0

Accelerating Open-Source Hardware Simulation using Optimistic Parallel Discrete Event Simulation

Maximilian Bremer, Nirmalendu Patra, Tan Nguyen, Cy Chan and Dilip Vasudevan*

***Applied Mathematics and Computational Research (AMCR)* Division, LBNL**

OSCAR Workshop,

June 18th. 2023

Maximilian Bremer
Research Scientist
mb2010@lbl.gov

Nirmalendu Patra
CS Engineer
nbp@lbl.gov

Tan Thanh Nhat Nguyen
Research Scientist
TanNguyen@lbl.gov

Cy Chan
Computer Research Scientist
cychan@lbl.gov

Dilip Vasudevan
Research Scientist
dilipv@lbl.gov

Contents

- **Motivation**
  - **Need for fast simulation of large scale systems**
- PARADISE Framework
  - End-to-End Device level to Architectural Simulation flow
- Devastator – Parallel Discrete Event System (PDES) Framework
  - Optimistic Synchronization based Parallel Discrete Event Simulation
- PARADISE++ for Large Scale Optimistic Simulations
  - SST simulation Framework
  - Recent Results
- Future work

Simulator host architectures are subject to same trends

Hopper (NERSC-6): 24 cores/node [3]

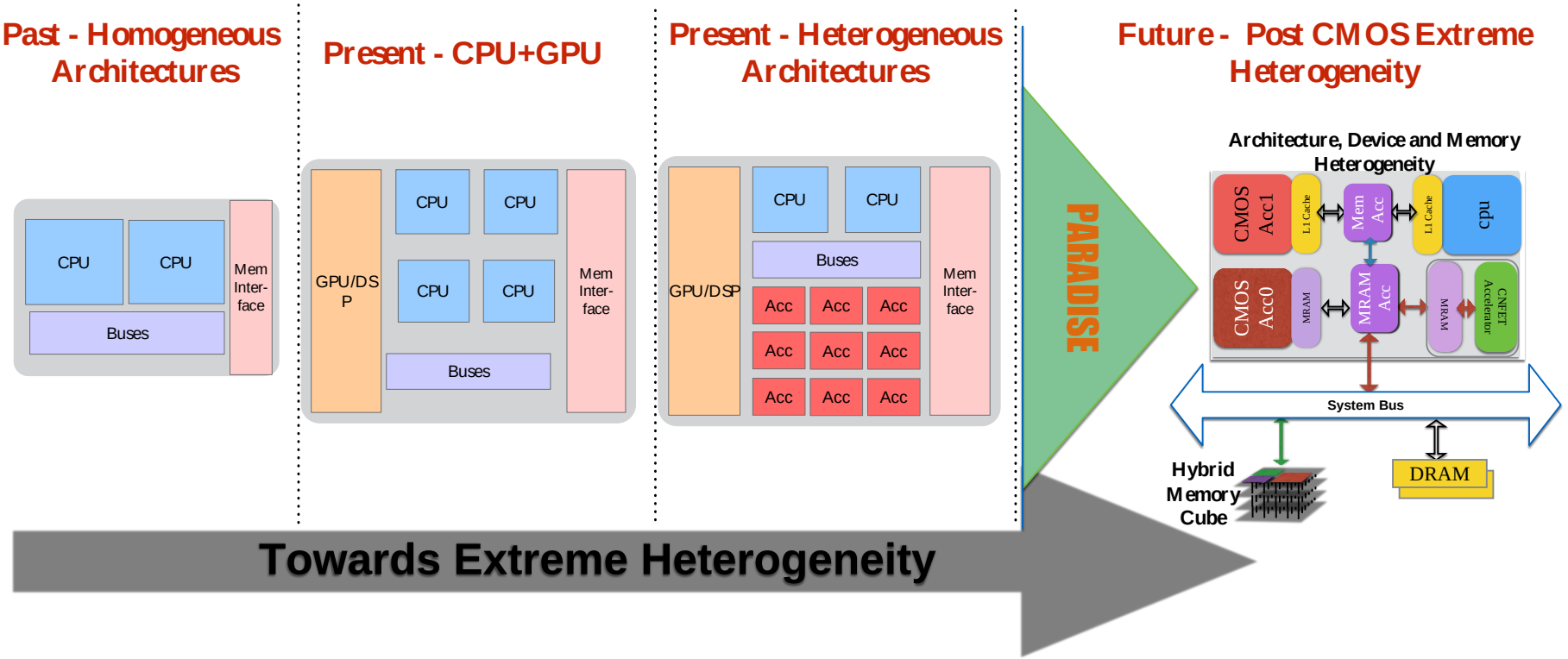
Perlmutter (NERSC-9): 128 cores/node [4]

Architectural simulators need to effectively utilize concurrency on modern supercomputers

[3] Jon Bashor. "NERSC's Hopper breaks petaflops barrier." NERSC. https://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2010/nersc-s-hopper-breaks-petaflops-barrier-ranks-5th-in-the-world/. Accessed 30 Jan. 2023.
[4] Kathy Kincade. "Berkeley Lab deploys next-gen supercomputer, Perlmutter, bolstering U.S. scientific research." NERSC. https://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2021/berkeley-lab-deploys-next-generation-supercomputer-perlmutter-bolstering-u-s-scientific-research/. Accessed 30 Jan. 2023.

Future of Computing Architecture Design

- **Past**
  - CPU + Mem + Bus
- **Present**
  - CPU,GPU
  - CPU, GPU, Accelerators
- **Future**
  - Device Heterogeneity
  - Memory Heterogeneity
  - Architectural Heterogeneity



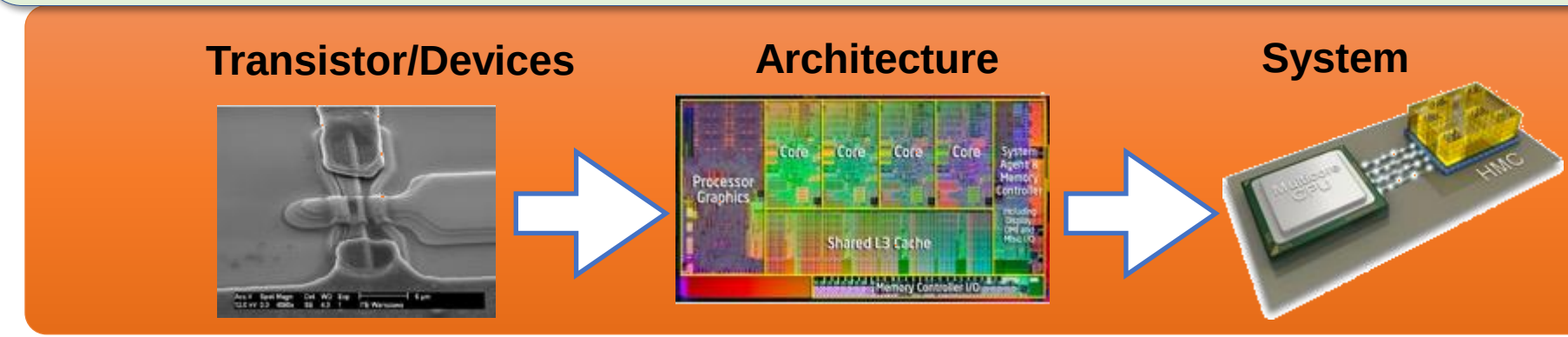
Dilip Vasudevan 2016

Architecture Modeling: PARADISE

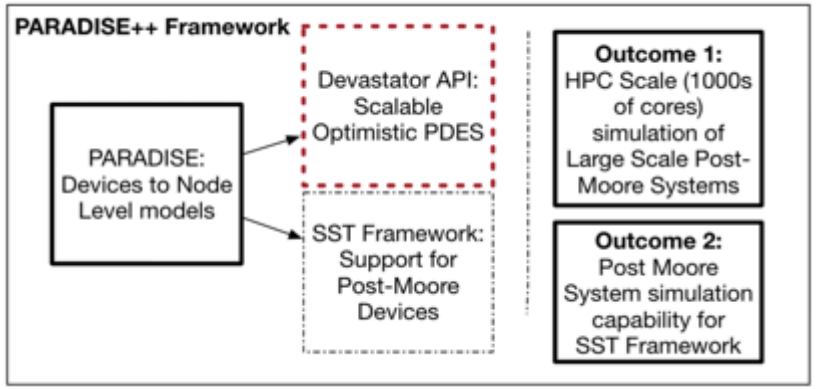
Post-Moore Architecture and Accelerator Design Space Exploration

- **Multiple devices, memories, and other "post Moore" technologies in development**
- **Evaluating each option in isolation misses big picture**
  - Devices can be better designed with high-level metrics
  - Architects can figure out how to best use new technologies

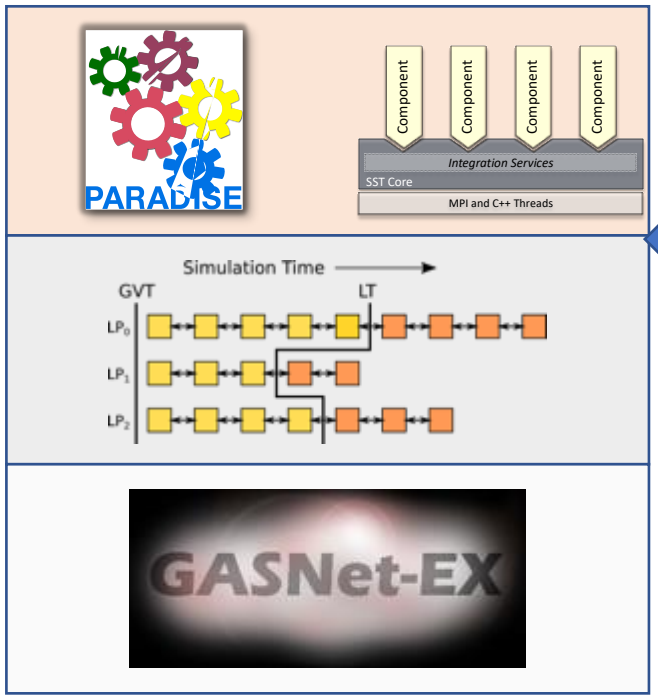
❑ ***Until now, we lacked the tools to do so systematically and rapidly for many technologies (PARADISE addresses that gap)***

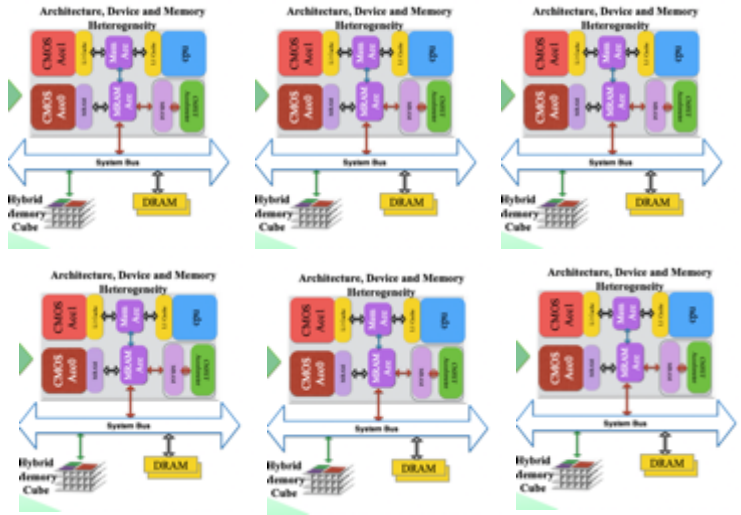


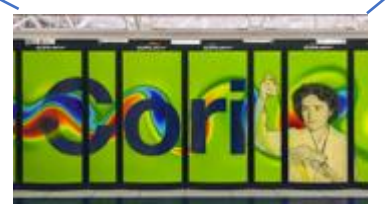
Technical Overview



- Post Moore Device level to architectural level simulation support
- Optimistic Synchronization based PDES simulation
- SST extension for Post Moore Architectural support





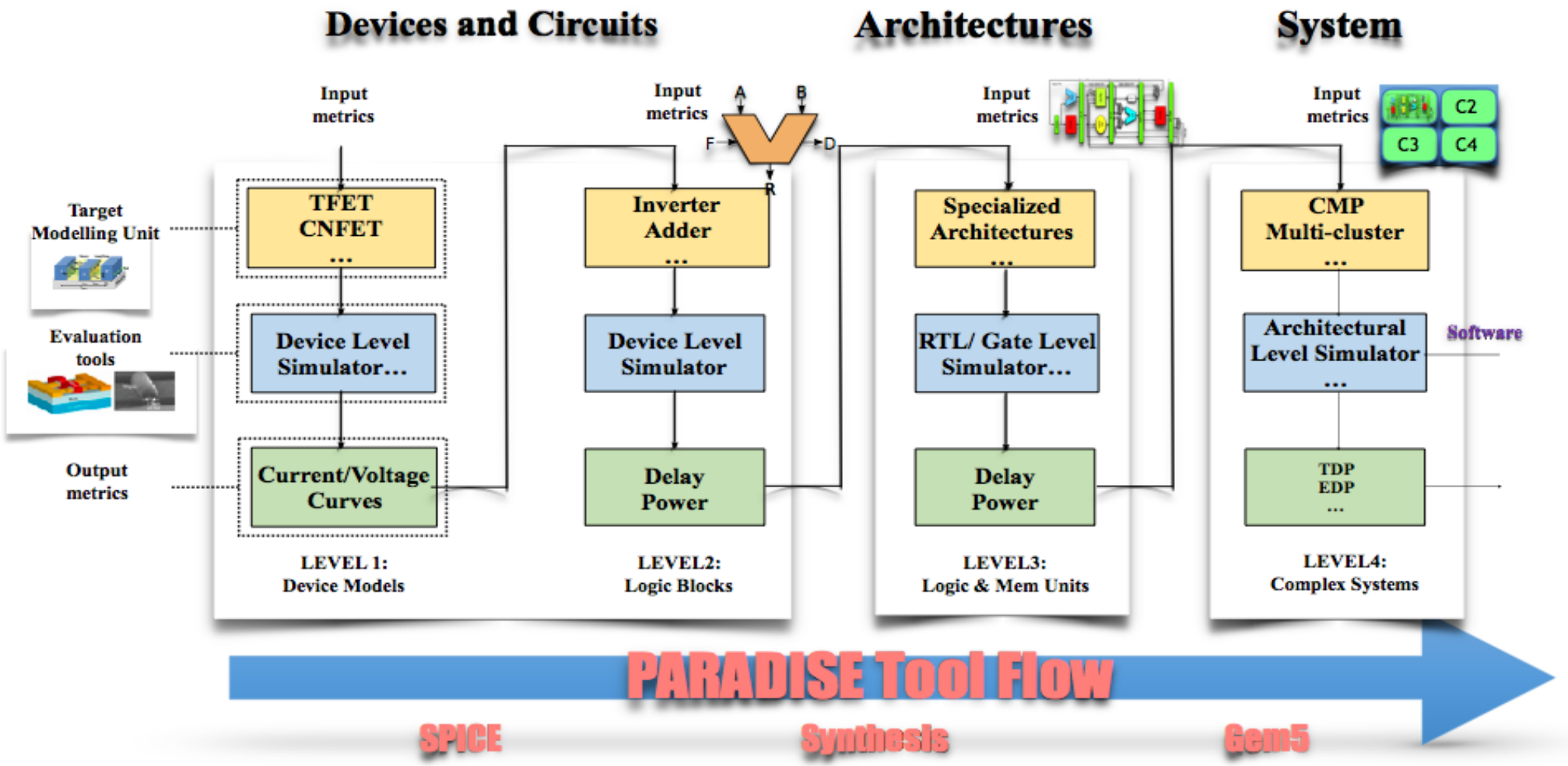


PARADISE++ simulation framework and its potential outcomes

Contents

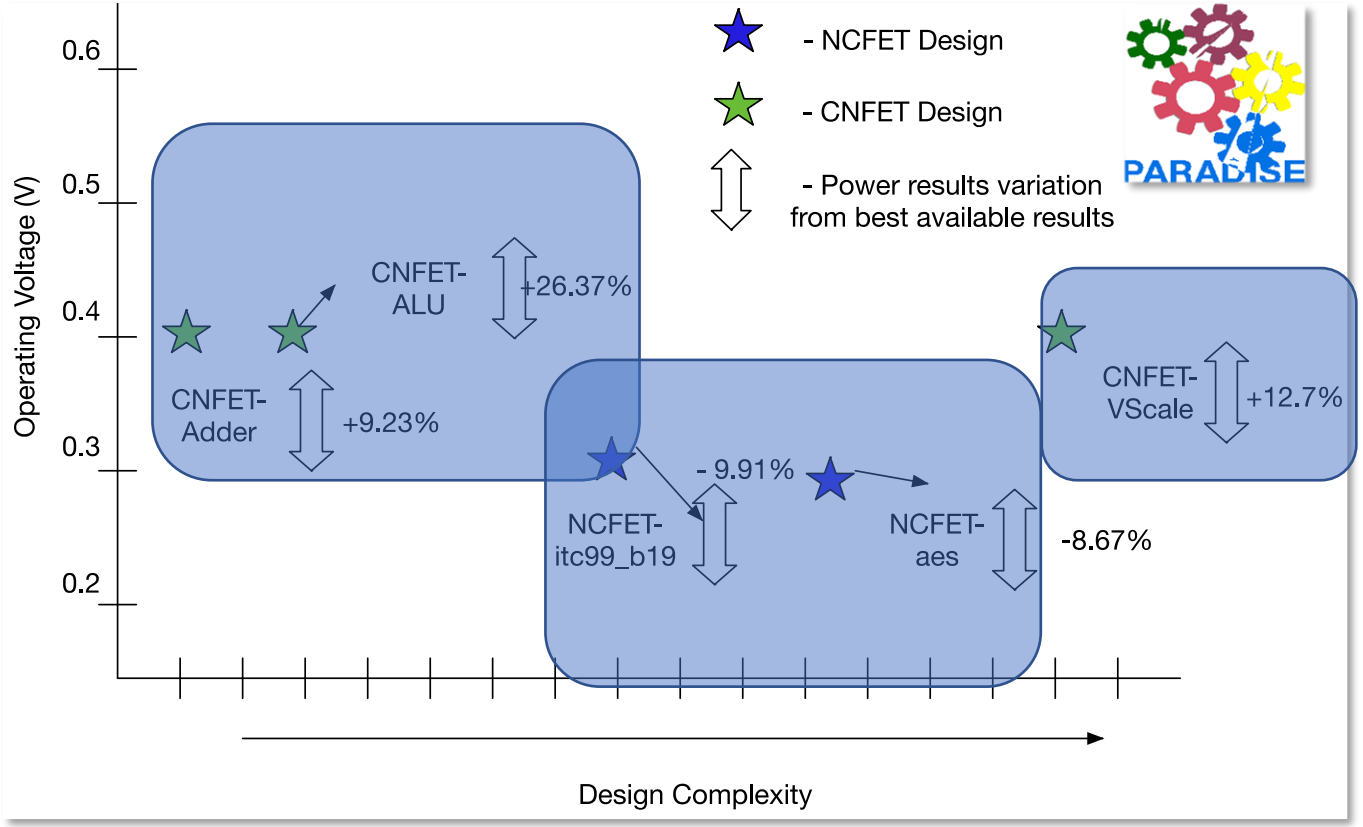
- Motivation
  - Need for fast simulation of large scale systems
- **PARADISE Framework**
  - **End-to-End Device level to Architectural Simulation flow**
- Devastator – PDES Framework
  - Optimistic Synchronization based Parallel Discrete Event Simulation
- PARADISE++ for Large Scale Optimistic Simulations
  - SST simulation Framework
  - Recent Results
- Ongoing tasks

PARADISE: End-to-End Modelling



Validation of Hierarchical Model Against Experimentally Collected Data

- Four device models (*validated against the best available published experimental results*)
- Accelerator hardware designs
  - Several computational cores

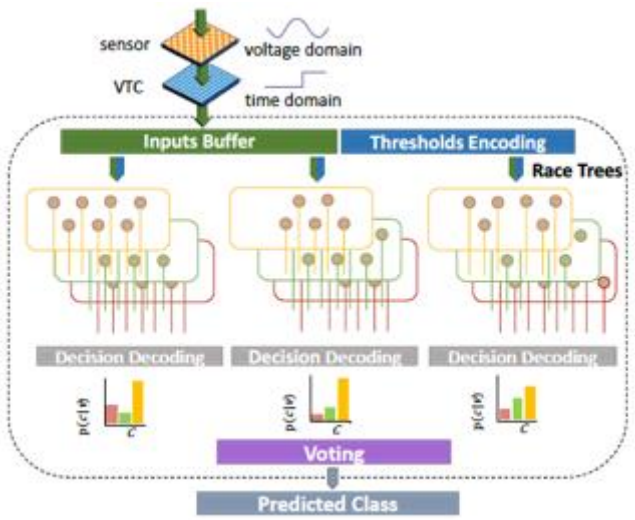


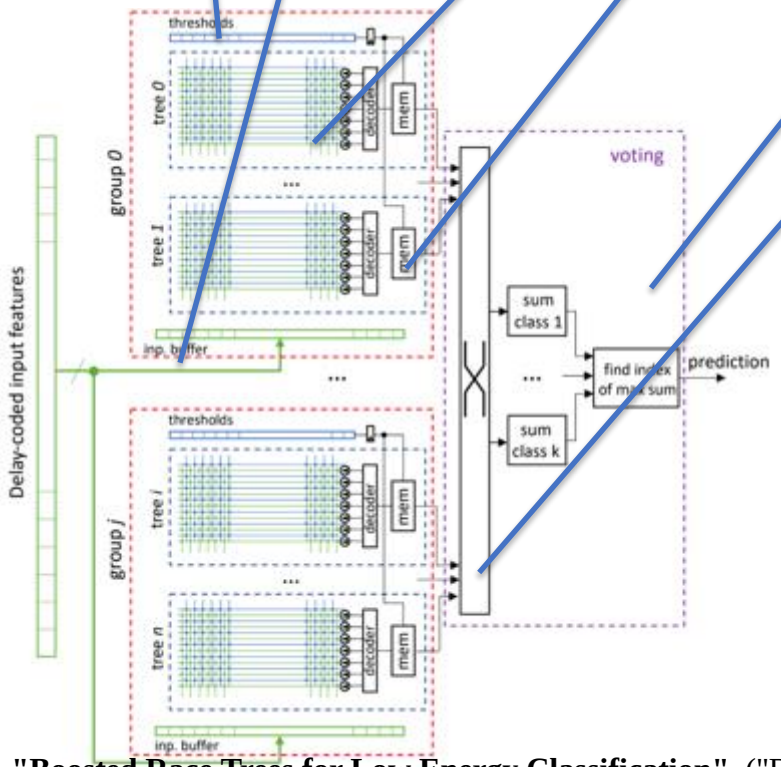
D. Vasudevan, G. Michelogiannakis, D. Donofrio and J. Shalf, "PARADISE - Post-Moore Architecture and Accelerator Design Space Exploration Using Device Level Simulation and Experiments," *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Madison, WI, USA, 2019, pp. 139-140.

14 nm CMOS evaluation with PARADISE

# Trees	Depth	Inp. res.	Mem. bits	Groups	Tech. node	Thresholds		Inp. Buffers		Trees & Dec.		Memory		Voting		Progr. Intercon.		Total	
						P (μW)	A (mm^2)	P (μW)	A (mm^2)	P (μW)	A (mm^2)	P (μW)	A (mm^2)	P (μW)	A (mm^2)	P (pW)	A (mm^2)	P (mW)	A (mm^2)
1,000	6	8	8	10	14nm	7,237	1.1e-2	21,999	3.5e-2	529,603	0.5	111,492	0.05	2,008	0.03	0.252	–	673	0.63
1,000	6	4	4	10	14nm	359	7.5e-4	21,999	3.5e-2	529,603	0.5	59,289	0.03	1,673	0.03	0.016	–	613	0.59
200	8	4	4	10	14nm	359	7.5e-4	21,999	3.5e-2	452,701	0.38	96,689	1.3e-2	321	5.8e-3	0.013	–	571	0.44
200	6	4	4	10	14nm	359	7.5e-4	21,999	3.5e-2	105,921	0.1	11,857	5.6e-3	321	5.8e-3	0.003	–	140	0.15

Boosted Race Trees System Architecture





- Component Level Evaluation with PARADISE
- For 14 nm CMOS technology
- Decision Trees for Machine Learning

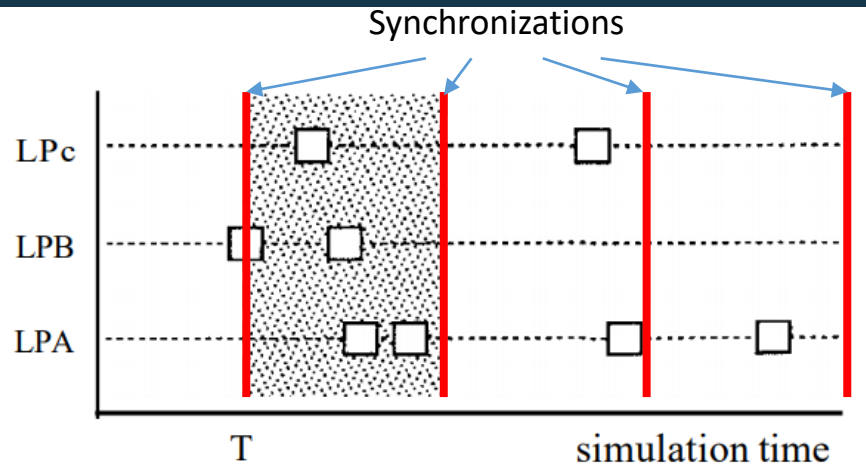
G Tzimpragos, T Sherwood, A Madhavan, D. Vasudevan, T. Sherwood and D Strukov, **"Boosted Race Trees for Low Energy Classification",** ("Best Paper Award"), ASPLOS 2019, April 2019

Contents

- Motivation
  - Need for fast simulation of large scale systems
- PARADISE Framework
  - End-to-End Device level to Architectural Simulation flow
- **Devastator – PDES Framework**
  - **Optimistic Synchronization based Parallel Discrete Event Simulation**
- PARADISE++ for Large Scale Optimistic Simulations
  - SST simulation Framework
  - Recent Results
- Ongoing tasks

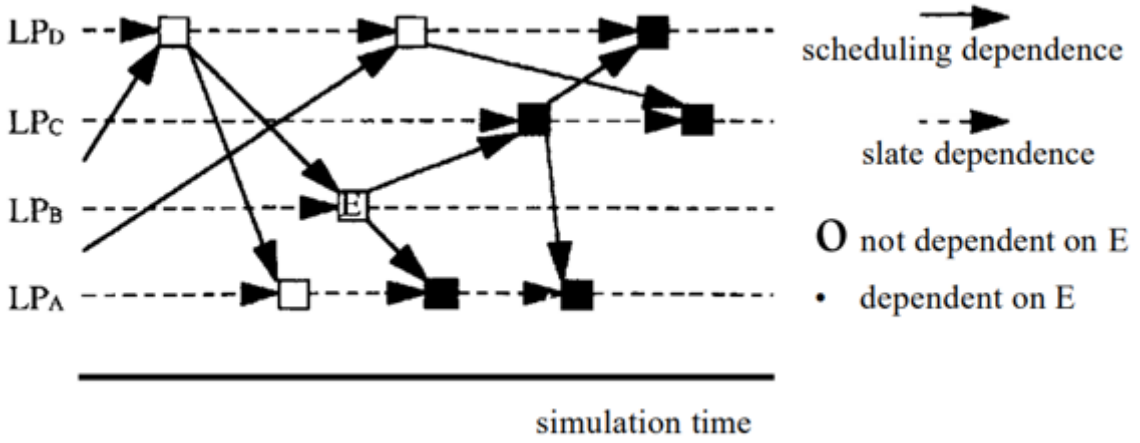
Proposed Evaluation of Optimistic Simulation for Paradise++

- Conservative approach:
  - Drawback: synchronizations may be expensive and prevent scalable performance
  - Advantageous if synchronization is cheap or if look-ahead interval is large, reducing the frequency of synchronization and enabling parallelism within each epoch



Conservative Approach

- Optimistic approach:
  - Advantageous if synchronization is expensive or look-ahead interval is small
  - Effective if remote events are rare relative to the characteristic look-ahead frequency
  - Drawback: requires actors to **support rollback** through **reversible computation or logging**

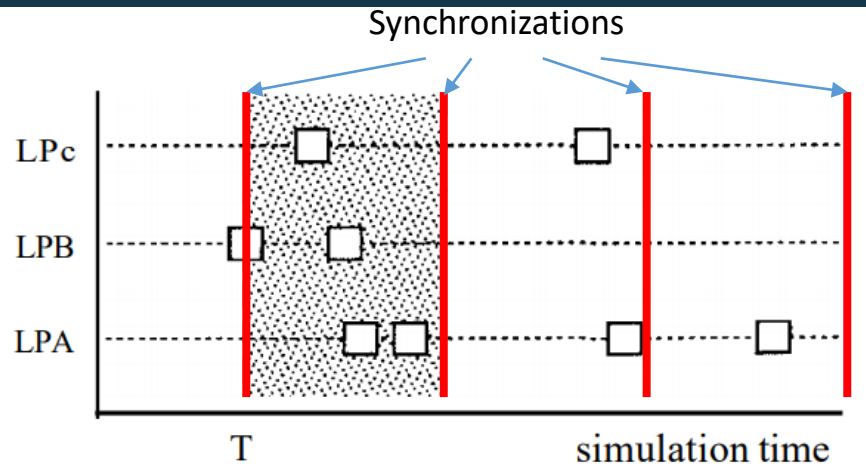


Optimistic Approach

Above Images: R. Fujimoto

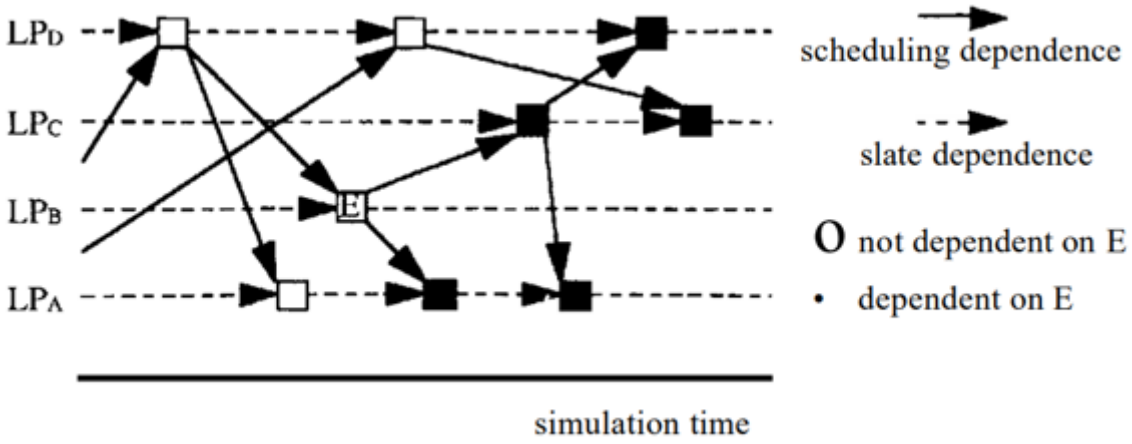
Proposed Evaluation of Optimistic Simulation for Paradise++

- Conservative approach:
  - Must synchronize every time an ordering violation could possibly happen
  - Components are parameterized with a *look-ahead* interval (i.e. distance/delay) where events within an epoch cannot produce new events on a remote actor within the same epoch
  - Actors block/synchronize before every interval epoch to ensure receipt of incoming events from neighbors



Conservative Approach

- Optimistic approach:
  - Avoids synchronization by optimistically assuming no ordering violation will occur
  - No look-ahead window or epochs, events may produce new events with arbitrarily increasing timestamps
  - Actors speculatively execute events in their local event queue, remote events are inserted into the queue, and events are rolled back if required

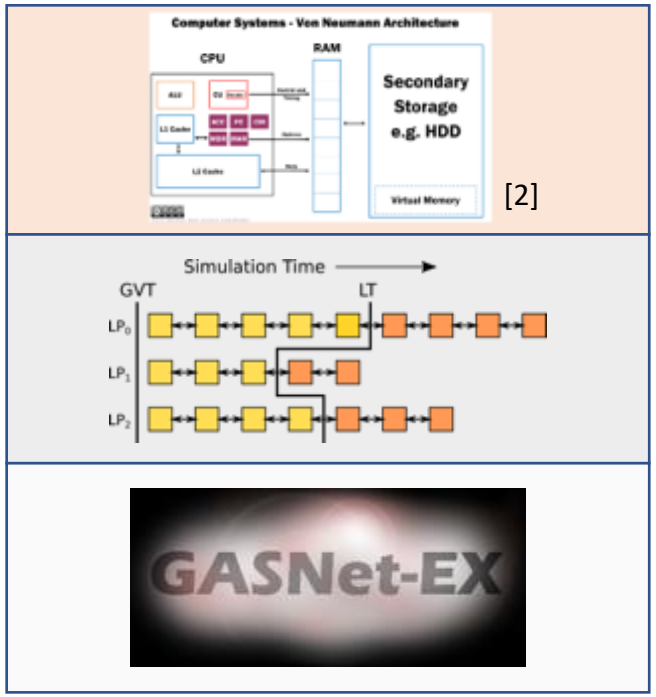


Optimistic Approach

Above Images: R. Fujimoto

Devastator Simulation Software Stack

Software Stack



Application: provides the domain-specific logic that defines the actors and events of a simulated system and determines the parallel domain decomposition mapping actors to ranks

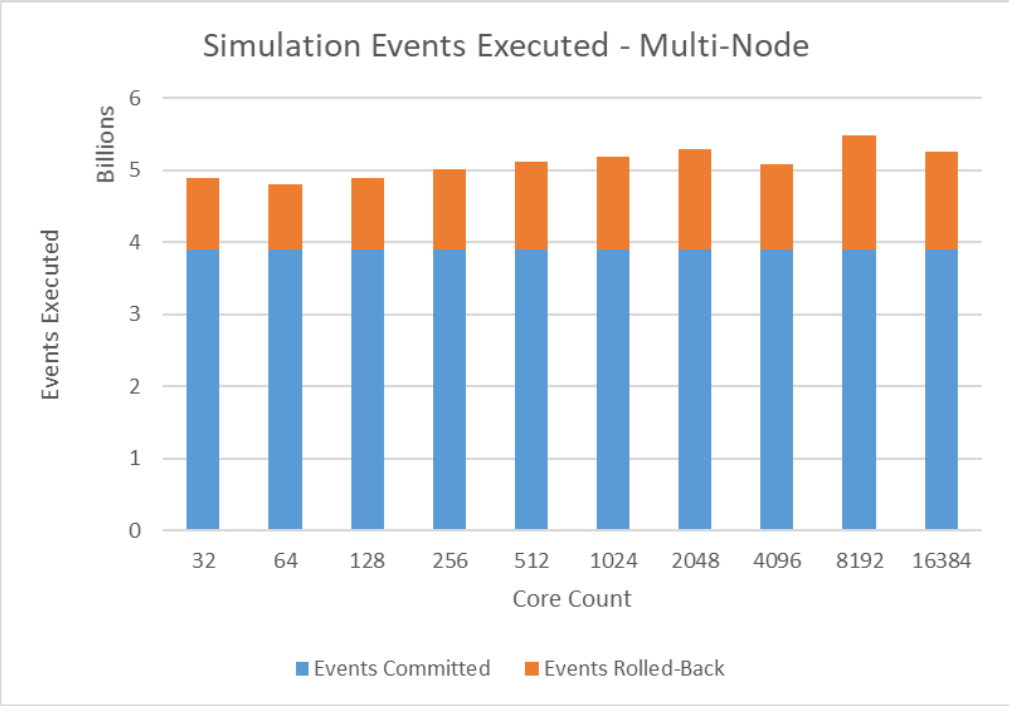
Devastator: C++14 PDES framework for modern programmers with a simple and easy API, implements Jefferson's Time Warp optimistic parallel discrete event protocol [1] to handle event scheduling, execution, rollback, commit, and global virtual time

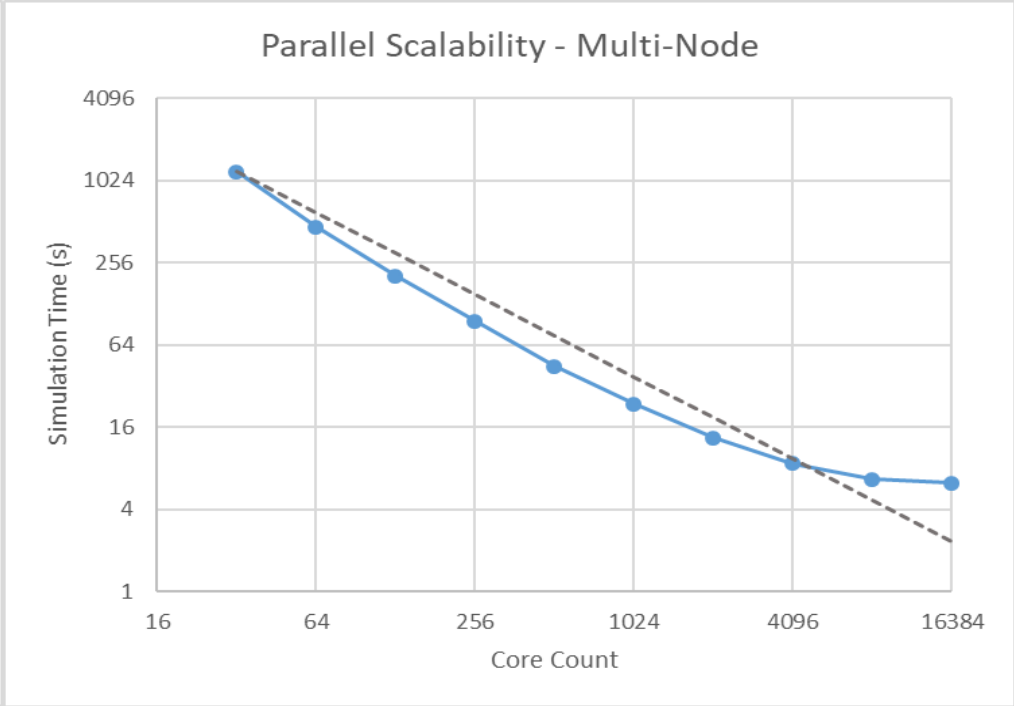
GASNet-Ex: provides high-performance inter-process communications across distributed memory, in particular for small active messages

[1] David R. Jefferson. 1985. Virtual time. ACM Trans. Program. Lang. Syst. 7, 3 (July 1985), 404-425

[2] Image Credit: Wikimedia

Example Application: Mobiliti Transportation System Simulation Using Devastator





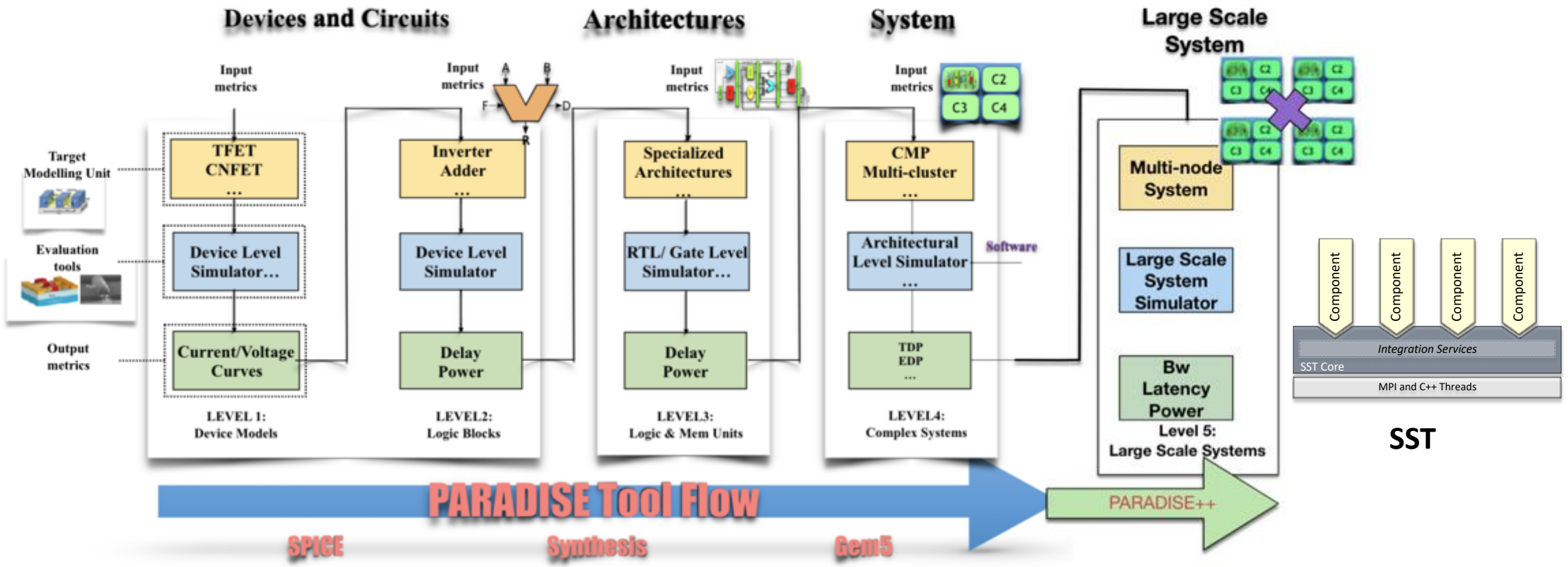
Demonstrated Parallel Scalability

- Enabled large-scale, distributed-memory traffic simulation of 1 million node, 2 million link road network with 20 million vehicles using 512 nodes of Cori
- Devastator simulator maintains deterministic causality across all actors

Contents

- Motivation
  - Need for fast simulation of large scale systems
- PARADISE Framework
  - End-to-End Device level to Architectural Simulation flow
- Devastator – PDES Framework
  - Optimistic Synchronization based Parallel Discrete Event Simulation
- **PARADISE++ for Large Scale Optimistic Simulations**
  - **SST simulation Framework**
  - **Recent Results**
- **Ongoing tasks**

PARADISE to PARADISE++



What is SST?

Goals

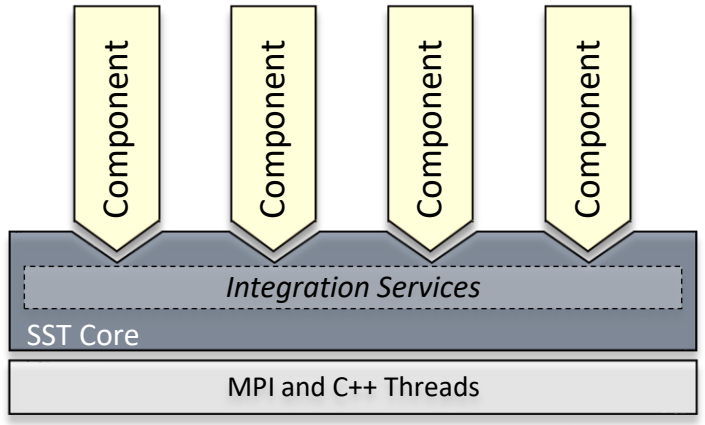
- Create a standard architectural *simulation framework* for HPC*
- Ability to evaluate future systems on DOE/DOD workloads
- *Use supercomputers to design supercomputers*

Technical Approach

- **Parallel** Discrete Event core
  - With conservative optimization over MPI/Threads
- **Interoperability**
  - Node and system-scale models
- **Multi-scale**
  - Detailed (~cycle) and simple models that interoperate
- **Open**
  - Open Core, non-viral, modular

Status

- Parallel framework (*SST Core*)
- Integrated libraries of components (*Elements*)
- Current Release (13.0.0)
  - https://sst-simulator.org
  - https://github/sstsimulator



Content Courtesy: SST Tutorials

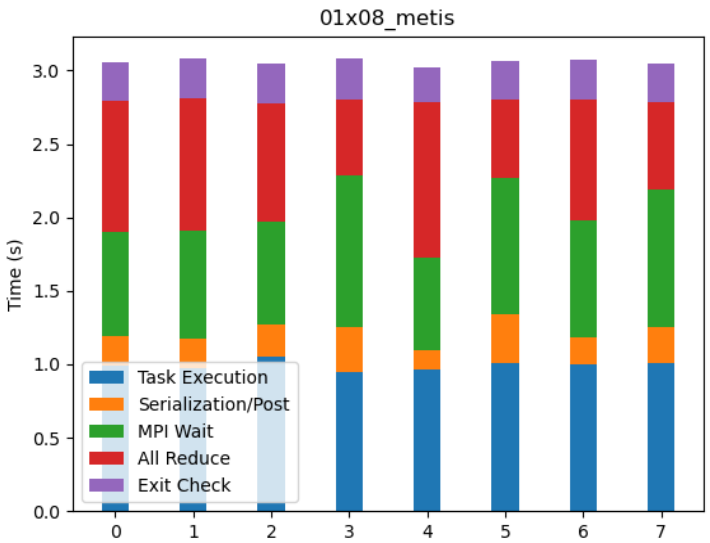
SST Components

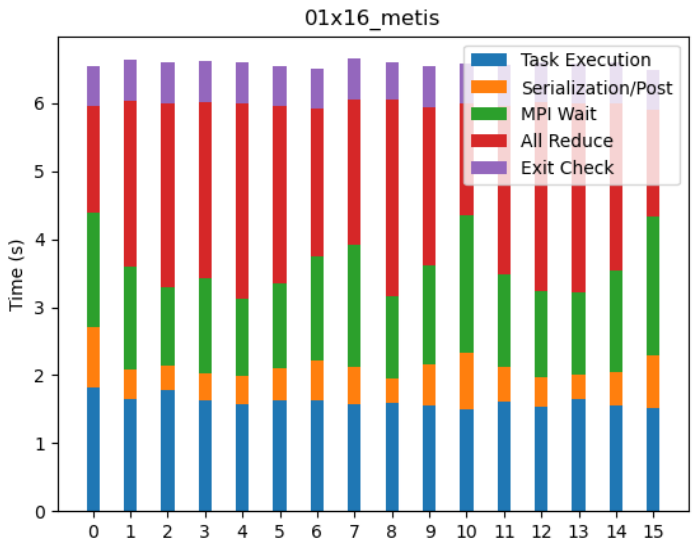
SST 9.0 Elements

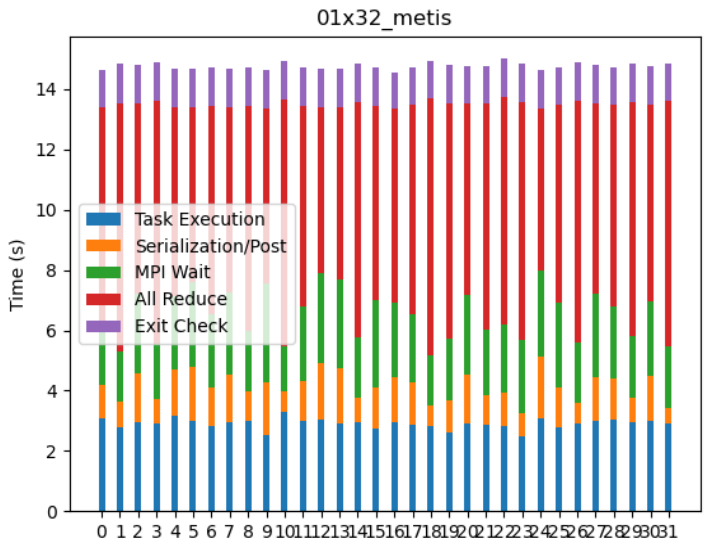
- **Processors**
  - Ariel – PIN-based
  - Juno – simple ISA processor
  - Miranda – pattern generator
  - Prospero – trace execution
  - GeNSA – Spiking temporal processing unit
- **Memory Subsystem**
  - cacheTracer – cache tracing
  - Cassini – cache prefetchers
  - CramSim – DDR, HBM
  - MemHierarchy – caches, directory, memory
  - Messier - NVM
  - Samba – TLB
  - VaultSimC – vaulted stacked memory
- **Network drivers**
  - Ember – communication patterns
  - Firefly – communication protocols
  - Hermes – MPI-like driver interface
  - Zodiac – trace based driver
  - Thornhill – memory models for Ember sims
- **Networks/NoCs**
  - Merlin – flexible network modeling
  - Kingsley – mesh NoC
  - Shogun – crossbar NoC
- **Others**
  - sst-macro – network drivers/network
  - scheduler – job scheduling
  - simpleSimulation – “car wash” example
  - simpleElementExample – many examples
  - sst-external-element – example element

Content Courtesy: SST Tutorials

SST Mesh NOC SpMV Performance (Breakdown and

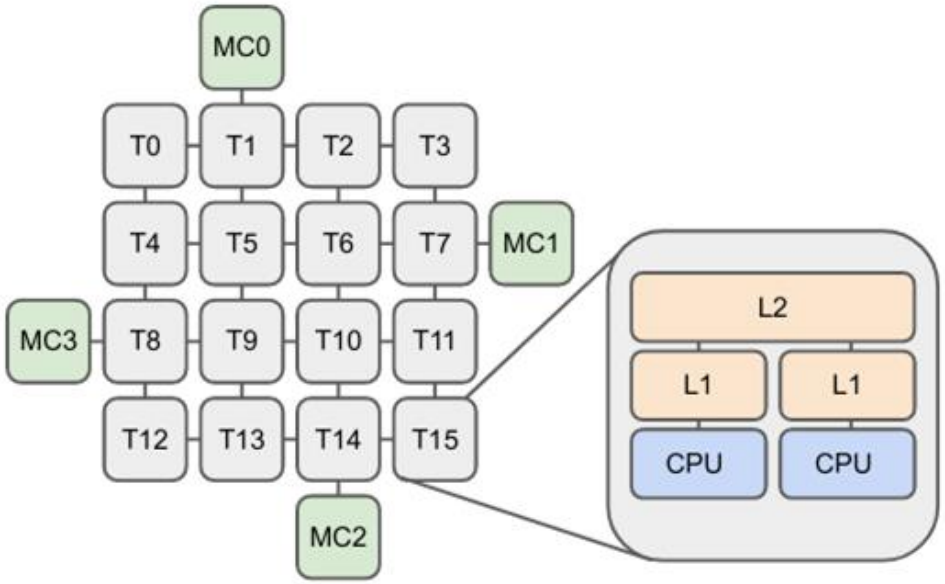




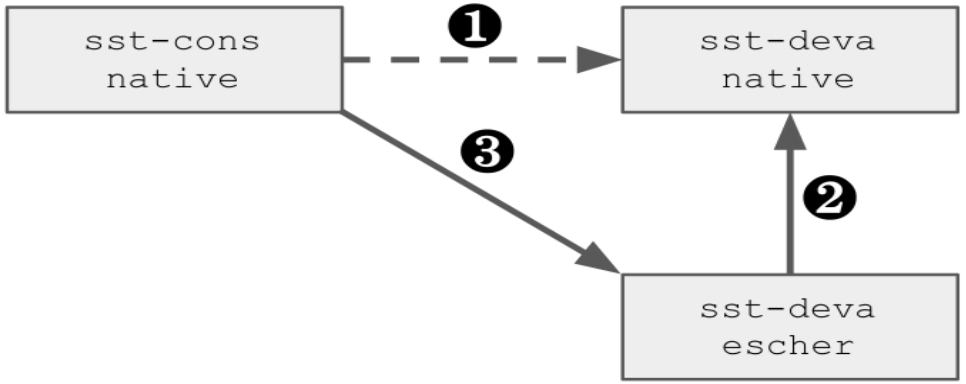


- Even with balanced load (METIS partitioner), large percent of **time spent in synchronization and communication**
- This configuration appears to be a good **potential test case for the Devastator back end** to improve simulation performance due to high amounts of time spent in the synchronization required for conservative simulation

Optimistic PDES Simulation: We simulate a tiled architecture

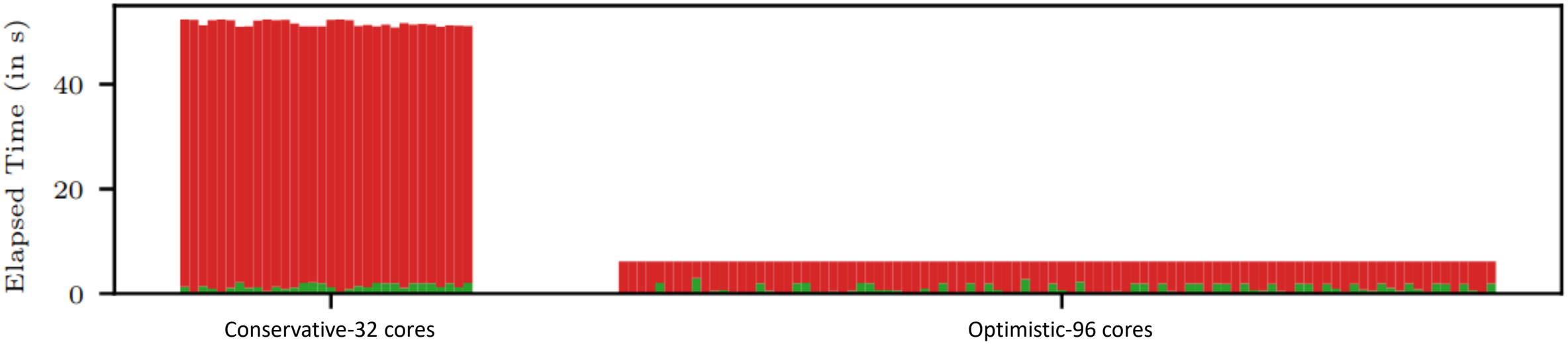


- Configured using components from sst-elements library
- Running stream benchmark
- Mesh network-on-chip has a lookahead of 100 ps
- Use a trace of SST to model optimistic synchronization performance



Optimistic simulation utilizes parallel resources more efficiently

Performance breakdown of best time-to-solutions



Colors indicate time spent in: ? elements and ? sst-core.

Faster computer architecture simulation via better parallelization

Key Takeaways

- Fast computer architecture simulation needed in post-Moore era
- Optimistic PDES reduces synchronization overhead

Next Steps

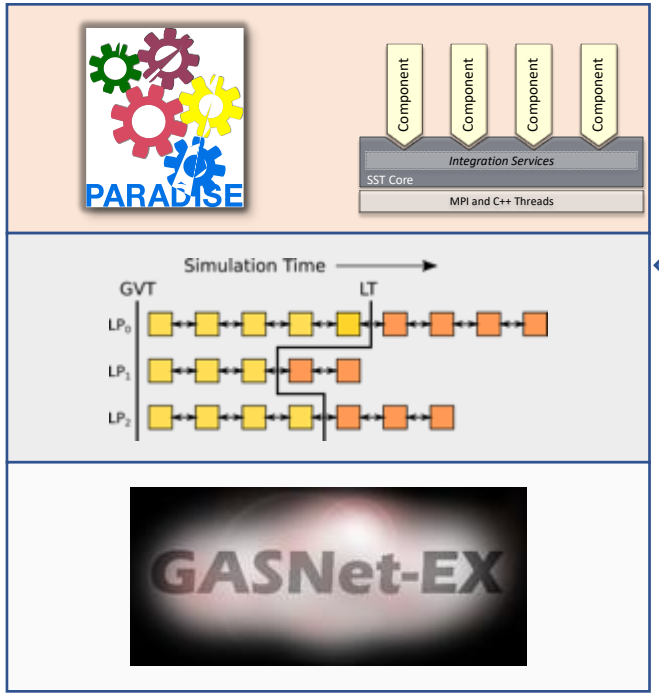
- Use Backstroke (source-to-source compiler tool) to automatically generate reversible SST-Elements

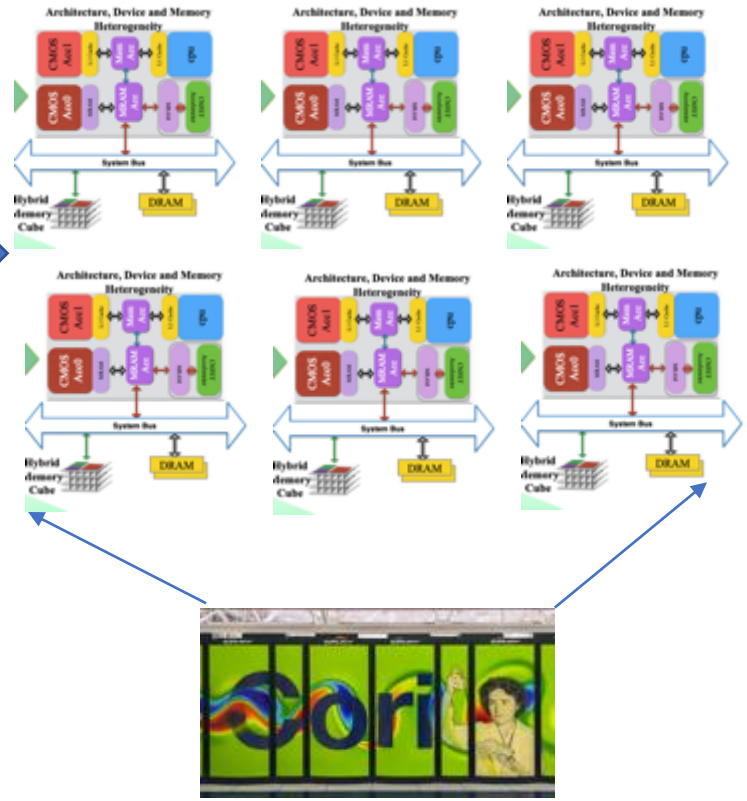
Things Not Discussed (Ask me questions!)

- Modifications to SST
- Discrepancy in event traces
- Devastator—our optimistic PDES engine

Summary

- PARADISE: Post-Moore device level to architectural level simulation support
- Devastator: Large Scale Simulation - Optimistic Synchronization based PDES
- SST extension for Post Moore Architectural support





- SST Current Release (13.0.0)
  - https://sst-simulator.org
  - https://github/sstsimulator

Thank You!

Contact us:
dilipv@lbl.gov
https://crd.lbl.gov/departments/computer-science/cag/