LA-UR 10-03242

Monte Carlo Transport on Heterogeneous Architectures

Tim Kelley

Applied Computer Science Group (CCS-7)

Los Alamos National Laboratory

SciDAC Computational Astrophysics Consortium

SLAC: May 19, 2010

Heterogeneous computing: using different types of processors to work together on a problem

- Why use heterogeneous computing architectures?
- What programming challenges do they bring?
- How does MC transport fit with heterogeneous computing?
- Our experience adapting Implicit Monte Carlo transport to *Roadrunner*

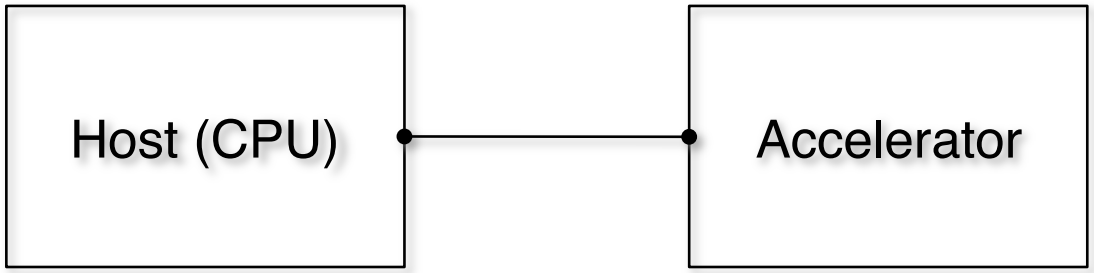
We are living in one of the most disruptive periods in computing history.

- Three “walls” simultaneously blocked progress in CPU performance
  - power, memory, and instruction width
  - remaining avenue: core count
- New design thinking is emerging from major manufacturers
  - Cell processor (Sony-Toshiba-IBM), Larrabee (Intel)
- Graphics Processing Units (GPU) are moving toward broader targets
  - floating point performance approaching TFlops (double precision)
  - vendors opening up:
    - providing specs, new API’s (OpenCL), hardware support for programming
- Extreme scale computing is driving a search for efficiency in hardware and better software models

Why use a heterogeneous architecture?

- Different workloads suit different processor types
  - many simple, in-order cores maximize parallel throughput (Cell SPE, Larrabee, Blue Gene, GPU)
  - fewer complex, out-of-order cores maximize sequential, single-thread performance (Opteron, Nehalem, Power7)
- Amdahl's law: parallelism is limited by sequential parts of code
- Heterogeneous strategy: get higher performance and efficiency by mixing core types
- Cost: code complexity
- Instances of heterogeneous computers:
  - Roadrunner: Cell processors accelerate ~50 TF cluster to 1 PF
  - ORNL projected: 10 PF-ish cluster accelerated by "Fermi" GPUs

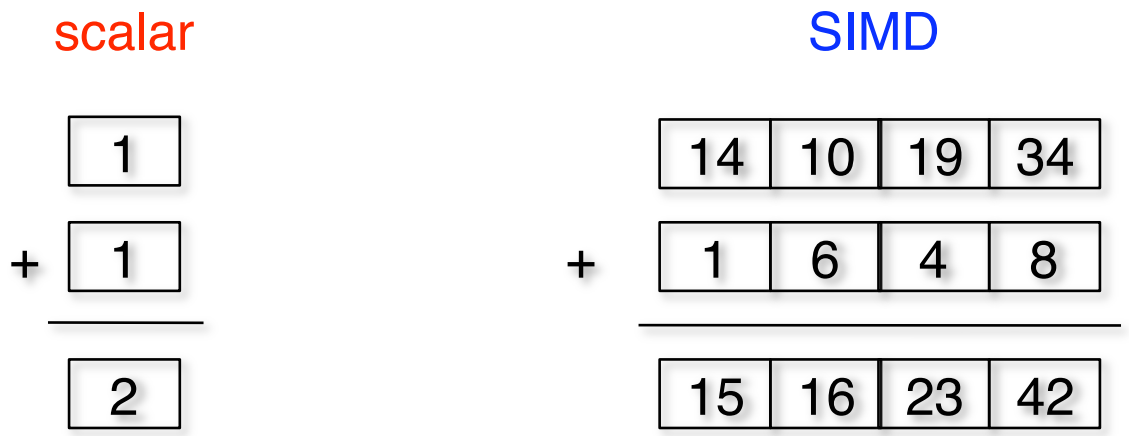
What do heterogeneous architectures look like?



- "Host" computer with an attached "accelerator"
- Accelerator examples: Cell processor, GPU, FPGA
- Moving toward single-chip architectures

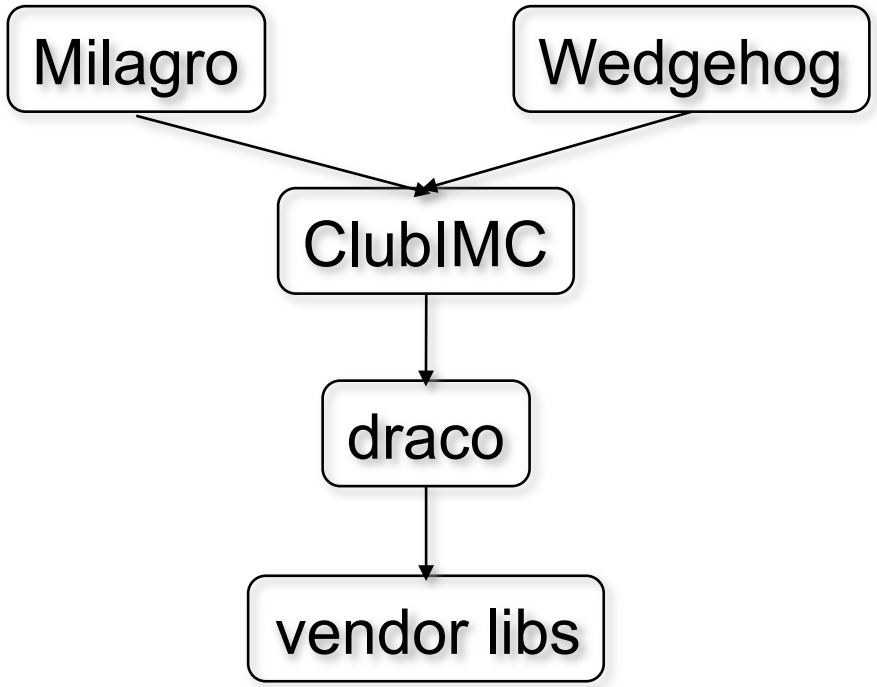
Accelerator characteristics introduce programming challenges

- large scale parallelism
- memory space disjoint from host
- complex memory hierarchies
- various hardware favoring different programming flavors
  - example: Single Instruction Multiple Data (SIMD, aka vector)



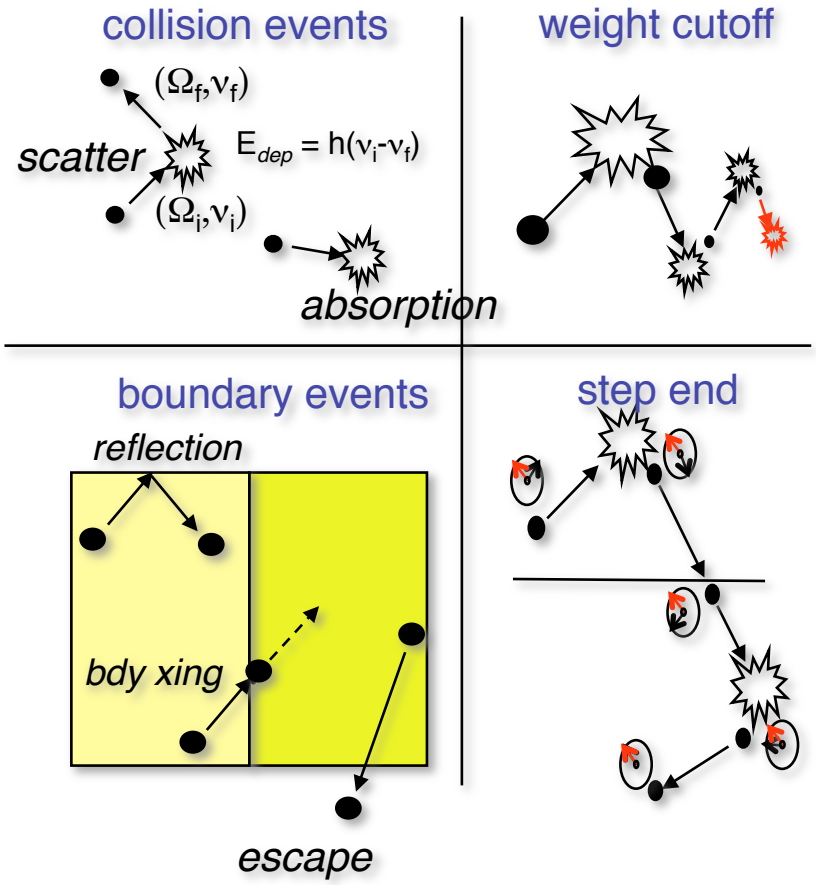
Milagro Implicit Monte Carlo code overview

- Fleck & Cummings time discretization
- object-oriented, generic C++:
  - templated on mesh type, freq type, particle type
- transports particles 3D, meshes articulated in 1,2,3D
- multigroup frequency treatment
- supports AMR
- two distributed parallel modes: mesh replicated, decomposed
- Wedgehog: Fortran callable interface library



Monte Carlo transport characteristics

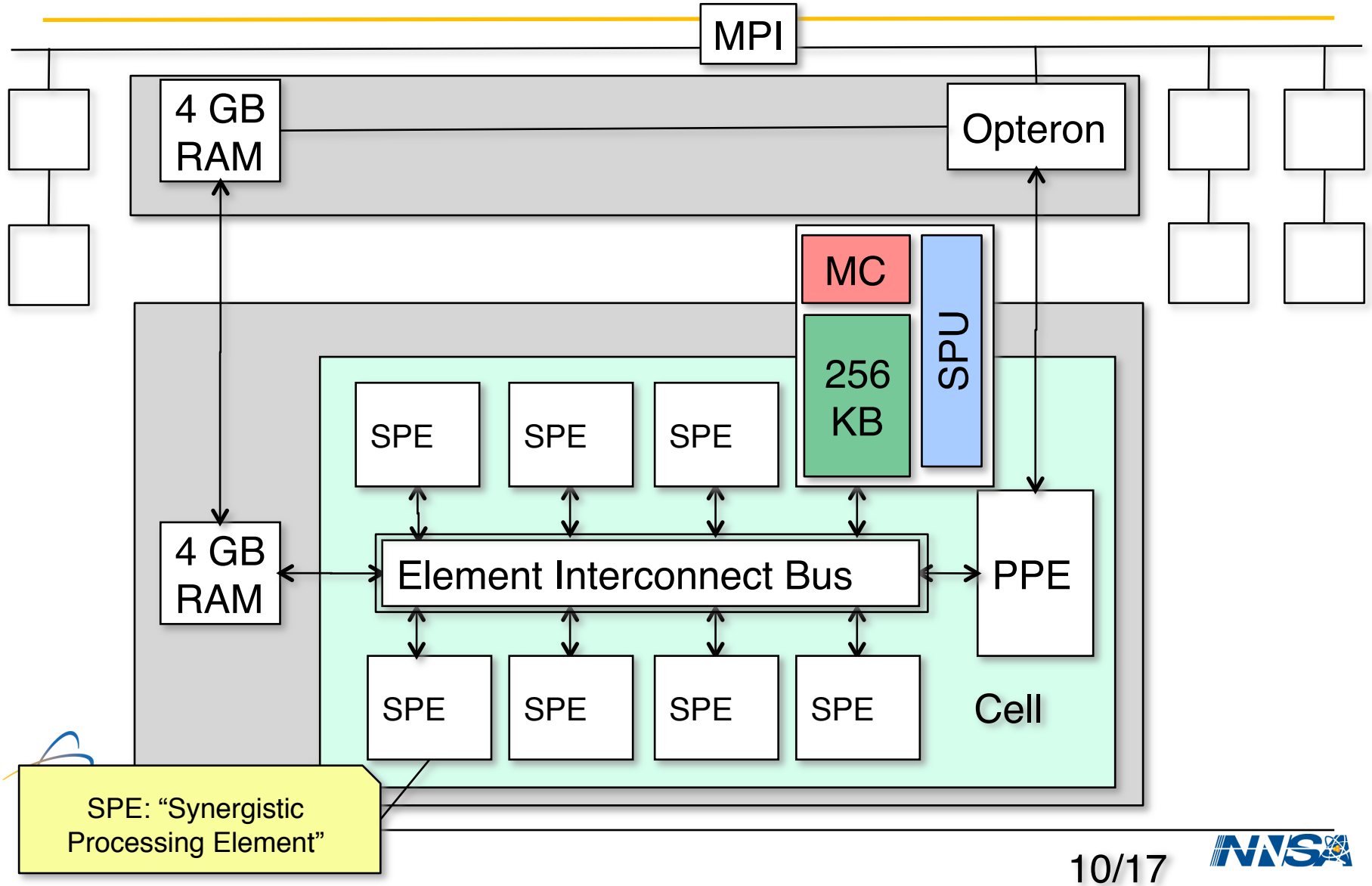
- MC transport tracks independent particles through mesh, tallies interactions with material in linearized, operator-split time step
- accelerator benefits:
  - independent particles: lots of thread-level data parallelism (large scale parallelism)
- accelerator challenges
  - random physical paths→random execution paths: hard to vectorize
  - randomly access mesh & material data sets: too large for fastest memories
  - branch-heavy code: weak or no branch prediction



Example of transport on heterogeneous architecture: Implicit Monte Carlo transport for *Roadrunner*

- Roadrunner supercomputer:
  - #1 on Top500 for 1.5 years
  - first to sustained petaflop
  - also very efficient— #3 on Green500
  - heterogeneous architecture
    - Opteron CPUs + FP-intensive Cell accelerators

An app programmer's view of Roadrunner hybrid node: one Opteron + one Cell



Roadrunner gives us a jump on advanced architectures

- hierarchical concurrency on many cores and threads
  - how to partition and control programs over hybrid resources?
- complex memory hierarchies
  - With RR, one *must* program the data motion (both weakness & strength)
- vectors are back
  - 128b now, 256-512b soon
  - much wider on GPUs (sort of—"SIMT" instead of SIMD)
- simple core architectures
  - little hardware tolerance for mediocre programming

Avoid branches by computing both legs, then masking

a simple example

```
for i in (0,1):
  if a[i] > b[i]:
    c[i] = a[i]
  else:
    c[i] = b[i]
```

3.14159265359	5.43656365918	a
6.28318530718	2.71828182846	b

`cmpgt(a,b)` produces `mask`

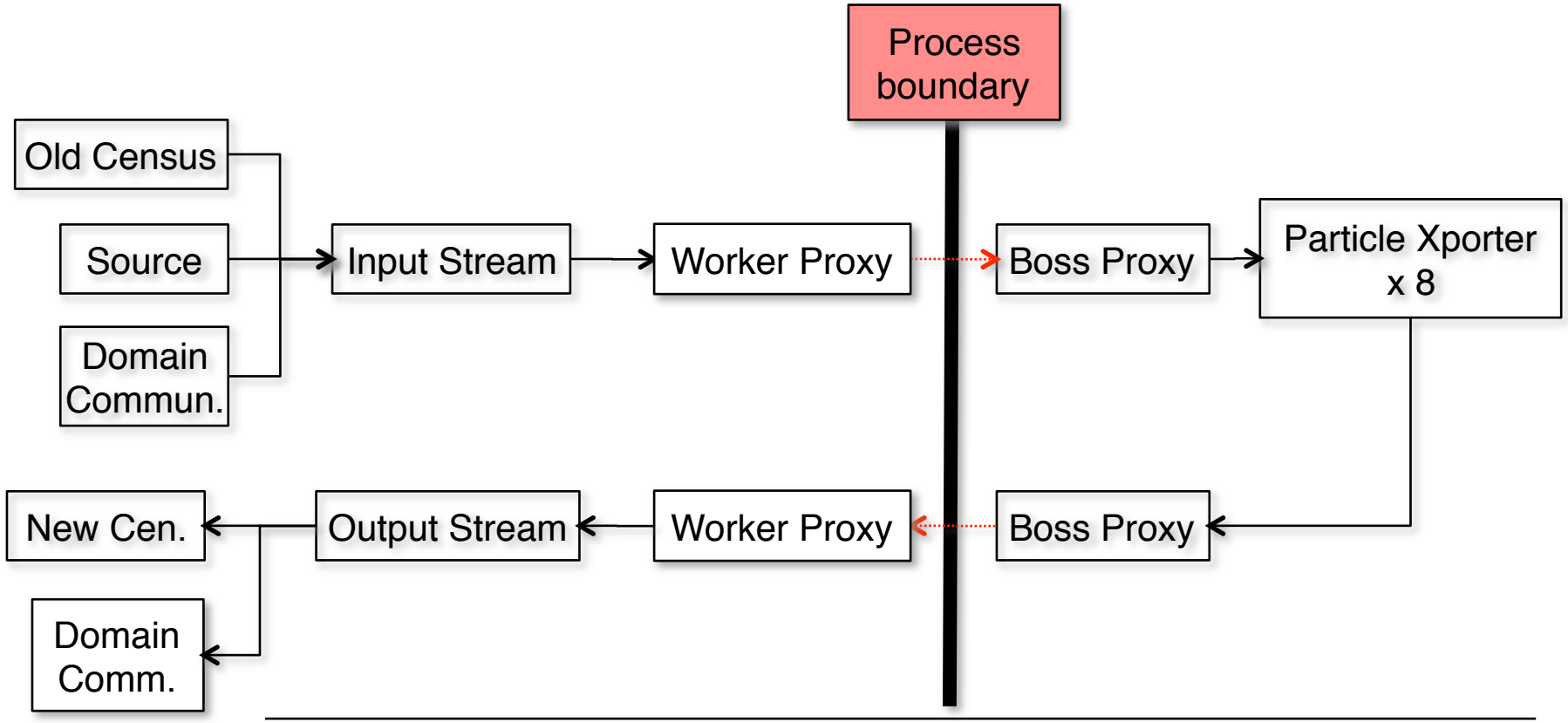
00000…0000000	1111111…1111111

`(a AND mask) OR (b AND ~mask)` produces `c:`

6.28318530718	5.43656365918

A few ideas enable heterogeneous decomposition

- Use streams, proxies to decouple particle generation from transport
  - these now happen concurrently

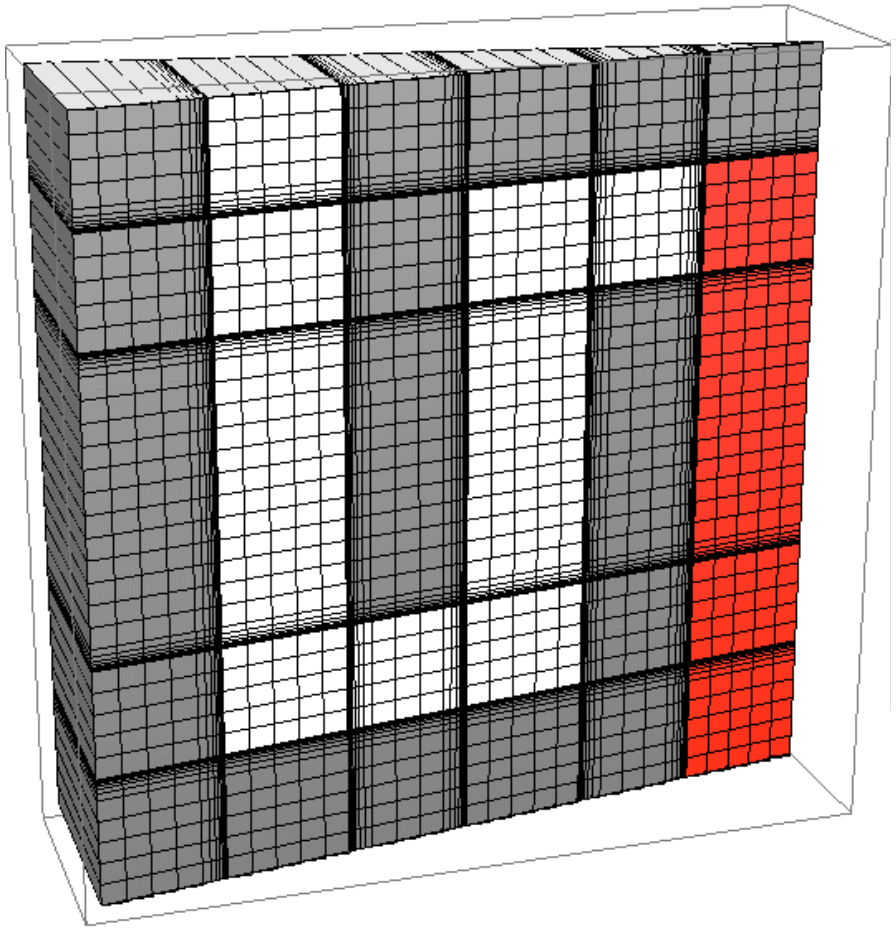


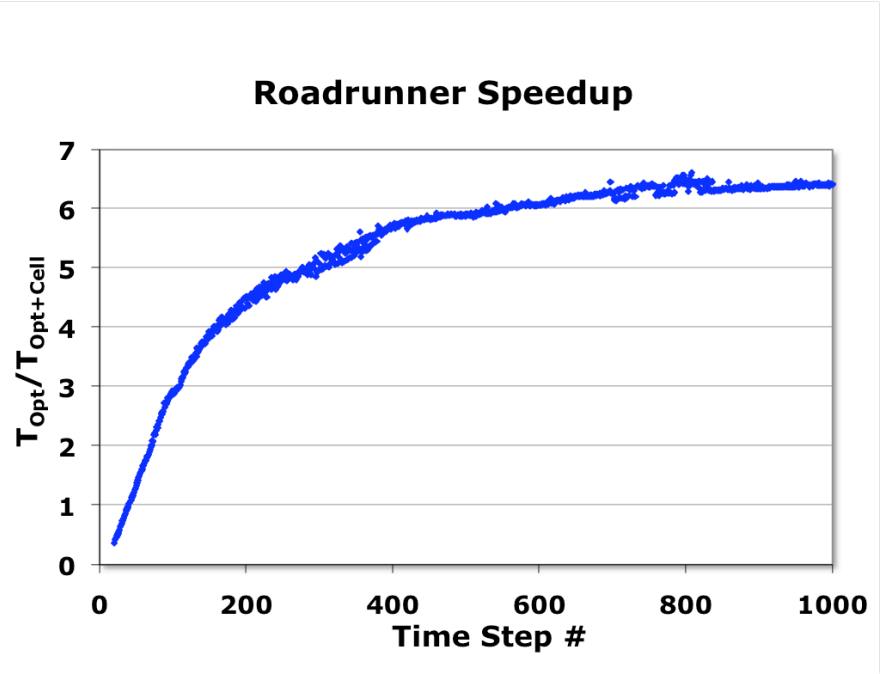
A hybrid time step:

who does what in each phase of a time step

	Opteron: host	PPE: manager	SPE: worker
initialize	▪ signal Cell to begin time step ▪ compute mesh, opacity ▪ send mesh, opacity to Cell	▪ receive mesh/opacity ▪ start SPE threads	▪ wait
transport	▪ generate particles, send to PPE ▪ recover spent particles from PPE, retire them (kill, census)	▪ synchronize particle I/O between host & workers	▪ load particles, mesh, opacity, tally data ▪ transport particles -- refresh mesh, tally opacity data ▪ store particles, tallies
finalize	▪ signal Cell ▪ wait for Tally finished signal ▪ recover Tally, update material state	▪ join SPE threads ▪ merge thread-private tallies ▪ signal host	▪ idle

Test problem “double bend”





Los Alamos NATIONAL LABORATORY EST.1943

NNSA

Future directions

- Revive Vector Monte Carlo transport for wider vector machines
- Continue to evaluate emerging hardware architectures
- Research programming languages, new programming paradigms
  - parallel Haskell
  - domain-specific languages

Applied Computer Science group

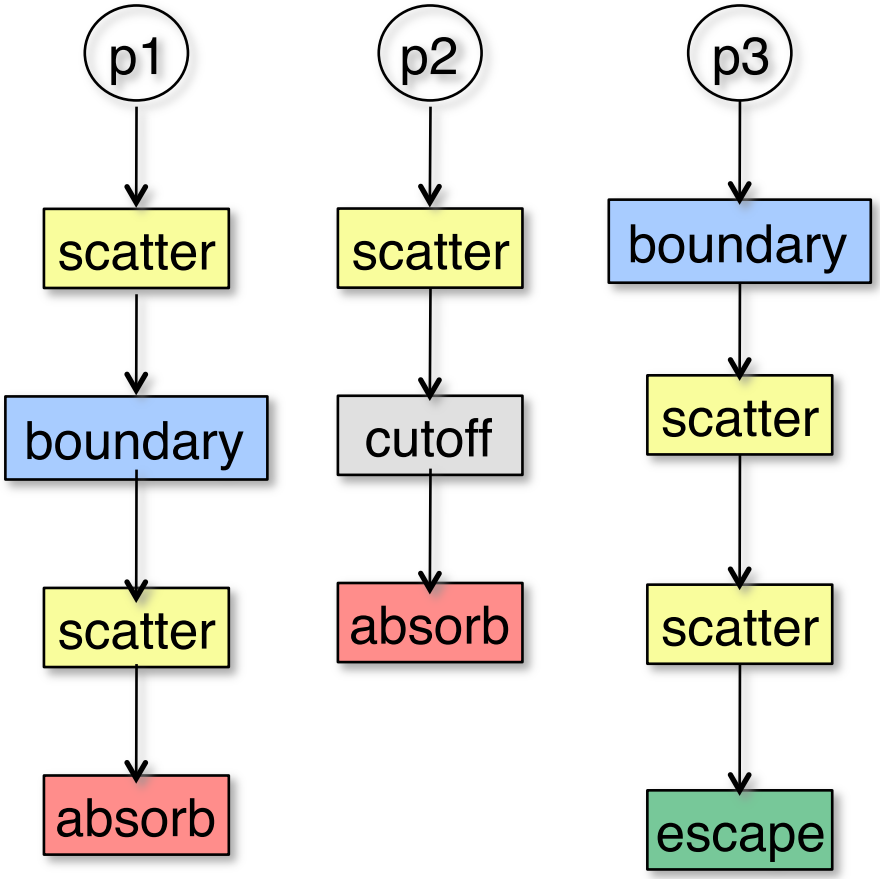
leading computational science onto novel computing architectures

- Four mutually-supporting teams:
  - algorithm+architecture co-design
    - jointly design machines and architectures
  - collaborative development
    - teach production code teams to design and code for new architectures
  - programming models and languages
    - develop tools and domain-specific languages to ease architecture migration
  - data science at scale
    - large scale data-mining and data-intensive problems
- Interdisciplinary group: computer scientists, physicists, engineers, applied mathematicians
- ***Major goal: train students & postdocs!***

Backup slides

MC particles follow different execution paths

this is difficult to vectorize



On the Cell SPE, we used scalar code written with vector intrinsics.