LA-UR-10-06873

Approved for public release;
distribution is unlimited.

Title:	Recent Advances and Future Prospects for Monte Carlo
Author(s):	Forrest B. Brown
Intended for:	Invited Plenary Session Talk Supercomputing in Nuclear Applications & Monte Carlo 2010 17-21 October 2010, Tokyo, Japan

Form 836 (7/06)

SNA + MC 2010
Tokyo, 17-21 Oct 2010

LA-UR-10-06873

Recent Advances & Future Prospects for Monte Carlo

Forrest Brown

Senior Scientist
Monte Carlo Codes, XCP-3
Los Alamos National Laboratory

Los Alamos NATIONAL LABORATORY EST.1943

NNSA Nuclear Criticality Safety Program

Recent Advances & Future Prospects for Monte Carlo

Forrest Brown (LANL)

The history of Monte Carlo methods is closely linked to that of computers: The first known Monte Carlo program was written in 1947 for the ENIAC; a pre-release of the first Fortran compiler was used for Monte Carlo in 1957; Monte Carlo codes were adapted to vector computers in the 1980s, clusters and parallel computers in the 1990s, and teraflop systems in the 2000s. Recent advances include hierarchical parallelism, combining threaded calculations on multicore processors with message-passing among different nodes. With the advances in computing, Monte Carlo codes have evolved with new capabilities and new ways of use. Production codes such as MCNP, MVP, MONK, TRIPOLI, and SCALE are now 20-30 years old (or more) and are very rich in advanced features. The former “method of last resort” has now become the first choice for many applications. Calculations are now routinely performed on office computers, not just on supercomputers. Current research and development efforts are investigating the use of Monte Carlo methods on FPGAs, GPUs, and many-core processors. Other far-reaching research is exploring ways to adapt Monte Carlo methods to future exaflop systems that may have 1M or more concurrent computational processes.

Monte Carlo Codes

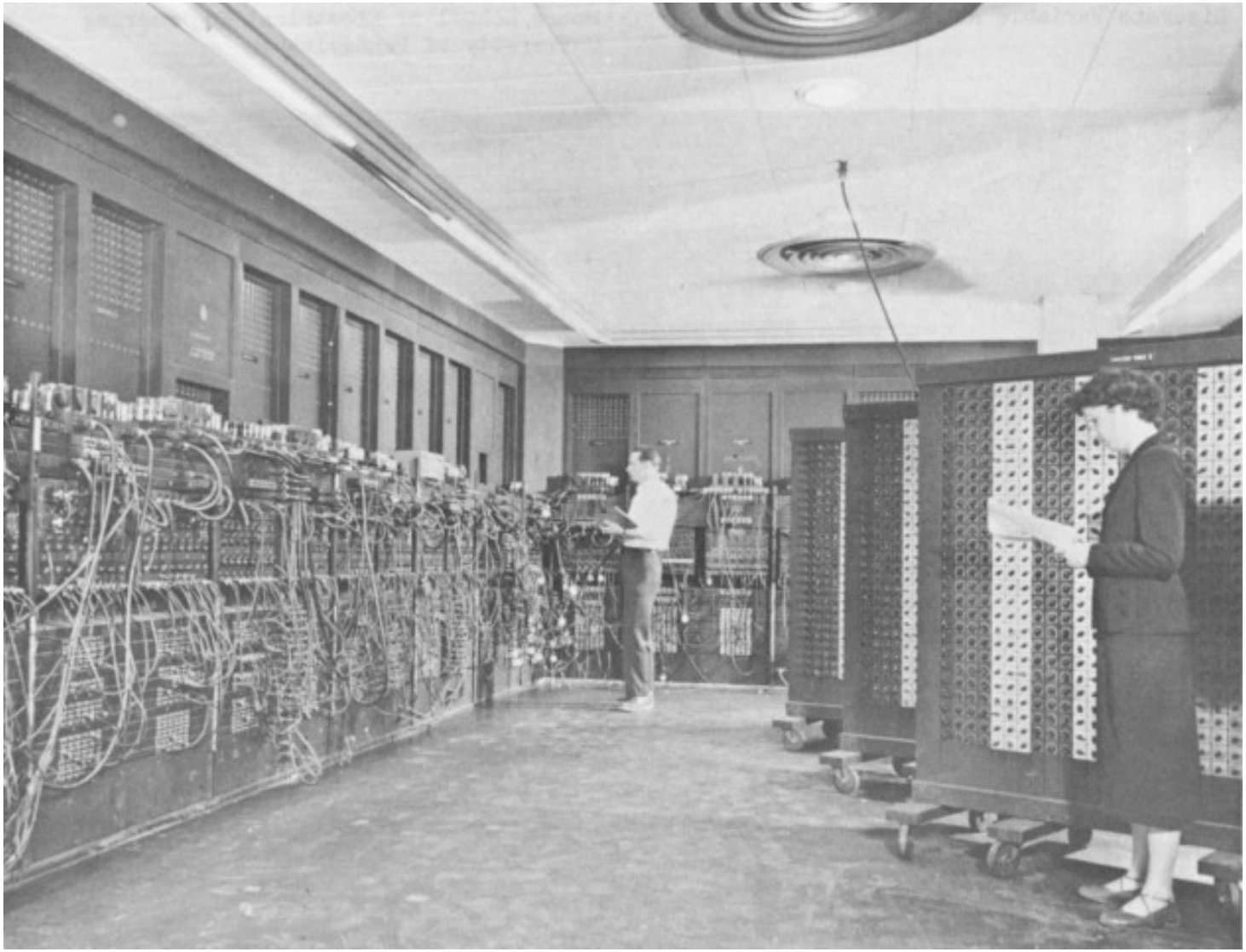
- The First 60 Years
- The Next Years
- Future Prospects

Monte Carlo
-
The First 60 Years

(I've only seen 30+ of those)

The First Computers

- **ENIAC - 1945, 30 tons, 18,000 vacuum tubes**

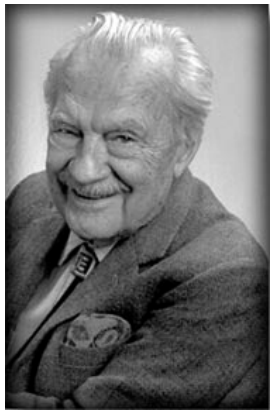


ENIAC Computer - 1945

- 20 ft x 40 ft room, 30 tons, 18,000 vacuum tubes
- Eckert solved the tube reliability problem through extremely careful circuit design. He was so thorough that before he chose the type of wire cabling … he first ran an experiment where he starved lab rats for a few days and then gave them samples of all the available types of cable to determine which they least liked to eat
- ENIAC could only hold 20 numbers at a time
- ENIAC's clock speed was 0.1 MhZ
- ENIAC's first task was to compute whether or not it was possible to build a hydrogen bomb. …. half a million punch cards for six weeks …

- **During the Manhattan Project in 1945**
  - Discussions on using ENIAC for calculations
  - **Stan Ulam** suggested using the ENIAC for the “method of statistical trials”
  - **Nick Metropolis** suggested the name “Monte Carlo”
  - **John Von Neumann** developed the first computer code for Monte Carlo

Stan Ulam



Nick Metropolis

John Von Neumann

Computers - 1940-50s

Monte Carlo Codes
XCP-3, LANL

- Small computers, small memory, punced cards, small codes
- Assembly language, FLOCO, ….

Calculation

		Instructions	Explanations
	1	r of $C_1 - 1$, *see (1)*	r_{i-1}
	2	r of C_1, *see (1)*	r_i
	3	$(C_3)^2$	s^2
	4	$(C_2)^2$	r^2
	5	$3 - 4$	$s^2 - r^2$
	6	$C_3 \begin{cases} \geqq 0 \therefore \mathscr{A} \\ < 0 \therefore \mathscr{B} \end{cases}$	$s \begin{cases} \geqq 0 \therefore \mathscr{A} \\ < 0 \therefore \mathscr{B} \end{cases}$
Only for $\mathscr{B}$	7	$(1)^2$	r_{i-1}^2
	8	$5 + 7$	$r_{i-1}^2 + s^2 - r^2$
	9	$8 \begin{cases} \leqq 0 \therefore \mathscr{B}' \\ > 0 \therefore \mathscr{B}'' \end{cases}$	$r_{i\ 1}^2 + s^2 - r^2 \begin{cases} \leqq 0 \therefore \mathscr{B}' \\ > 0 \therefore \mathscr{B}'' \end{cases}$
	10	$\begin{cases} \mathscr{A} \text{ or } \mathscr{B}' \therefore 2 \\ \mathscr{B}'' \quad \therefore 1 \end{cases}$	$\left.\begin{array}{l} \mathscr{A} \text{ or } \mathscr{B}' \therefore r_i = \\ \mathscr{B}'' \quad \therefore r_{i-1} = \end{array}\right\} r^*$
	11	$\begin{cases} \mathscr{A} \text{ or } \mathscr{B}' \therefore +1 \\ \mathscr{B}'' \quad \therefore -1 \end{cases}$	$\left.\begin{array}{l} \mathscr{A} \text{ or } \mathscr{B}' \therefore +1 = \\ \mathscr{B}'' \quad \therefore -1 = \end{array}\right\} \varepsilon$
	12	$(10)^2$	r^{*2}
	13	$5 + 12$	$r^{*2} + s^2 - r^2$
	14	11 sign) $\sqrt{13}$	s^*
	15	$14 - C_3$	d
	16	x of C_1, *see (1)*	x_i

FLOCO

	C	OPERATION				ADDRESS				REMARKS
		P	R	S		X	R	S		Special tally of flux as function of energy
	1	2	3	4	5	6	7	8	9	
0	I	8					8	6	5	
1	10		S	X	D	4		7	7	Save entrance
2	19		L	X	D	2	A	4	3	c(2)=j
3	28		T	S	X	4	9	1	5	Compute μ_n
4	37		S	S	P					
5	46		S	T	ϕ		1	0	0	$\lvert\mu_n\rvert$
6	55		C	L	A		A	1	6	W(E)
7	64		F	D	H		1	0	0	X 0
0	I		S	T	Q		1	0	1	$\psi(E)=W(E)/\lvert\mu_n\rvert$
1	10		L	X	A	1	4	0	1	c(1)=n=1
2	19		C	L	A		A	1	7	E
3	28		C	A	S	1	L	0	0	Compare E with E_n
4	37	1			1	1	X	1	3	$E>E_n$: (n+1)→n

Monte Carlo - 1940-50s

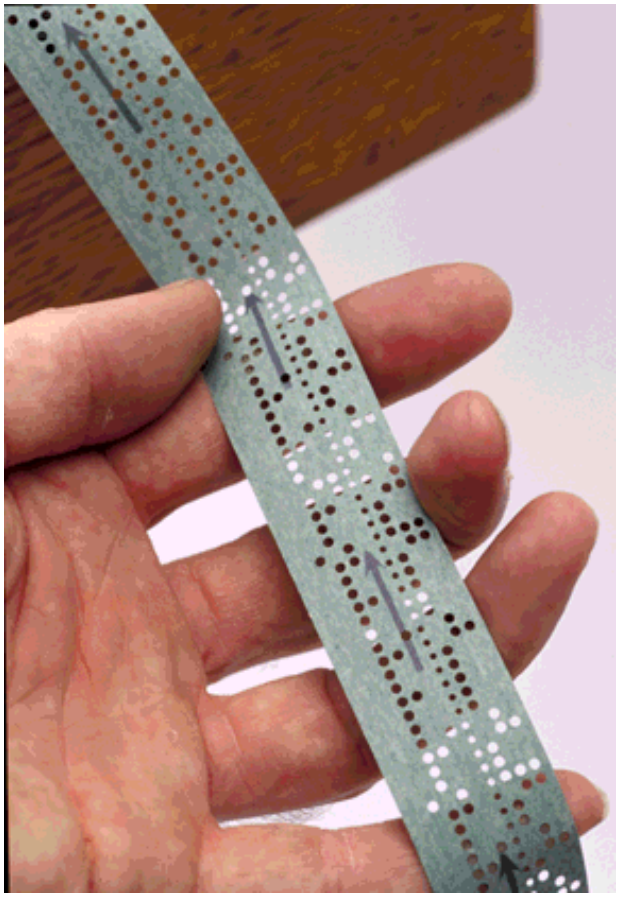
John Von Neumann (my hero)

- **Father of scientific computing**
  - Traditional computers called “Von Neumann” machines
  - Stored programs, not just data
  - Flowcharts, logic, …..
  - **Basic logic for Monte Carlo particle transport described in 1947, first MC computer code** (see LAMS-551, 1947)

100 neutrons x 100 collisions = 5 hr on ENIAC

- **Classic paper on rejection sampling methods for sin, cos, exp**
  - Still used in MCNP & other MC codes
- **Many small, special purpose MC codes**
  - MCS, MCN, MCP, …..

Computers - 1950-60s



- **1957 - first Fortran**
  - Pre-release of the first Fortran compiler from IBM arrived at Bettis
  - 2,000 punched cards for a binary image of the compiler
  - First program tested at Bettis failed due to a program syntax error
    - Missing comma, user input syntax error
    - First case of user frustration and debugging with Fortran
  - Then, successfully compiled and executed on an IBM 704
  - Lew Ondis began using the new compiler for Monte Carlo codes
    - Lew was one of the principle developers of RCP, an amazingly capable MC code for reactor analysis developed in the 1960s
    - Others include Norm Candelore, Bob Gast, Ely Gelbard
- **Fortran has been used for MC particle transport codes since 1957 !**

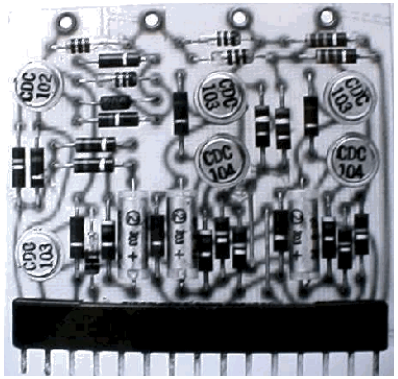
"CDC: the dawning era of supercomputers" - Gordon Bell

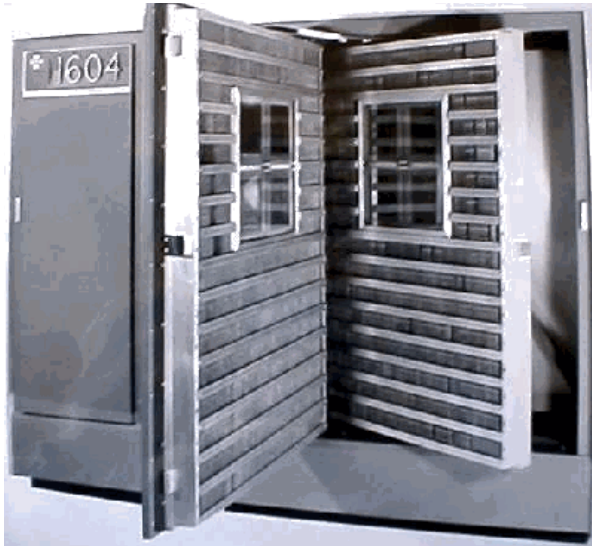
"The times, they are a changin …" - Bob Dylan, 1964

"The single person most responsible for supercomputers"
- Dave Patterson (coined the term RISC)

- **Seymour Cray**
  - Father of supercomputing
  - CDC-1604, CDC-6600, CDC-7600, Cray-1
  - Led to Cray XMP, YMP, C90, J90, T90, …
  - When asked what kind of CAD tools he used for the Cray-1, he said that he liked #3 pencils with quadrille pads …
  - When told that Apple bought a Cray to help design the next Mac, Seymour commented that he just bought a Mac to design the next Cray…

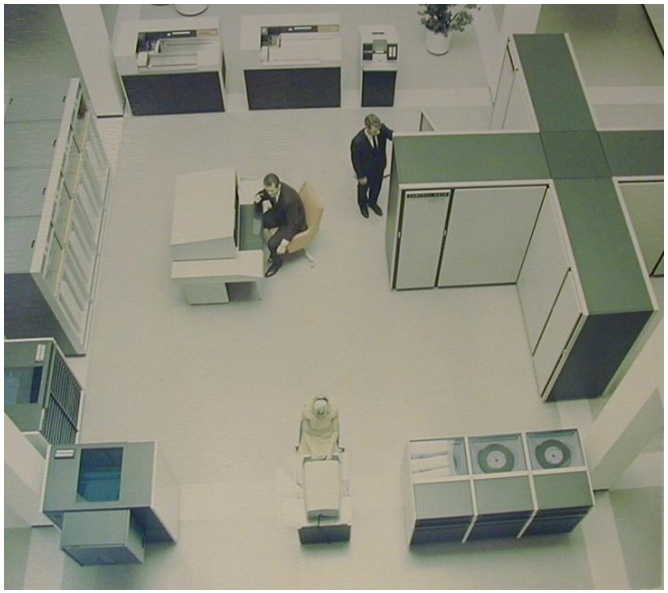
- **CDC 1604 - 1960 - Seymour Cray**
  - First CDC computer for technical computing
  - 32-bit words, 2.2 μsec memory access, 1.2 μsec op cycle
  - Functional parallelism, multithreading

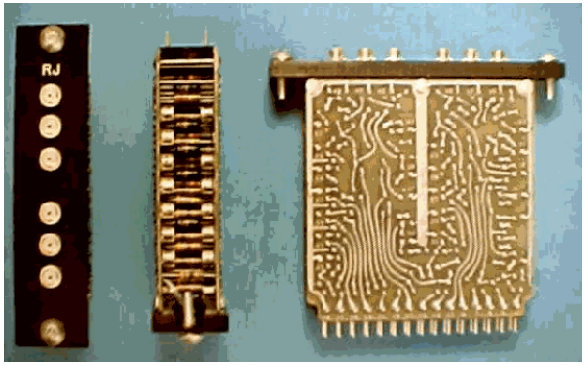


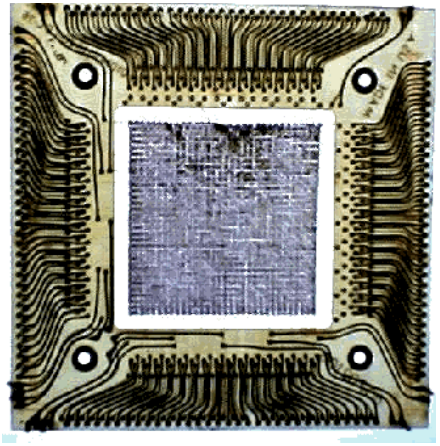


Supercomputers - 1960s …..

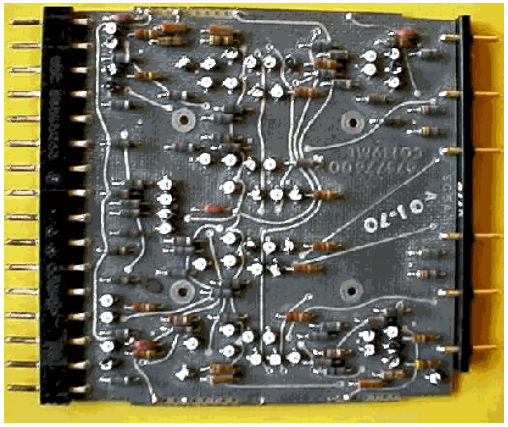
- **CDC 6600 - 1964 - Seymour Cray**
  - Only 50 produced
  - 60-bit words, 128 K word memory
  - 100 ns clock, 1000 ns op cycle
  - Functional parallelism, multithreading

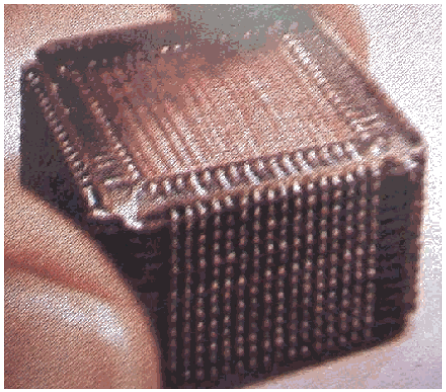






- **CDC 7600 - 1969 - Seymour Cray**
  - 60-bit words, 65 K word small core, 512 K word large core
  - 27.5 ns clock, 36 MHz
  - Pipelining, functional parallelism



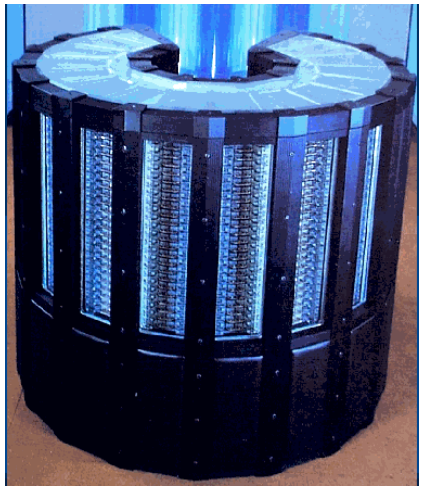


Supercomputers - 1970s …..

- **Cray-1 - 1976 - Seymour Cray, Cray Research**
  - 12 ns clock, 64-bit words, 1 M word memory
  - 8 vector registers, 64-words each
  - Pipelining, functional parallelism, vector unit “chaining”
  - Peak vector speed - 80 Mflops
  - Originally came with no OS & no compiler….

Cray-1

Cray-XMP (1985)



Cray-2

1996… - Tera flops

- **ASCI**
  - Accelerated Strategic Computing Initiative - LANL, LLNL, SNL
  - US Dept. of Energy, from mid 1990s, for Stockpile Stewardship
  - **First Tera-flop computer - 1996**
    - ASCI Red, Sandia National Laboratory
    - $55 M, Intel, 1600 sq. ft,
    - 9,072 Pentium Pro processors
    - Followed by ASCI Blue Mountain (LANL), ASCI Blue Pacific (LLNL)

1998 - ASCI Blue Mountain Tflop system at LANL

2008… - Peta flops

- **ASC**
  - Advanced Simulation & Computing
    - LANL, LLNL, SNL
  - US Dept. of Energy, NNSA, from mid-2000s, for Stockpile Stewardship

ASC™

  - **First Peta-flop computer - 2008 - Roadrunner at LANL**
    - 6,912 dual-core AMD Opteron, 50 Tflops
    - 12,960 Cell processors (attached to Opterons), 1.3 Pflops
    - 107 TB aggragate memory
    - 3.9 MW power requirement, 5500 sq ft

Monte Carlo - 1960-80s

- **Explosion of production codes, very capable & respected developers**
  - **O5R, MORSE, KENO (ORNL)**
  - **RCP, RECAP, PACER (Bettis)**
  - **STEMB, RACER (KAPL)**
  - **MCS, MCN, MCP, MCNP (LANL)**
  - **COG, TART (LLNL)**
  - **SAM (MAGI)**
  - **TRIPOLI (CEA)**
  - **MONK, MCBEND (UK)**
  - **VIM (ANL)**
  - **MVP (JAERI)**

- **Computers**
  - IBM, Univac, Philco, CDC-6600, CDC-7600, Cray-1, Vax, …..

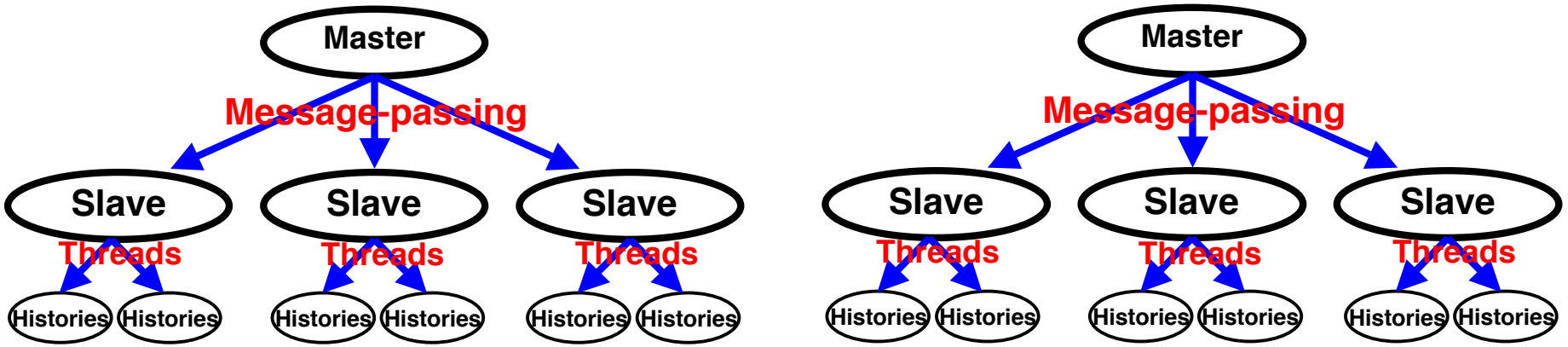
Computing - 1980-90s - Parallelism

- **Vector - 1980s**
  - Cray-1 — Special-purpose MC codes for photon transport, LANL
  - Cyber-205 — RACER MC at KAPL, reactor analysis
- **Parallel/Vector - 1980s**
  - Cray-XMP - 2 vector processors
  - Cray-YMP - 4 vector processors
    - Vector + parallel MC - RACER at KAPL, for reactor analysis
  - Cray-C90 - 16 vector cpus
    - Vector + parallel + threads - RACER at KAPL for reactor analysis
    - Reactor assembly depletion, big-time
  - Do-it-yourself message passing - before PVM & MPI
- **Parallel - 1990s**
  - PVM, MPI
  - “massive parallelism” - RACER, RCP, MCNP (but, nothing useful)
  - Clusters of workstations - useful, precursor to today’s Linux clusters
    - RACER, RCP, PACER, VIM, MVP, MCNP

Computing - 1990s-2000s - Parallelism

- **Message-passing Parallelism**
  - MPI & PVM - 1990s (Today - OpenMPI is "the" standard)
  - Heavyweight parallel - separate tasks that communicate
- **Threading**
  - Initially vendor-specific, compiler directives for some machines, …
  - Today - OpenMP standard
  - Typically effective only for 2-16 processors
- **In the 1990s, only a few dozen processors could effectively be used for realistic MC calculations**
- **MC was successfully demonstrated on loosely-coupled clusters of workstations (a few dozen) - VIM, MCNP**
- **In the 2000s, large clusters & Tflop systems appeared, & calculations with 1000s of processors became routine**
- **Hierarchical Parallelism was established & used in some MC codes**

Concurrent Jobs →



Parallel Processes

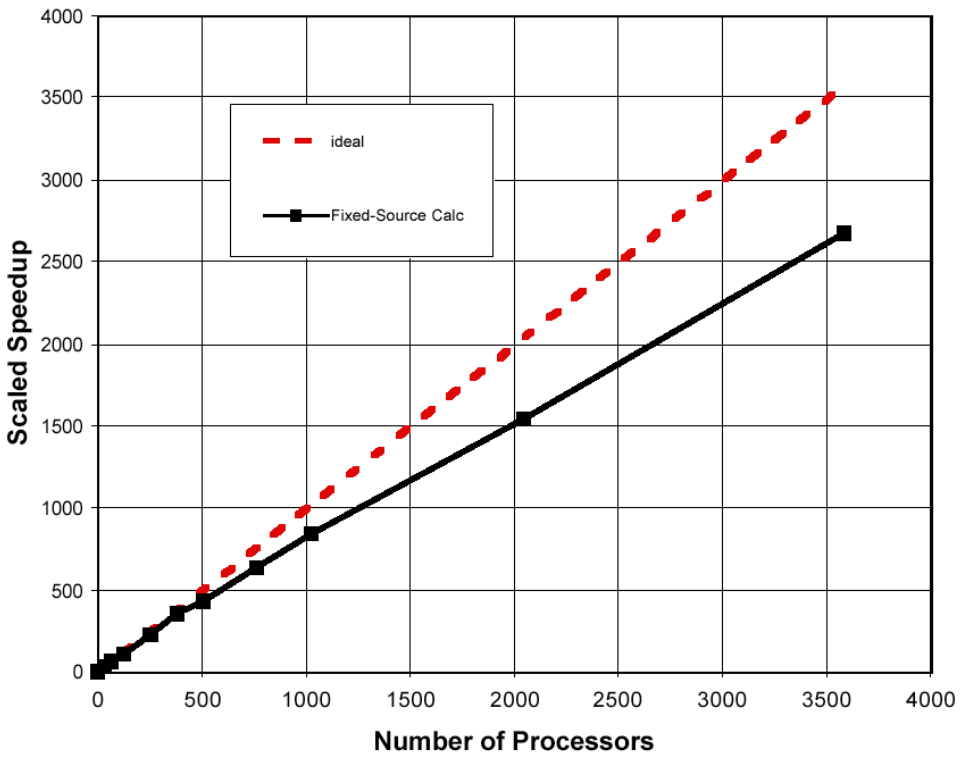
- **Total processes = (# jobs) x (# MPI processes) x (# threads)**

- **Tradeoffs:**
  - **More MPI processes - lots more memory**
  - **More threads - contention from lock/unlock shared memory**
  - **More jobs - system complexity, combine results**

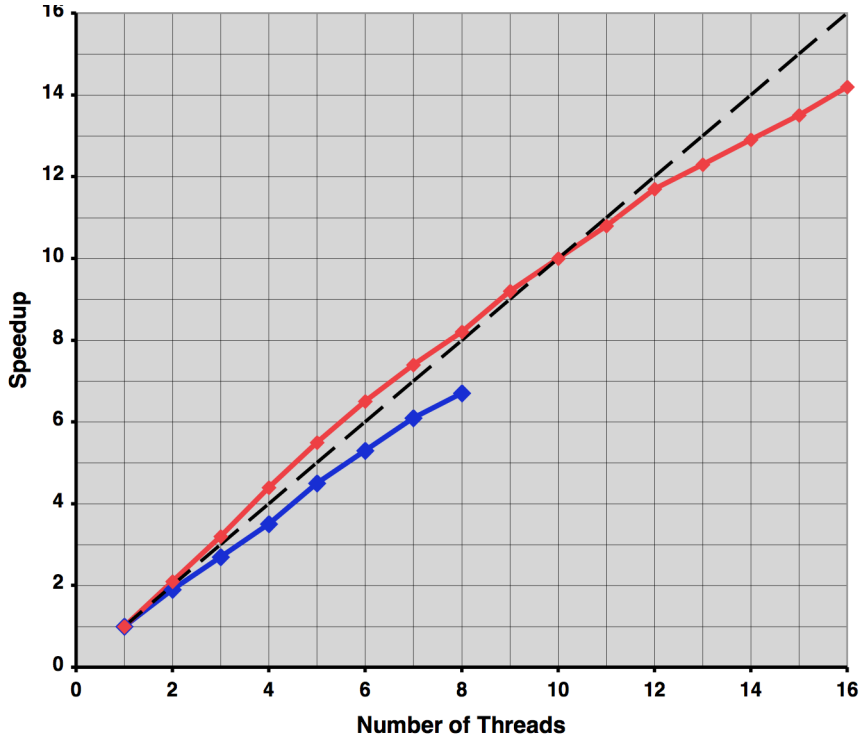
MCNP - Hierarchical Parallelism

Monte Carlo Codes
XCP-3, LANL

- Use MPI to distribute work among slaves ("boxes")
- Use threading to distribute histories among individual cpus on box



ASCI Q, MPI+OpenMP,
4 threads/MPI-task



Lobo - 4 x Quad-core, 16 threads/node

Mac Pro - 2 x Quad-core, 8 threads/node

Monte Carlo
-
The Next Years

- **Computers continue to evolve - speed & accessibility**
  - **Everyone now has multicore, Gflop computers - laptops, deskside**
  - **Almost everyone now has access to Linux clusters**
  - **New computers may have 16, 32, 48, 64, 80, … cores per processor**
- **MC codes must evolve**
  - **All MC codes - new & old - must be parallel, with threading + MPI**
  - **Increased & varied usage produces new demands**
    - Many 1000s or millions of regions & tallies
    - 1000s of materials with 100s of isotopes each
  - **Rewriting / restructuring old MC codes is expensive**
    - 10 people for 2 years to rewrite MCNP & produce MCNP5
  - **Often cheaper / quicker to write new MC codes**
    - SERPENT, MC21, McCARD, MCP, Monaco, MVP-2, …..
  - **GPGPUs - fastest hardware**
    - Much like vector processing of 1980s
    - Rewriting old MC codes is prohibitively expensive
    - Can be effective for small, special-purpose, new MC codes

A Digression .....

- **Many people still argue passionately over what computer language to use - Fortran, C, C++, Java, ...**
  - At best, these arguments provide entertainment over beers
  - At worst, they obscure the real issues & waste time
- **Experience has proven that the choice of language has no impact on the speed of MC codes, nor on development time or QA**
  - **Beginners can write bad code in any language**
  - **Experts can write good code in any language**
  - **Experiments at LANL**
    - MC research code for criticality
      - Written in Fortran, rewritten in C++
      - **Same speed & capabilities**
    - Million-line Stockpile Stewardship code system
      - 50 person effort, rewritten in C++
      - **Same speed & capabilities**
- **Pick a language that best matches people's skills & experience**
- **OK to mix languages - Fortran for some things, C++ for others**

Perspective - Reactor Analysis with MC

- **As computing power has increased, the use of Monte Carlo methods for reactor analysis has grown**
- **Also, since more histories give better localized statistics, the principal uses of Monte Carlo in reactor analysis have evolved:**

1960s:	K-effective
1970s:	K-effective, detailed assembly power
1980s:	K-effective, detailed 2D whole-core
1990s:	K-effective, detailed 3D whole-core
2000s:	K-effective, detailed 3D whole-core, depletion, reactor design parameters
2010s:	All of above + Total Uncertainty Quantification (impact of uncertainties in cross-sections, manufacturing tolerances, methodology, …)

Top Ten R&D Efforts

- **With faster, more capable computers, there are many new R&D efforts in progress to develop new MC analysis capabilities**
  - Some of the R&D efforts were not possible until now, due to computer limitations on speed &/or memory
  - Other R&D efforts are addressing new methods to eliminate approximations made in MC codes 20-30 years ago, that are now significant due to much small uncertainties
- **Of course, everyone's R&D project is of the highest importance to the future of mankind…..**
- **I've listed my 10 personal favorites on the next few slides.**
  These are not ranked in any particular order.

1. New adjoint-weighted tally schemes for continuous-energy Monte Carlo criticality calculations (Kiedrowski, LANL)

- Permits correct calculation of reactor kinetics parameters
- Perturbations in reaction rates
- Sensitivity-uncertainty parameters

B.C. Kiedrowski, F.B. Brown, & P. Wilson, "Calculating Kinetics Parameters and Reactivity Changes with Continuous-Energy Monte Carlo", ANS PHYSOR-2010, Pittsburgh, PA, May 9-13 (2009).

B.C. Kiedrowski, F.B. Brown, "Comparison of the Monte Carlo Adjoint-Weighted and Differential Operator Perturbation Methods", SNA+MC 2010

2. Sensitivity-uncertainty analysis of cross-section data (Rearden, ORNL)

- Forward/adjoint multigroup KENO, weighting with covariance data
- Can quantify significant uncertainty in results due to cross-sections

See SCALE6 documentation for TSUNAMI & TSURFER codes

3. New iteration methods for criticality calculations (Brown, LANL; and others)

- Wielandt's method
- May accelerate convergence
- May eliminate the underprediction bias in uncertainties

F.B. Brown, "Wielandt Acceleration for MCNP5 Monte Carlo Eigenvalue Calculations", M&C+SNA-2007, Monterey, CA, April 15-19, 2007 (April 2007).

4. On-the-fly Doppler broadening of neutron cross-sections (Yesilyurt, U Mich / ORNL)

- Permit a continuous distribution of material temperatures
- Essential for multiphysics calculations, with neutronics/CFD coupling

G. Yesilyurt, W.R. Martin, F.B. Brown, "On-The-Fly Doppler Broadening for Monte Carlo Codes", ANS M&C-2009, Saratoga Springs, NY, May 3-7 (2009).

5. New types of continuous estimators for tallies (Banerjee, U Mich; Griesheimer, Bettis)

- Kernel Density Estimators (KDE)
- Functional Expansion Tallies (FET)
- Permit continuous variation over regions, rather than just simple averages.

K. Banerjee & W.R. Martin, "Kernel Density Estimation Method for Monte Carlo Tallies with Unbounded Variance", *Trans. Am. Nucl. Soc.,* Vol. **101** (2009).

D.P. Griesheimer, "Functional Expansion Tallies for Monte Carlo Simulations", Ph.D dissertation, University of Michigan (2005).

6. Depletion analysis of fuel assemblies and reactors (Leppanen, VTT; KAPL/Bettis; many others)

- including equilibrium Xenon and control searches

Leppanen, J., 2009. PSG2/Serpent – A Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code, User's Manual (February 2, 2009). VTT Technical Research Centre of Finland.

T.M. Sutton, et al., "The MC21 Monte Carlo Transport Code", M&C+SNA-2007, Monterey, CA, April 15-19 (2007).

7. All-particle, all-energy Monte Carlo codes (MCNP6, LANL)

- MCNP6 = MCNP5 + { hi-energy modules from MCNPX }
- In progress…

"Reactor Physics Calculations with MCNP5 and the Status of MCNP6", workshop for PHYSOR-2010, Pittsburgh, PA, May (2010), available at URL:
mcnp.lanl.gov/publication/pdf/la-ur-10-02762_physor2010_workshop.pdf

8. Improved treatment of the neutron free-gas scattering model at epithermal energies (Becker, Dagan)

- Include important resonance scattering effects
- Fixes long-standing approximation

B. Becker, R. Dagan, G. Lohnert, "Proof and Implementation of the Stochastic Formula for Ideal Gas, Energy Dependent Scattering Kernel", *Annals of Nuclear Energy* 36, 470-474 (2009).

9. Coupled calculations involving both Monte Carlo and deterministic transport codes (Wagner, Peplow, ORNL; many others)

- Use deterministic code for fast multigroup adjoint calculation
- Use multigroup adjoint for MC importance sampling
- Enables more effective variance reduction

D. E. Peplow and J. C. Wagner, "Automated Variance Reduction for SCALE Shielding Calculations," in *Proc. of ANS 14th Biennial Topical Meeting of the Radiation Protection and Shielding Division*, pp. 556-558, Carlsbad, New Mexico, April 2-6, 2006.

C.J. Solomon, A. Sood, T.E. Booth, and J.K. Shultis, "An Sn Approach to Predicting Monte Carlo Cost with Weight Dependent Variance Reduction", Trans. Am. Nuc. Soc, 103, Nov 2010, [also, LA-UR-10-04536] (2010).

10. Stochastic geometry (Brown, LANL; many others)

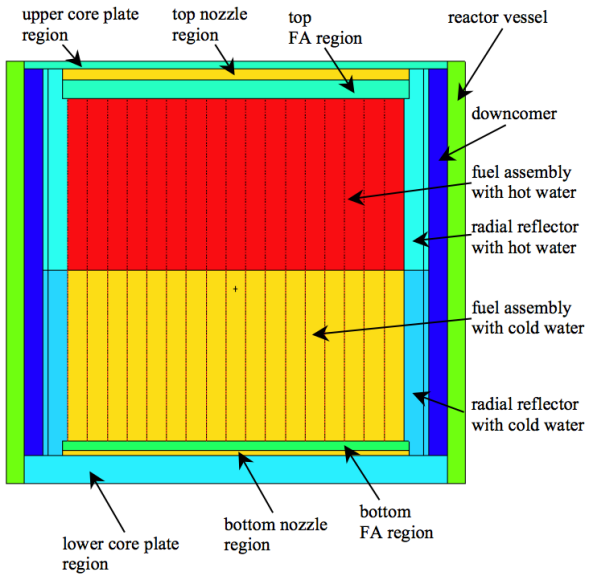
- For modeling random locations of fuel particles in new reactor fuel systems

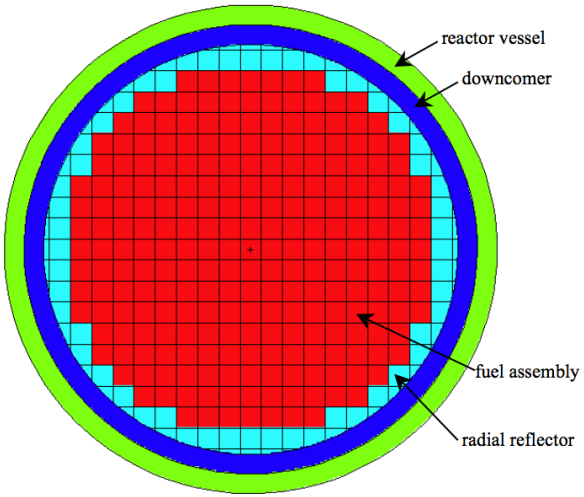
F.B. Brown & W.R. Martin, “Stochastic Geometry Capability in MCNP5 for the Analysis of Particle Fuel”, *Annals of Nuclear Energy*, Vol 31, Issue 17, pp 2039-2047 (2004).

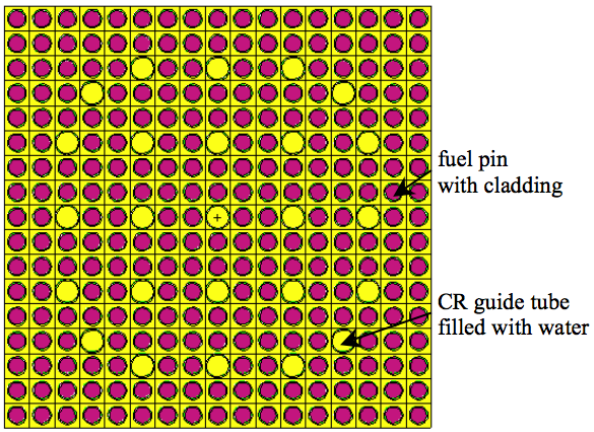
MCNP & the "Kord Smith Challenge"

Monte Carlo Codes
XCP-3, LANL

- **Full core, 3D benchmark for assessing MC computer performance**
  - **Specified by Hoogenboom & Martin (M&C 2009), rev for OECD 6/2010**
  - **LWR model: 241 assemblies, 264 fuel pins/assembly**
  - **Fuel contains17 actinides + 16 fission products; borated water**
  - **Detailed 3D MCNP model provided (Brown)**
    - **Mesh tallies for assembly powers, axially integrated**
    - **Mesh tallies for pin powers, (100 axial) x (264 fuel pins/assy) x (241 assy) = (63,624 pins) x (100 axial) = 6.3M pin powers**
    - **Runs easily on deskside computer (Mac Pro, 2 quad-core, 8 GB memory)**







MCNP & the "Kord Smith Challenge"

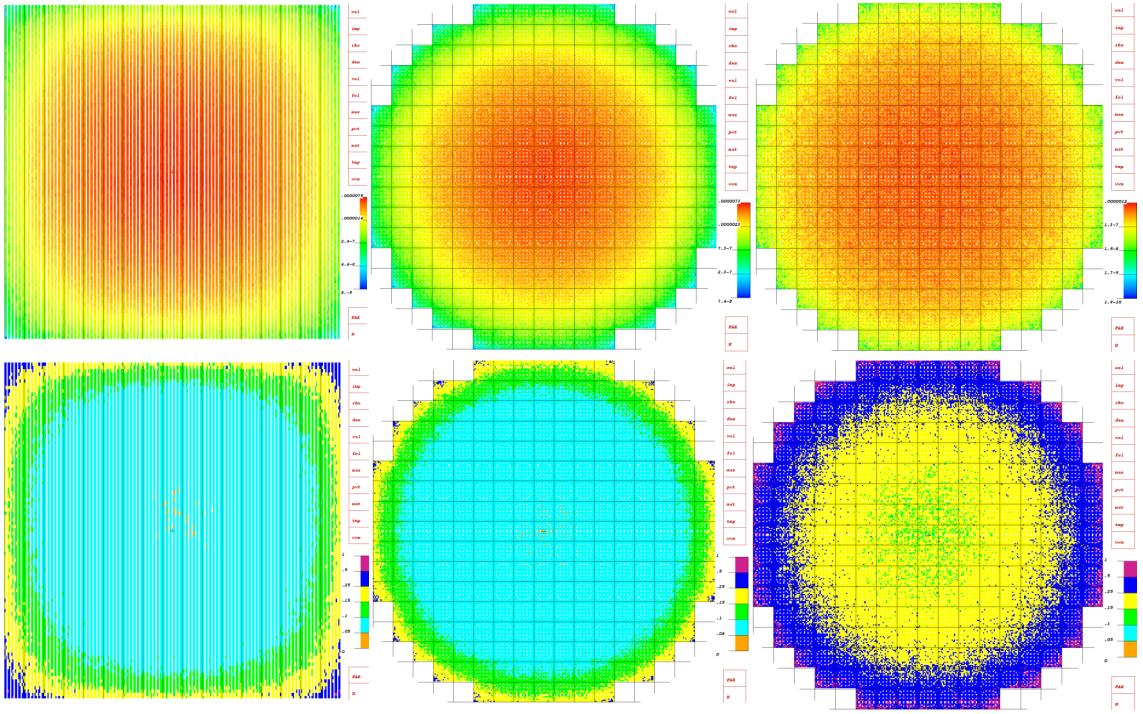
Monte Carlo Codes
XCP-3, LANL

Pin Powers & Std.Dev

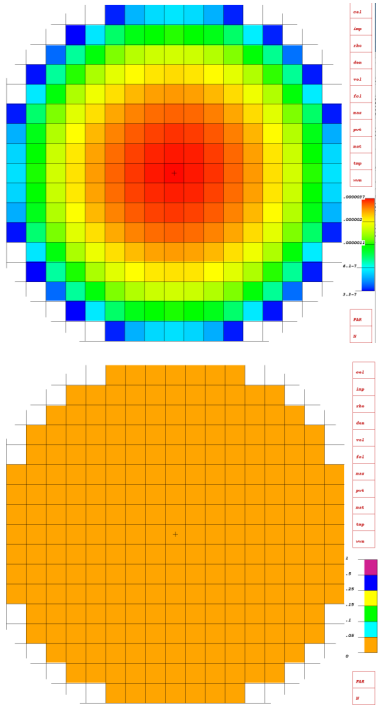
Axial

Mid

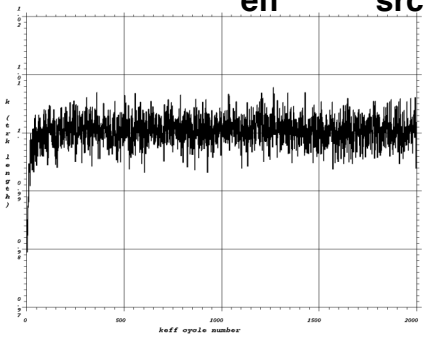
Top

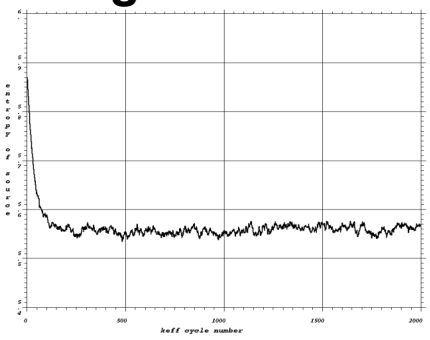


Assembly Power & Std.Dev



K_{eff} & H_{src} Convergence





200M neutrons
Mac Pro, 8 cpu

- **Some preliminary findings**

  - **MC21**
    - **See talk at PHYSOR-2010 by Kelly, Sutton, Trumbull, Dobreff**
    - **Roughly 6M neutrons/hr per cpu on Linux cluster**
    - **69 G neutrons per day on 400-cpu Linux cluster**

  - **MCNP5**
    - **Demo calculations, to help with problem specs & MCNP input**
    - **Roughly 3M neutrons/hr per cpu on Mac**
    - **.6 G neutrons per day on 8-cpu deskside Mac**
    - **Cluster results TBD**
    - **Runs easily on laptop or deskside computer** **(just not fast enough)**

Monte Carlo
-
Future Prospects

HPC Trends & Monte Carlo

- **Trends**
  - 1990s - teraflops, ~10^3 processors, clusters, homogeneous
  - 2000s - petaflops, ~10^4 processors, clusters, multicore, heterogeneous
  - 2010s - exaflops, ?????

- **Power limits require new architecture for exaflop systems**
  - Theme of 2009 Salishan conference
  - **10^6 - 10^9 processors**, clusters, multicore, heterogeneous
  - **Reduced memory/cpu**
  - **Possible limitations on node connectivity**

- **LANL Monte Carlo transport codes**
  - MCNP
    - Hierarchical parallelism on particles, MPI + threads, works well
    - Limitation: memory - must replicate problem on each node
  - MC++, IMC
    - Domain decomposition, MPI only, move particles among nodes
    - Limitation: load imbalances, poor scaling for many problems

Parameter Studies & Uncertainty Quantification

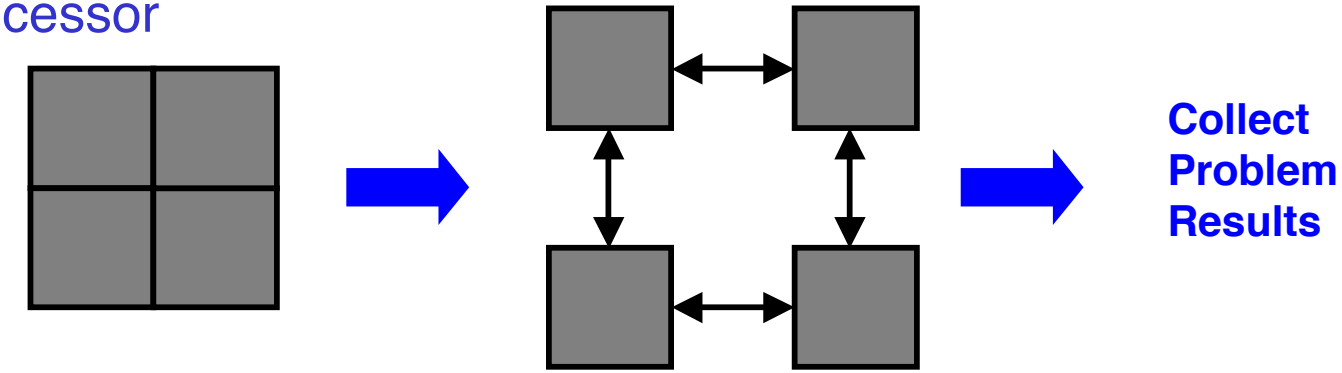
- **Future Exaflop systems may have 1 M to 1 G cpu-cores**
  - None of today's MC transport codes can use that many cpu-cores in a single large job
  - The only way to fully utilize the massive parallelism on exaflop systems with today's software is to run many 1000s or more parallel jobs

- **Parameter Studies**
  - Many 1000s of parallel jobs are run with different combinations of code input parameters to span the phase space of a multidimensional problem
- **Uncertainty Quantification**
  - Very many individual code input parameters are varied in separate calculations to assess the sensitivity of problem results to uncertainties in the code input parameters and then estimate the overall uncertainty on calculation of a physical problem
  - Many 1000s of parallel jobs

- **Pflop & Eflop computers could enable routine use of parameter studies & uncertainty quantification, turning around 1000s of parallel jobs in < 1 hr**

Parallel MC - Domain Decomposition

If a Monte Carlo problem is too large to fit into memory of a single processor

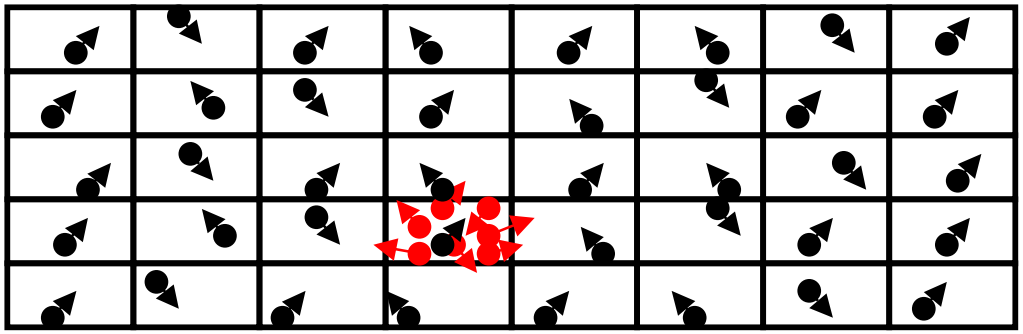


- Need periodic synchronization to interchange particles among nodes
- Use message-passing (MPI) to interchange particles

➜ **Domain decomposition is often used when the entire problem will not fit in the memory of a single SMP node**

Parallel MC - Domain Decomposition

- **Inherent parallelism is on particles**
  - Scales well for all problems
- **Domain decomposition**
  - Spatial domains on different processors
  - Scales OK for Keff or α calculations,
    where particle distribution among domains is roughly uniform
  - Does **not** scale for time-dependent problems
    due to severe load imbalances among domains
- **Domain decomposition - scaling with N processors**
  - **Best: performance ~ N (uniform distribution of particles)**
  - **Worst: performance ~ 1 (localized distribution of particles)**



Parallel MC - Data Decomposition

- Data is distributed by domain decomposition, but parallelism is on particles
- Solution ?

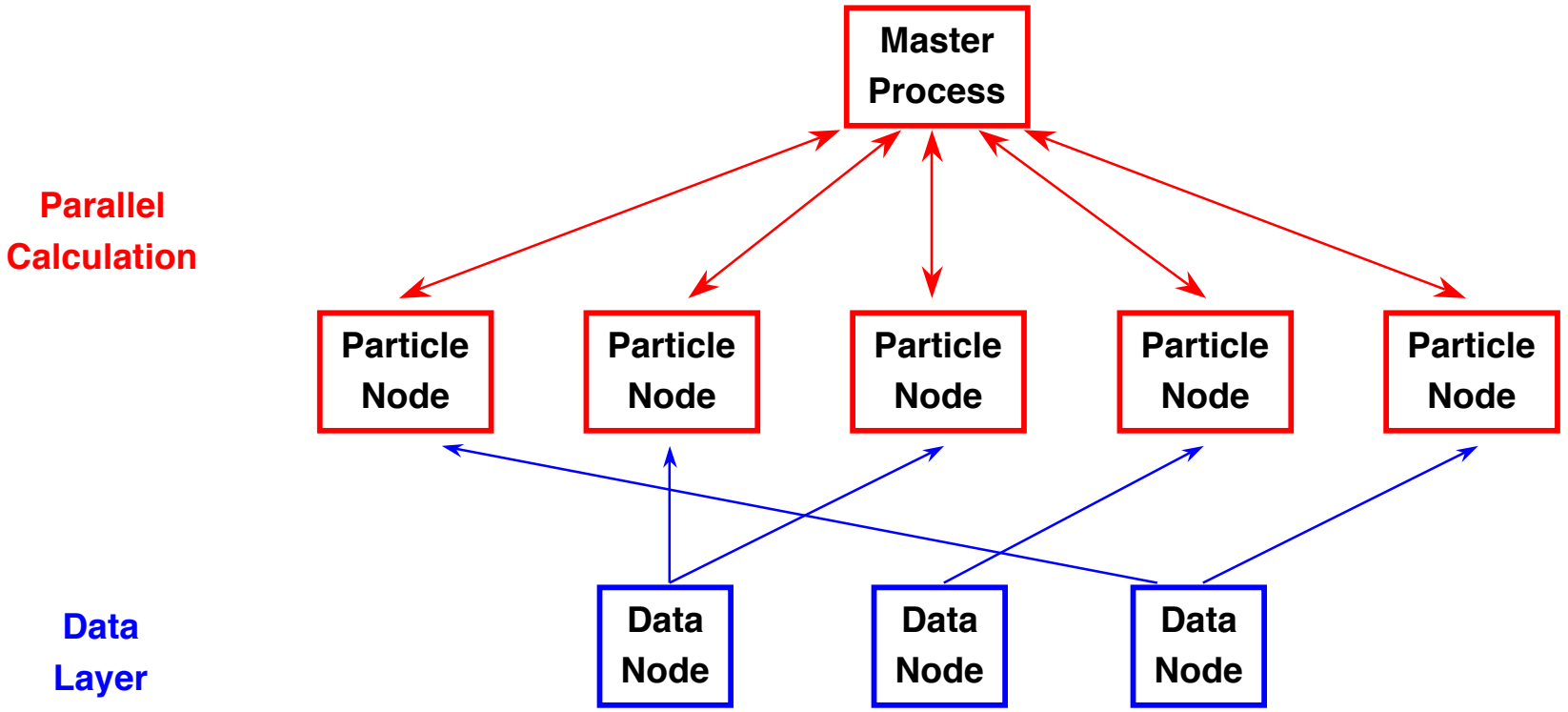
Parallel on particles + distributed data

- **Particle parallelism + Data Decomposition**
  - **Existing parallel algorithm for particles**
  - **Distribute data among processor nodes**
  - **Fetch the data to the particles as needed (dynamic)**
  - **Essentially same approach as used many years ago for CDC (LCM) or CRAY (SSD) machines**
  - **Scales well for all problems (but slower)**

See talk by Paul Romano - "Towards Scalable Parallelism in Monte Carlo Particle Transport Codes Using Remote Memory Access", SNA+MC 2010

Parallel MC - Data Decomposition

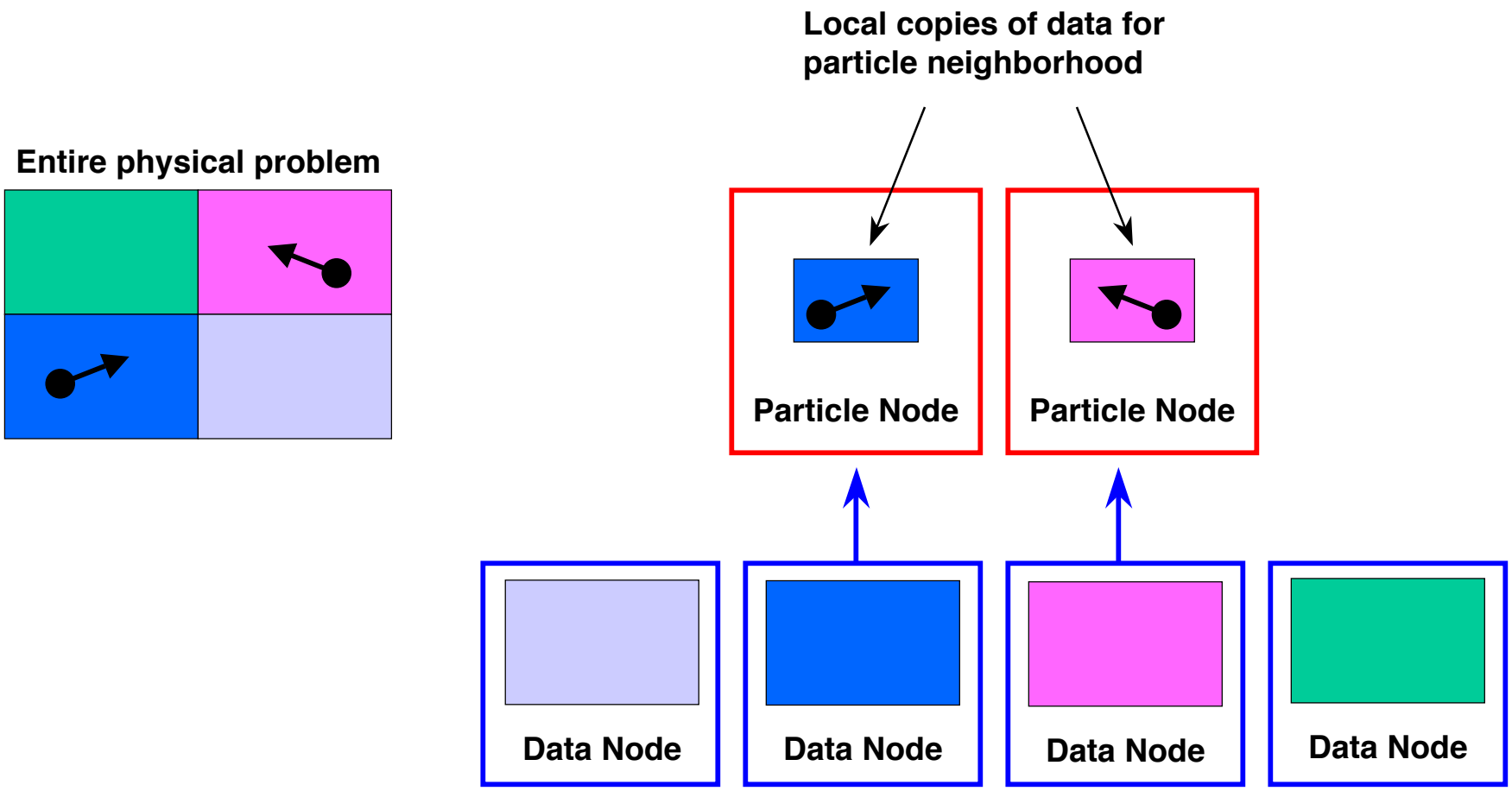
- **Particle parallelism + data decomposition -- logical view:**



- **Mapping of logical processes onto compute nodes is flexible:**
  - Could map particle & data processes to **different** compute nodes
  - Could map particle & data processes to **same** compute nodes
- **Can replicate data nodes if contention arises**

Parallel MC - Data Decomposition

- **Particle parallelism + data decomposition**



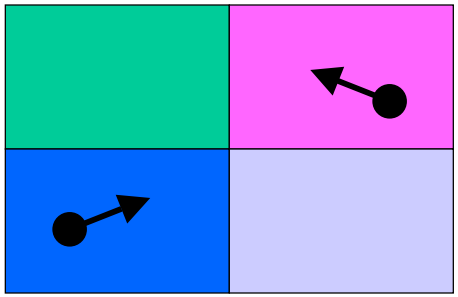
- **History modifications for data decomposition**

```
source

while   wgt > cutoff

.           compute distances & keep minimum:
.                       dist-to-boundary
.                       dist-to-time-cutoff
.                       dist-to-collision
.                       dist-to-data-domain-boundary

.           move particle
.           pathlength tallies

.           if    distance == dist-to-data-domain-boundary
.                       fetch new data

.           collision physics
.           roulette & split
. . .
```

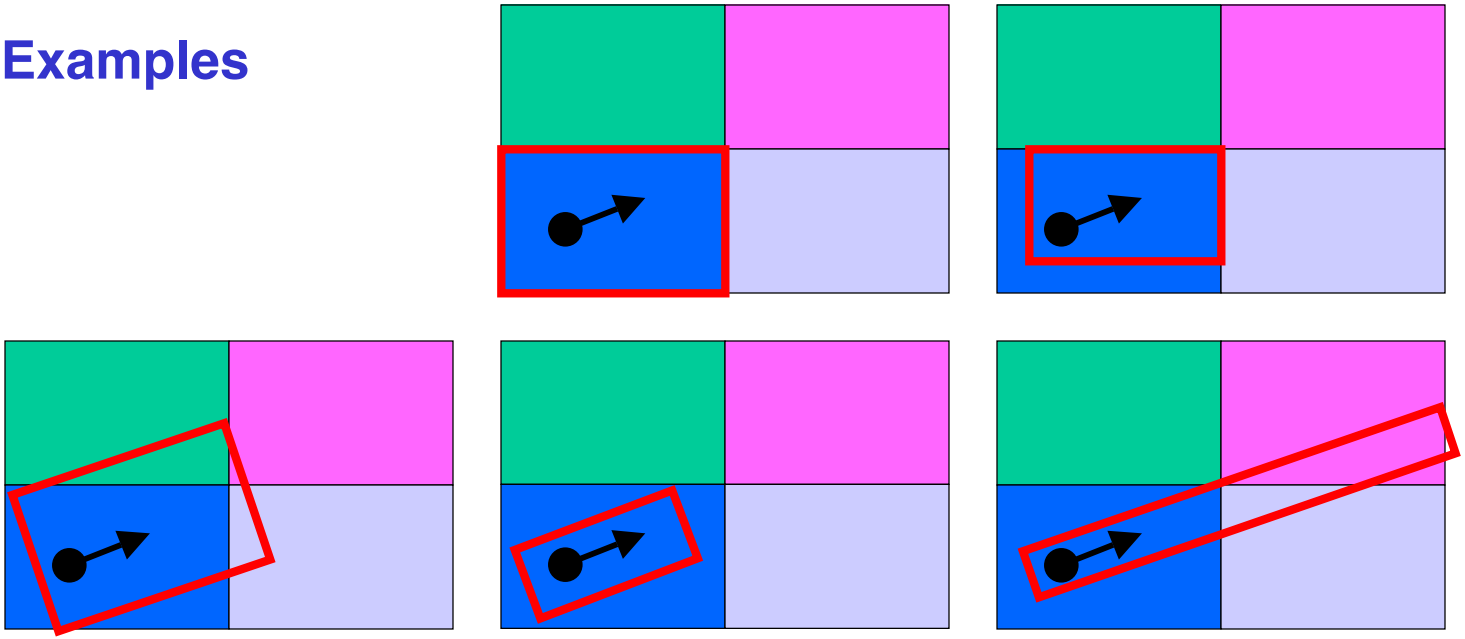
Parallel MC - Data Decomposition

- **Data windows & algorithm tuning**
  - Defining the "particle neighborhood" is an art
  - Anticipating the flight path can guide the pre-fetching of blocks of data
  - Tuning parameters:
    - How much data to fetch ?
    - Data extent vs. particle direction ?

Entire physical problem



- **Examples**



Conclusions

For Monte Carlo problems which can fit in memory:

- Concurrent scalar jobs - ideal for Linux clusters
- **Master/slave parallel algorithm (replication)** works well
  - Load-balancing: Self-scheduling
  - Fault-tolerance: Periodic rendezvous
  - Random numbers: Easy, with LCG & fast skip-ahead algorithm
  - Tallies: Use OpenMP "critical sections"
  - Scaling: Simple model, more histories/slave + fewer rendezvous
  - Hierarchical: Master/slave MPI, OpenMP threaded slaves
  - Portability: MPI/OpenMP, clusters of anything

For Monte Carlo problems too large to fit in memory:

- **Spatial domain decomposition** (with some replication) can work for some problems
- **Particle parallelism + data decomposition** is a promising approach which should scale for all problems

References

- Salishan conferences on HPC & future computers
  www.lanl.gov/orgs/hpc/salishan

- Gordon Bell, “A Seymour Cray Perspective”, Seymour Cray Lecture Series, University of Minnesota, Nov 10, 1997.