
Relative comparison of performance, energy	
Data movement patterns to memory and interconnect	
Relative evaluation of resource management techniques	
Expect	Not
Relative performance	Absolute system performance
Relative power and energy	System power and energy
Evaluation of core features	Absolute core performance and energy
Relative ease of programmability	Programming productivity

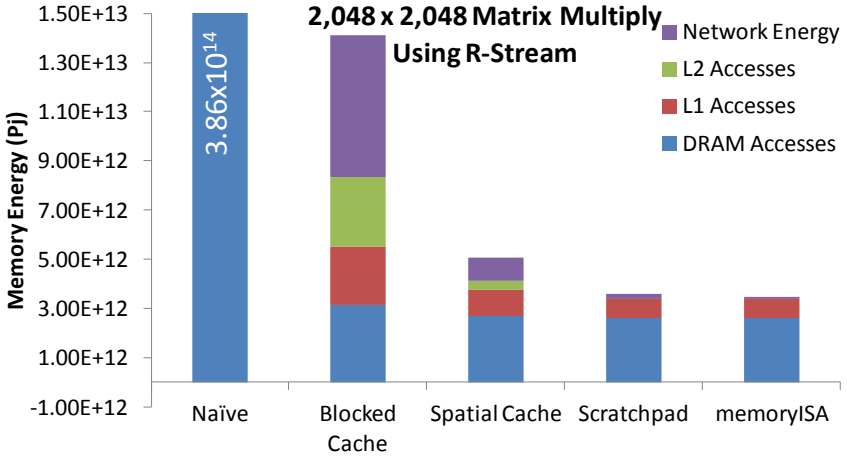
A Few Results using the Simulators





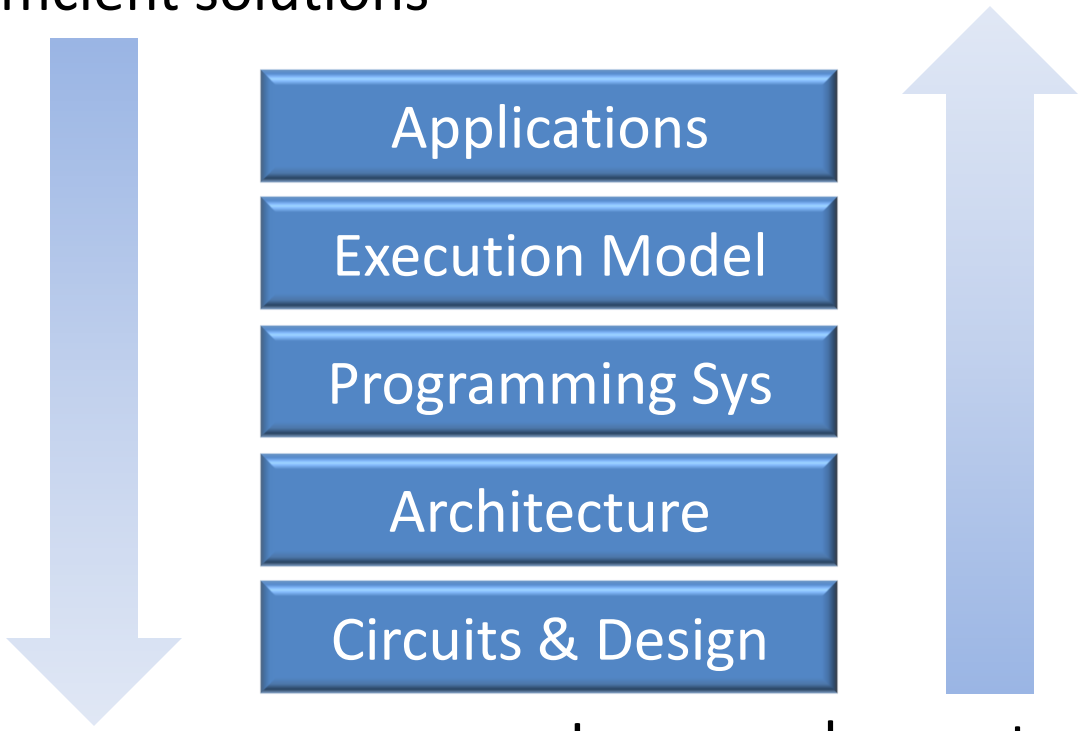






Applications and HW-SW Codesign

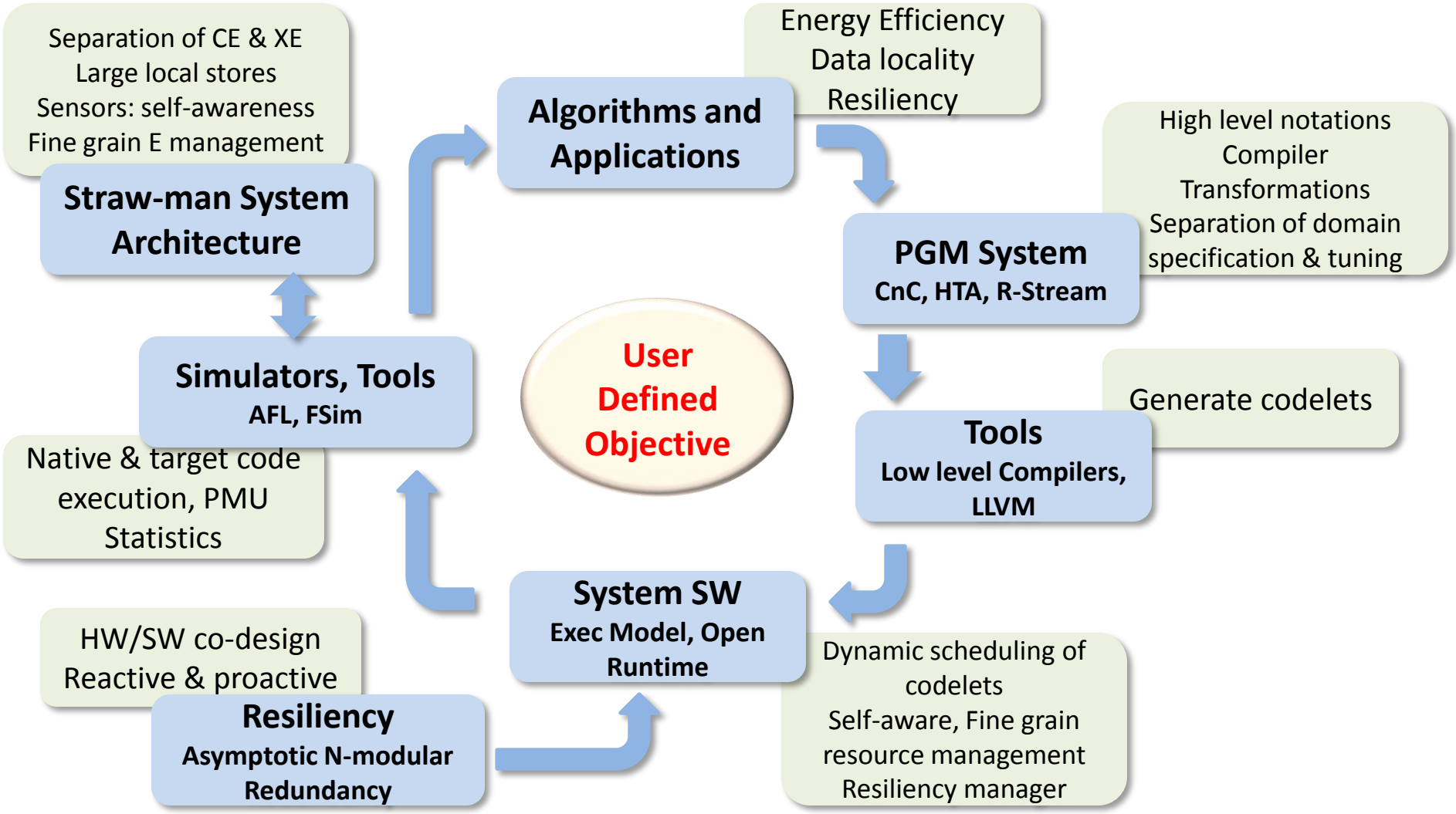
Analysis of applications to devise
the most efficient solutions



Issues and opportunities to exploit

Co-design centers invaluable for applications expertise

X-Stack Components Put Together



Metrics

Impact Matrix						
	Algorithm	**Apps**	**Pgm Sys.**	**Exec M.**	**Tools**	**Arch. ISA**
Energy	H	L	M	H	H	H
Data Locality	M	M	H	H	M	L
Parallel Fraction	H	M	M	M	L	M
Programmability	M	L	H	M	L	M
FIT Rate	L	L	L	H	L	L
Tapering Pressure	M	L	M	H	L	M
Legacy Scalability	M	H	M	H	M	M
Future Scalability	M	H	M	H	M	H

High
Med
Low

Metric with Aggressive Goals (High Impact)						
	Algorithm	**Apps**	**Pgm Sys.**	**Exec M.**	**Tools**	**Arch. ISA**
Energy	50%			20pJ/Op	5MW/Exa	20pJ/Op
Data Locality			1 D bit/Op	3pJ/Op		
Parallel Fraction	5X					
Programmability			2X			
FIT Rate				> 6 days		
Tapering Pressure				> 4X/level		
Legacy Scalability		5X		MPI+X		
Future Scalability		O(5,000)		O(B) tasks		O(M) nodes

3 Year Roadmap

	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--->	----------	----------	----------	---------->

Arch Rev 1 (UHPC leveraged)

Arch Rev 2 incorporates X-Stack

Arch Rev 2.5 enhanced for X-Stack
with initial findings

Arch Rev 3.0 X-Stack final

Final Evaluation
Release OCR 1.0

Summary

- **All X-Stack components are in place, research agenda is set**
- **Will leverage research momentum created by the UHPC program**
- **Will leverage and provide community support**
- **Traleika Glacier team is well equipped, excited, and poised for successful execution**