ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

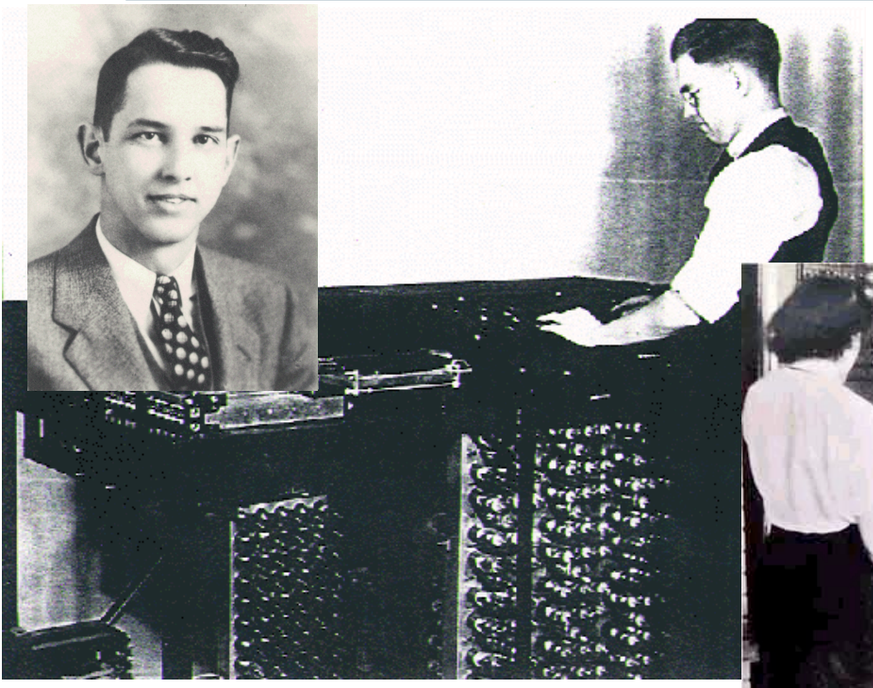
CSCS
Swiss National Supercomputing Centre

High Performance Computing:
A Tutorial for Scientists

Thomas C. Schulthess, CSCS / ETH Zurich
schulthess@cscs.ch

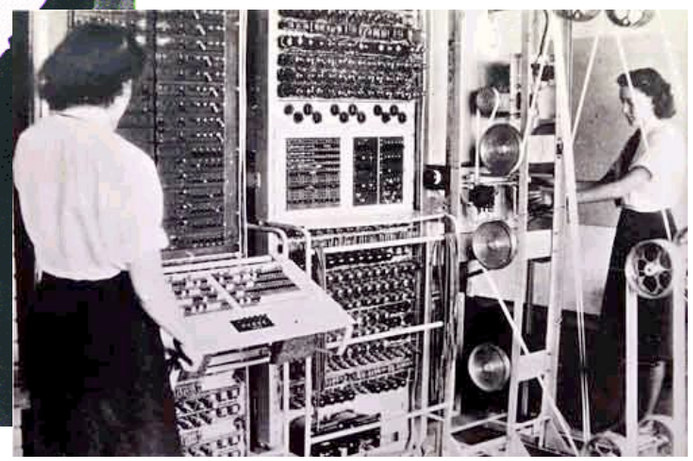
Numerical Approaches to Quantum Many-Body Systems
Institute for Pure & Applied Mathematics, UCLA, 01/23/2009

Electronic computing: the beginnings



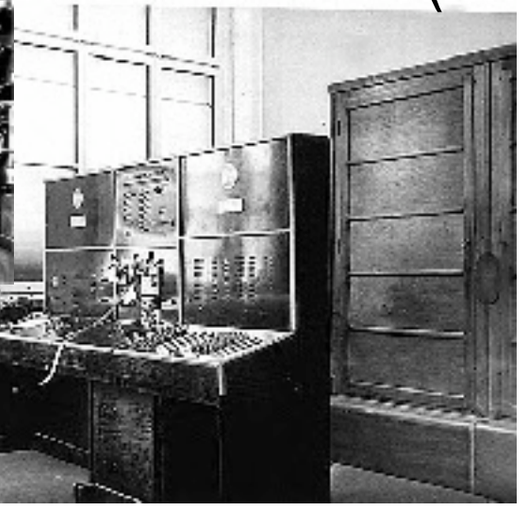
1939-42: Atanasoff-Berry Computer - Iowa State Univ.

1940: Konrad Zuse's Z2 - Germany



1943/44: Colossus Mark 1&2 - Britain

Zuse and Z3 (1941)



Z4 @ ETH (1950-54)

1945-51: UNIVAC I

Eckert & Mauchly - "first commercial computer"

1945: John von Neumann report that defines the "von Neuman" architecture

Since the dawn of High-performance computing:
Supercomputing at Los Alamos National Laboratory

1946: ENIAC

1952: MANIA C I

1957: MANIAC II

...

1974: Cray 1 - vector architecture

...

1987: nCUBE 10 (SNL) - MPP architecture

1993: Intel Paragon (SNL)

1993: Cray T_3D

...

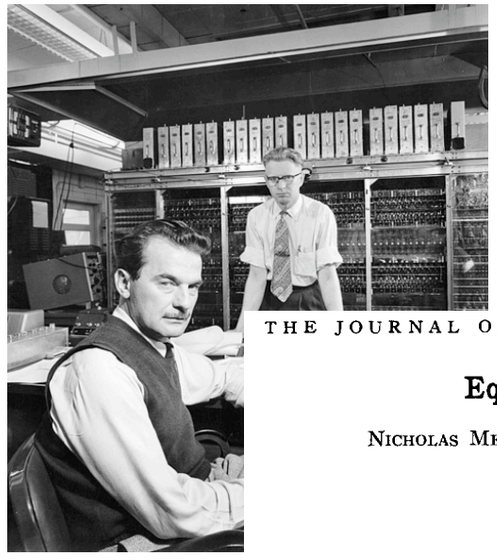
2004: IBM BG/L (LLNL)

2005: Cray Redstorm/XT3 (SNL)

2007: IBM BG/P (ANL)

2008: IBM “Roadrunner”

2008: Cray XT5 (ORNL)



Nicholas Metropolis: group leader in LANL’s **T Division** that designed MANIAC I & II

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

2002: Japanese Earth Simulator - Sputnik shock of HPC

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

Outline / goals

- History of scientific computing and HPC
- Overivew of typical HPC architectures
  - Important terminology used later
- Longterm development trends in HPC
  - Understand the brutal facts of HPC today
- Parallel programming models
  - There is more to parallel programming than MPI and OpenMP
- Mapping methods/algorithm onto hardware
  - Importance of computational methematics and computer science
- Example: running QMC/DCA at scale
  - Even simple things can be challenging on 150,000 processors

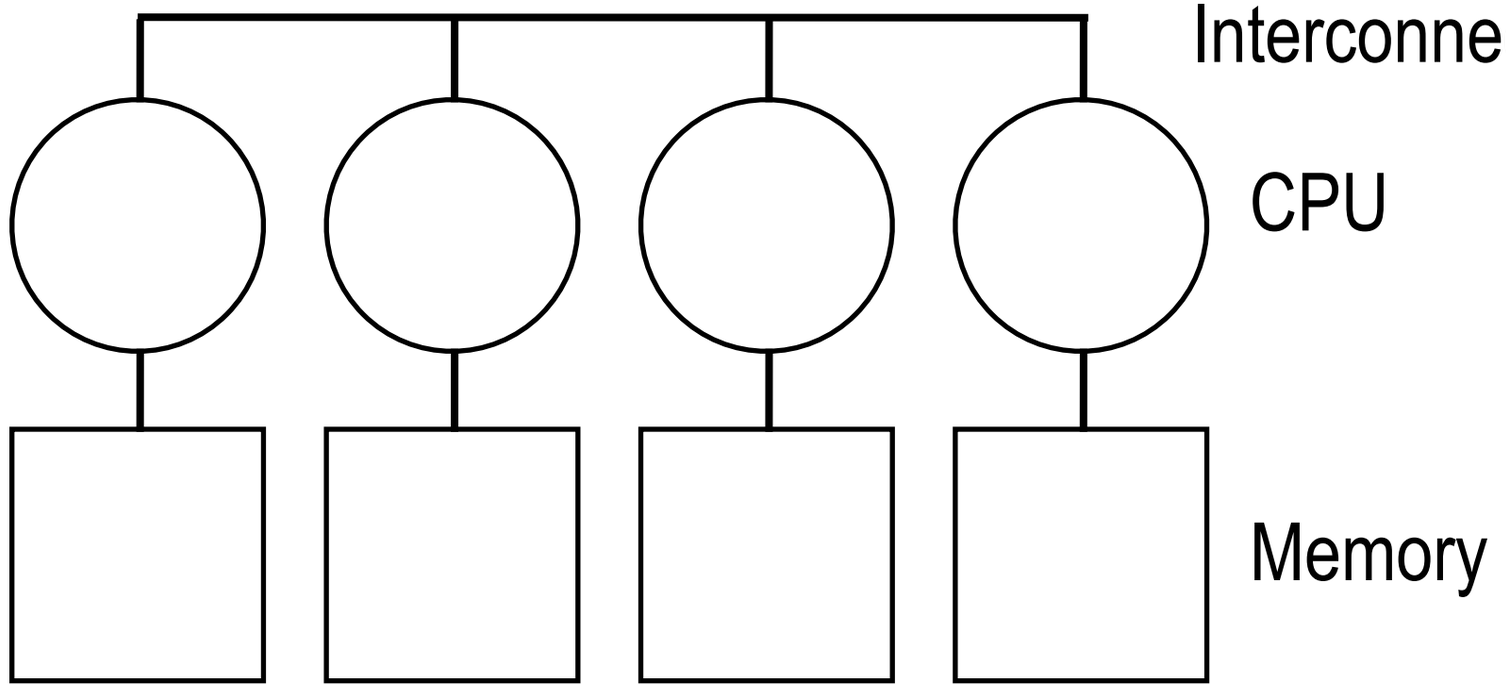
Scalar / superscalar / vector processors

- Scalar processor: process one data item (integer / floating point number) at a time
- Vector processor: a single instruction operates on many data items simultaneously
- Typical processor today: "pipelined superscalar"
  - Superscalar: simultaneously dispatch multiple instruction to redundant functional units (multiplier or adder)
  - Pipeline: set of processing elements connected in a series
  - Example: 2 multiplies and two add per cycle
    (4 floating point operations per cycle)

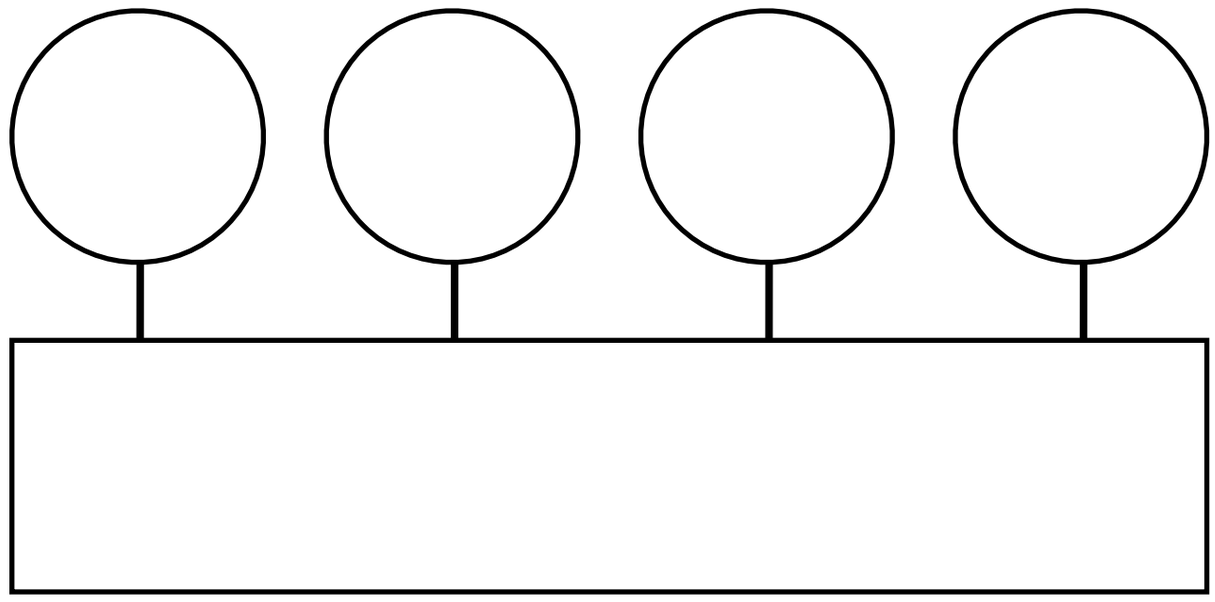
The good news: compiler-level optimization will take care of this!

Distributed vs. shared memory architecture

Distributed memory

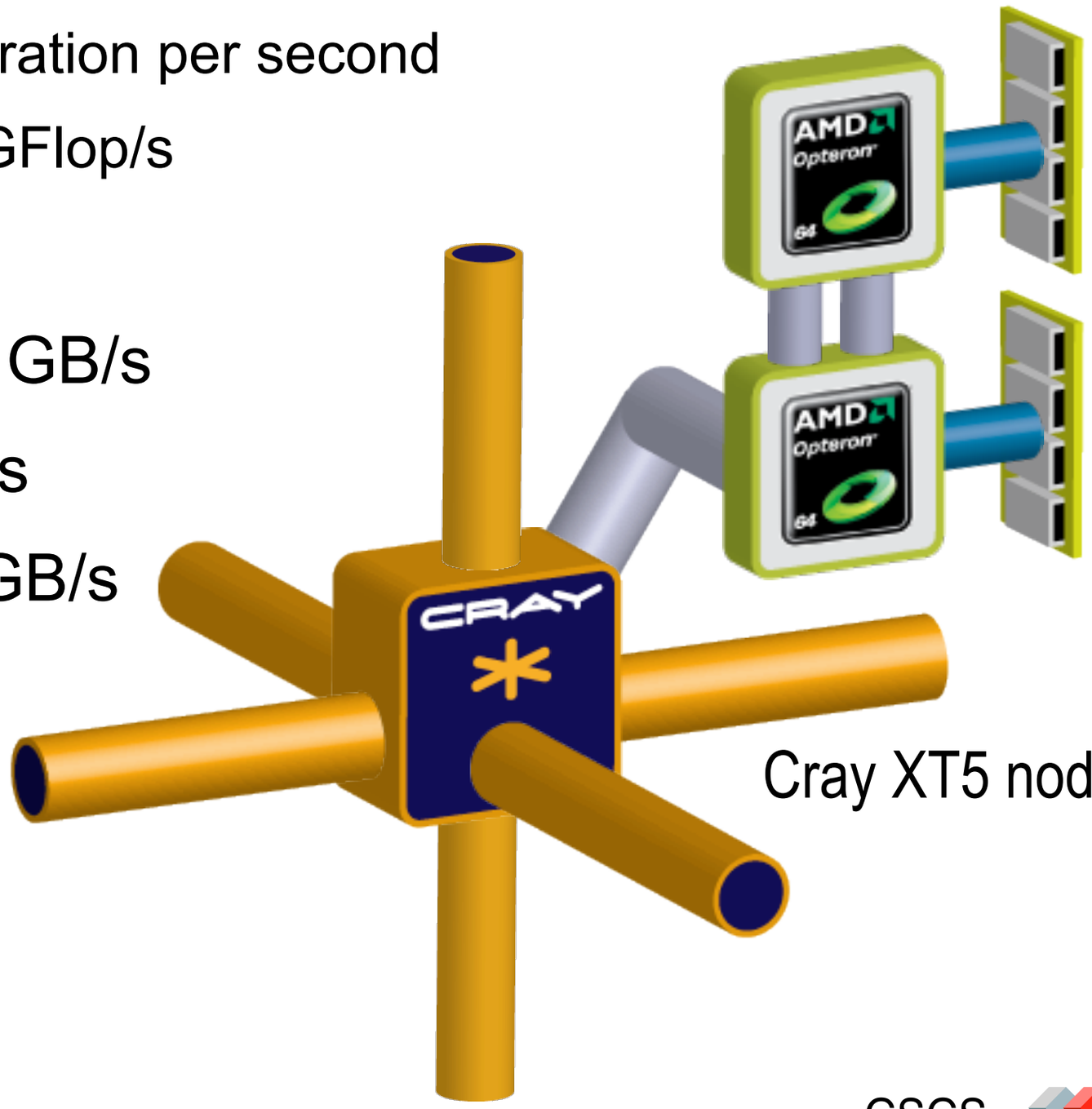


Shared memory



Aspects of performance - typical values in 2009

- Floating point (integer) performance: 2 or 4 per cycle
  - Flop/s = floating point operation per second
  - 2.4 GHz processors: 9.6 GFlop/s
- Memory latency: ~50 ns
- Memory bandwidth: ~10 GB/s
- Network latency ~2-10 µs
- Network bandwidth: ~5 GB/s
- Disk access time ~ ms
- I/O bandwidth ~ MB/s



Cray XT5 node

Outline / goals

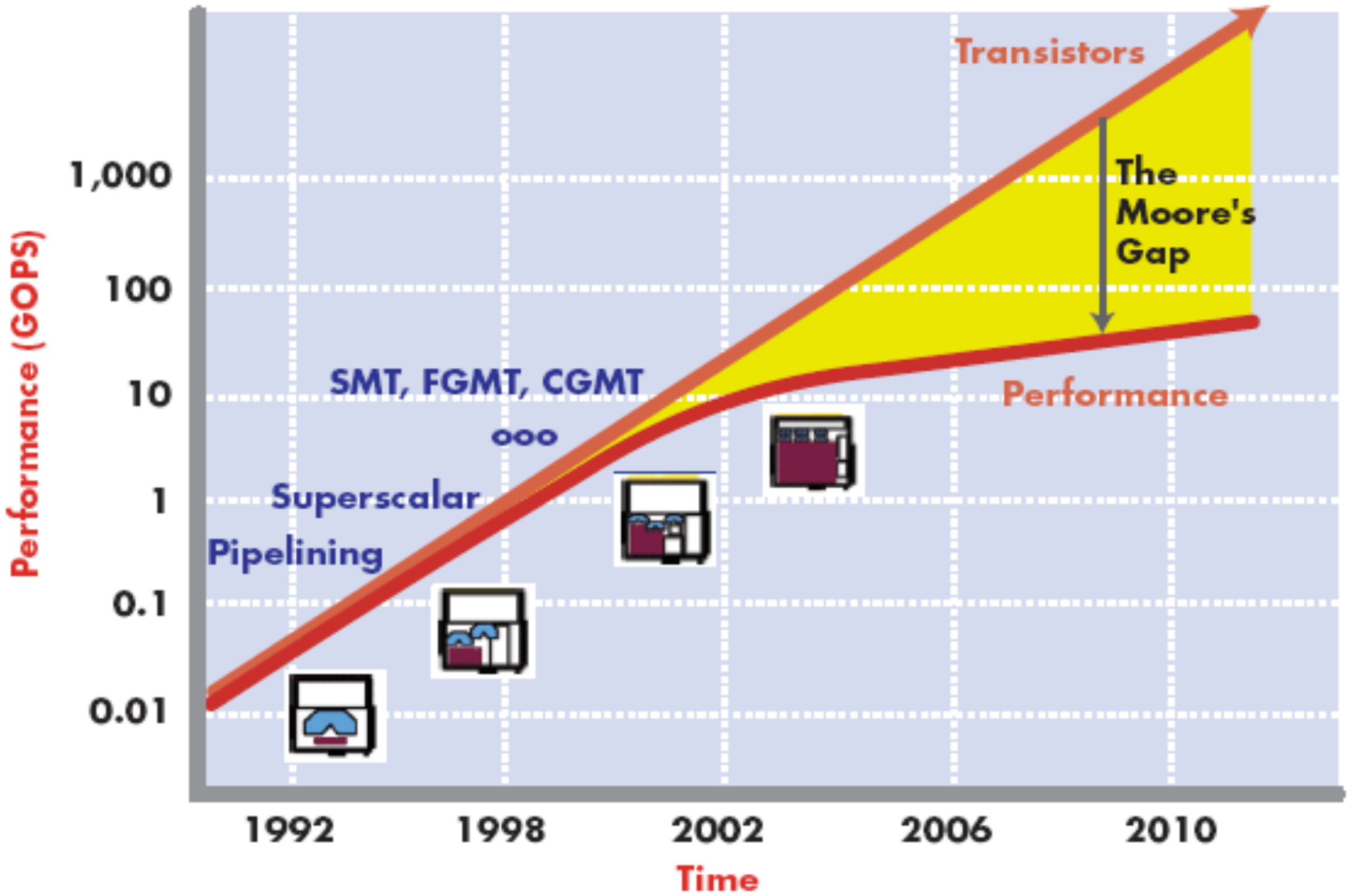
- History of scientific computing and HPC
- Overivew of typical HPC architectures
  - Important terminology used later
- Longterm development trends in HPC
  - Understand the brutal facts of HPC today
- Parallel programming models
  - There is more to parallel programming than MPI and OpenMP
- Mapping methods/algorithm onto hardware
  - Importance of computational methematics and computer science
- Example: running QMC/DCA at scale
  - Even simple things can be challenging on 150,000 processors

Computers in the past and today

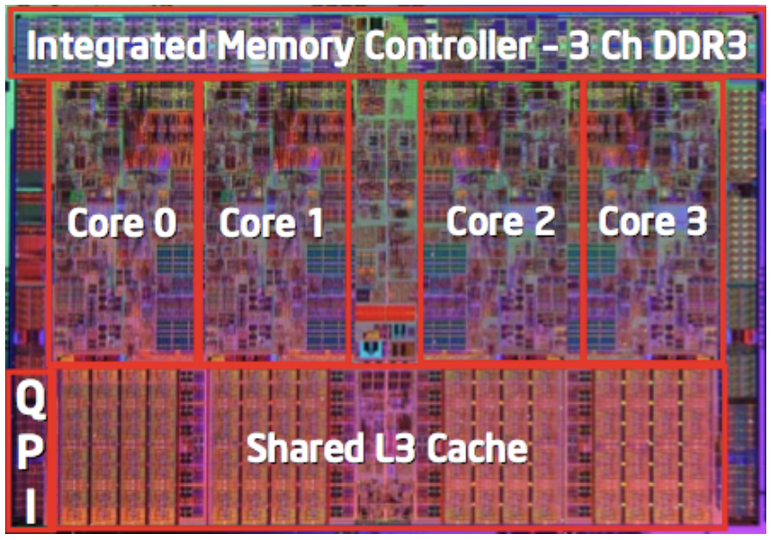
	1970s (*)	my laptop	improvement
clock (CPU)	6 MHz	2GHz	300 x
Flop/s	6 MFlop/s	8 GFlop/s	10^3 x
RAM	128kB	~2GB	10^5 x
Mem. latency	850ns	~50ns	20 x

(*) Charles Thacker's computer in the 1970s

Single processor performance is no longer tracking Moore's Law

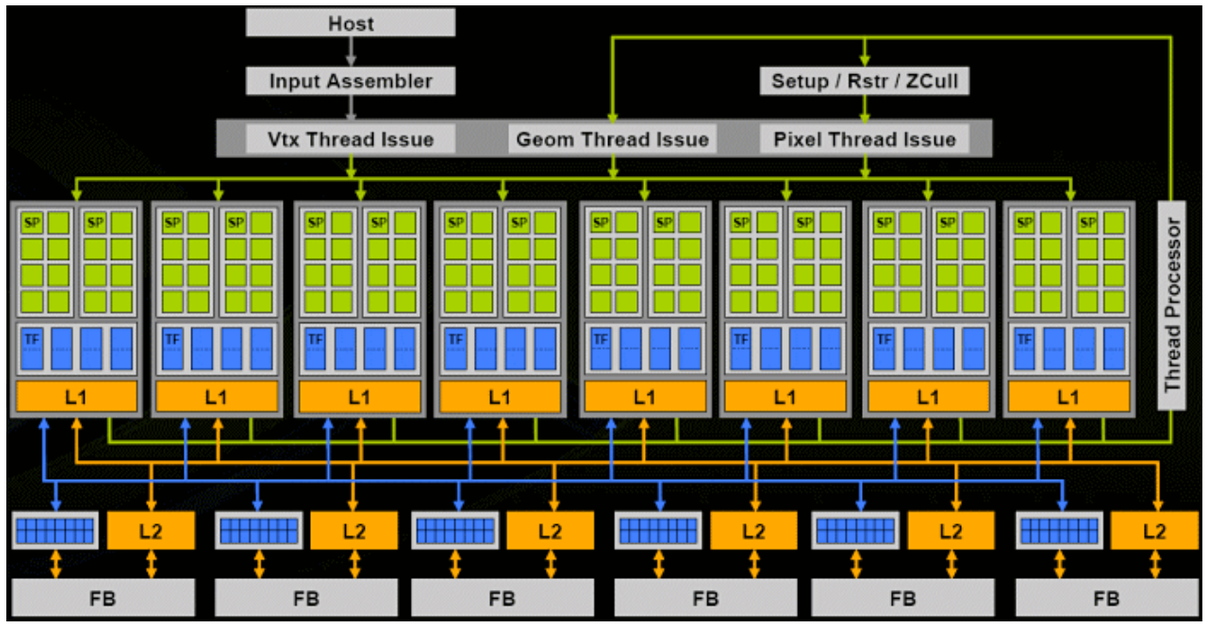


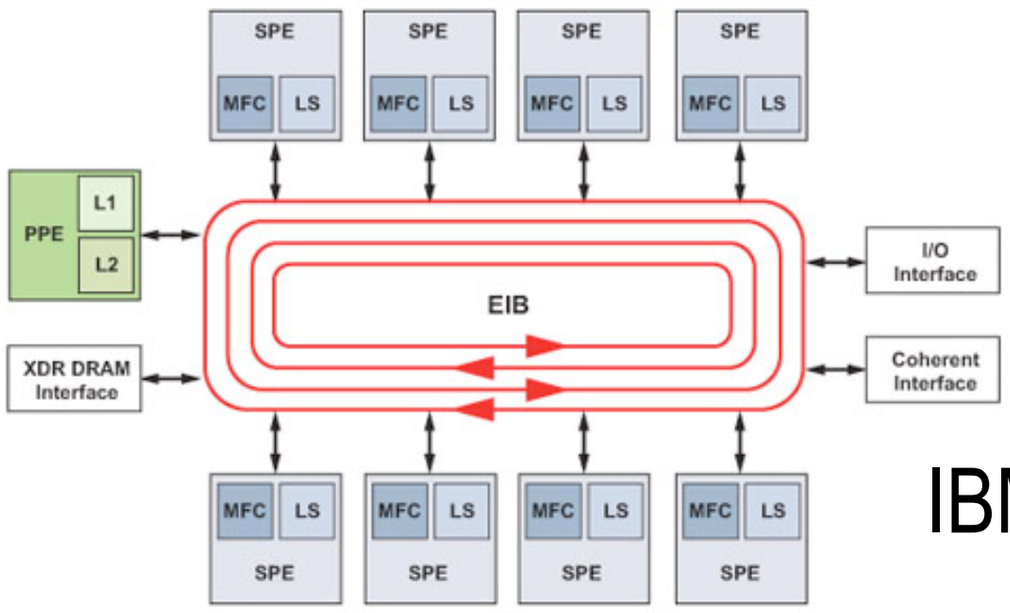
Multi-core and heterogeneous processors architectures



Multi-core processors: OpenMP (or just MPI)

NVIDIA G80 GPU: CUDA, cuBLAS

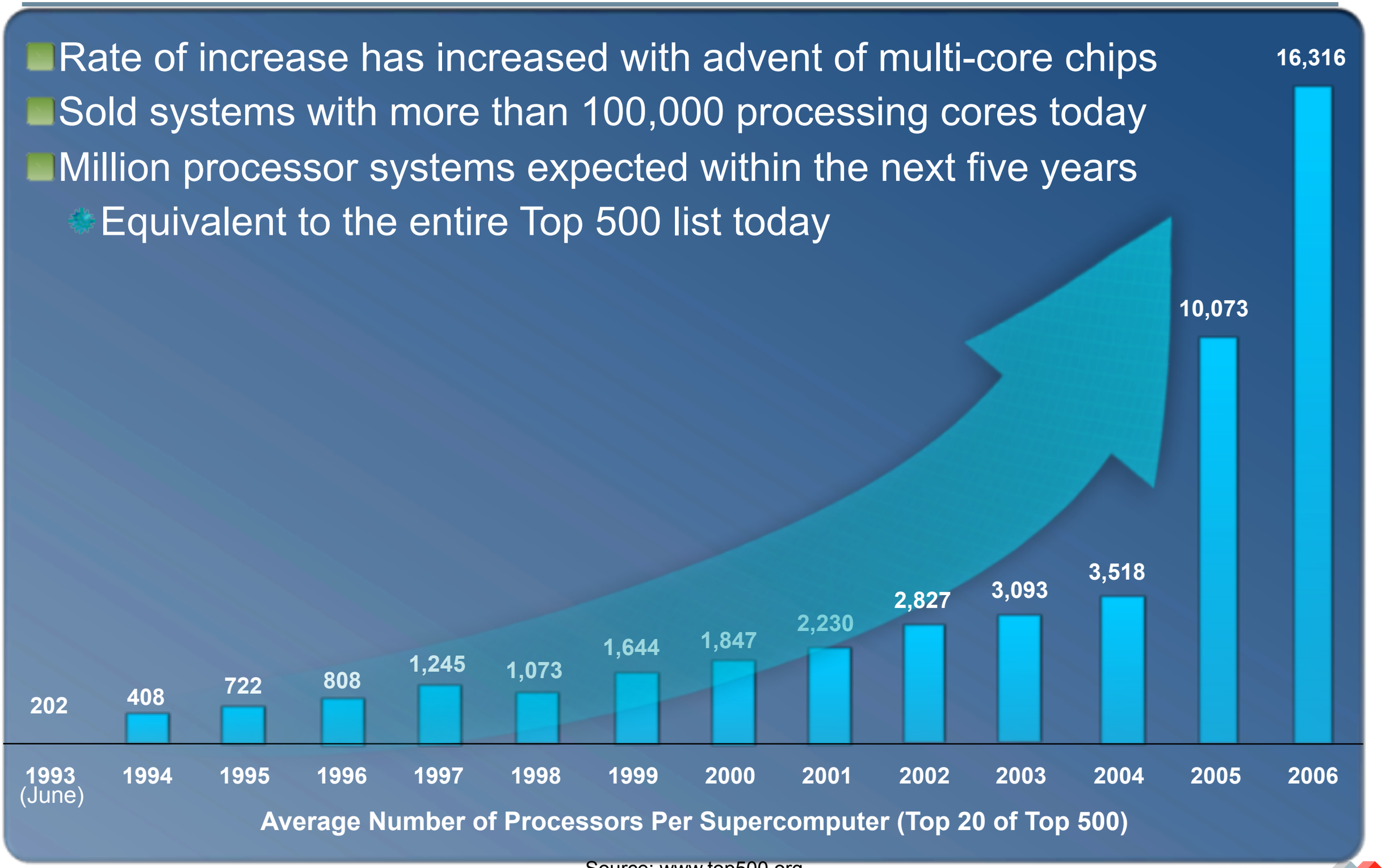




IBM Cell BE: SIMD, threaded prog.

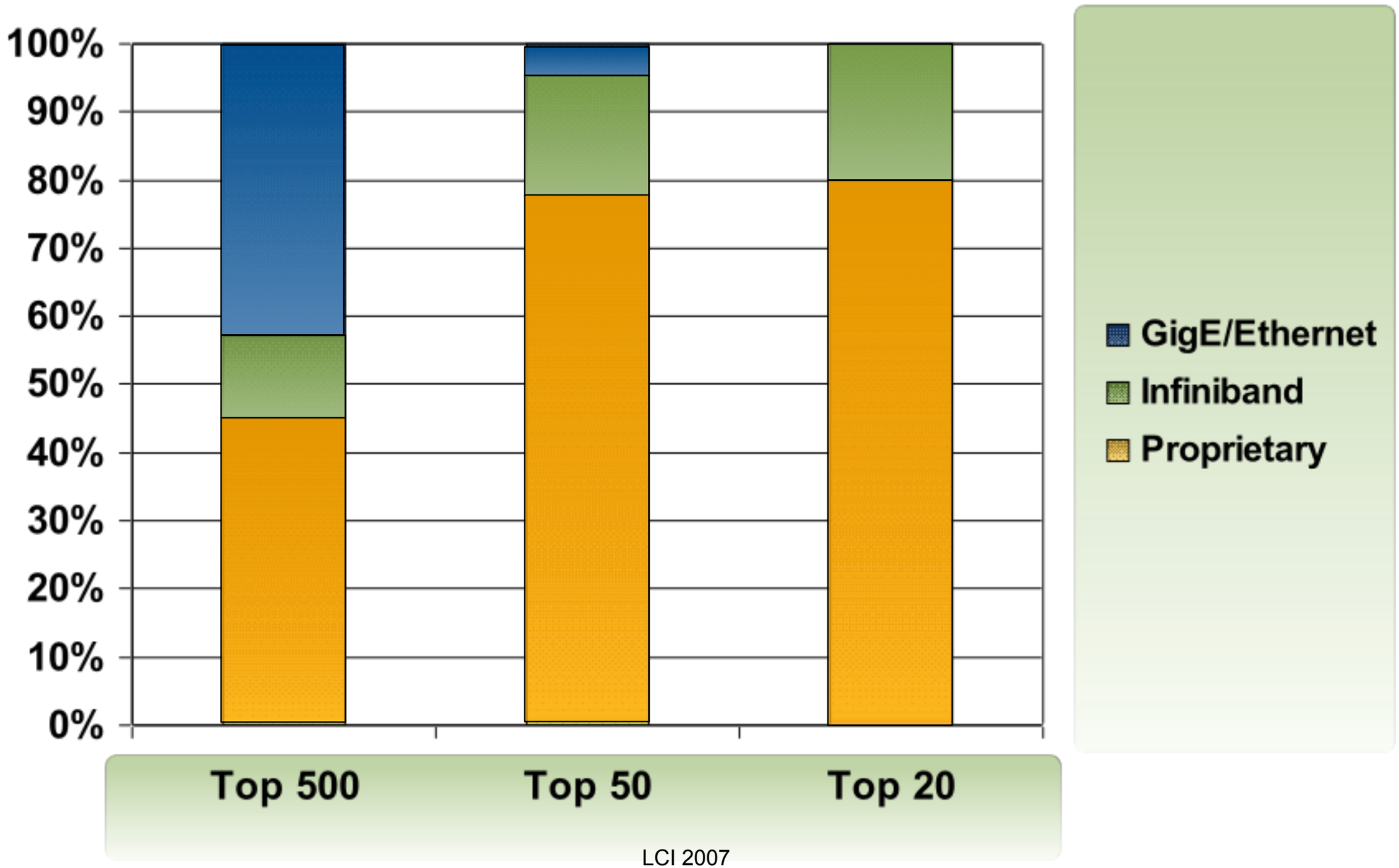
Explosion in the number of processing cores

- Rate of increase has increased with advent of multi-core chips
- Sold systems with more than 100,000 processing cores today
- Million processor systems expected within the next five years
  - Equivalent to the entire Top 500 list today



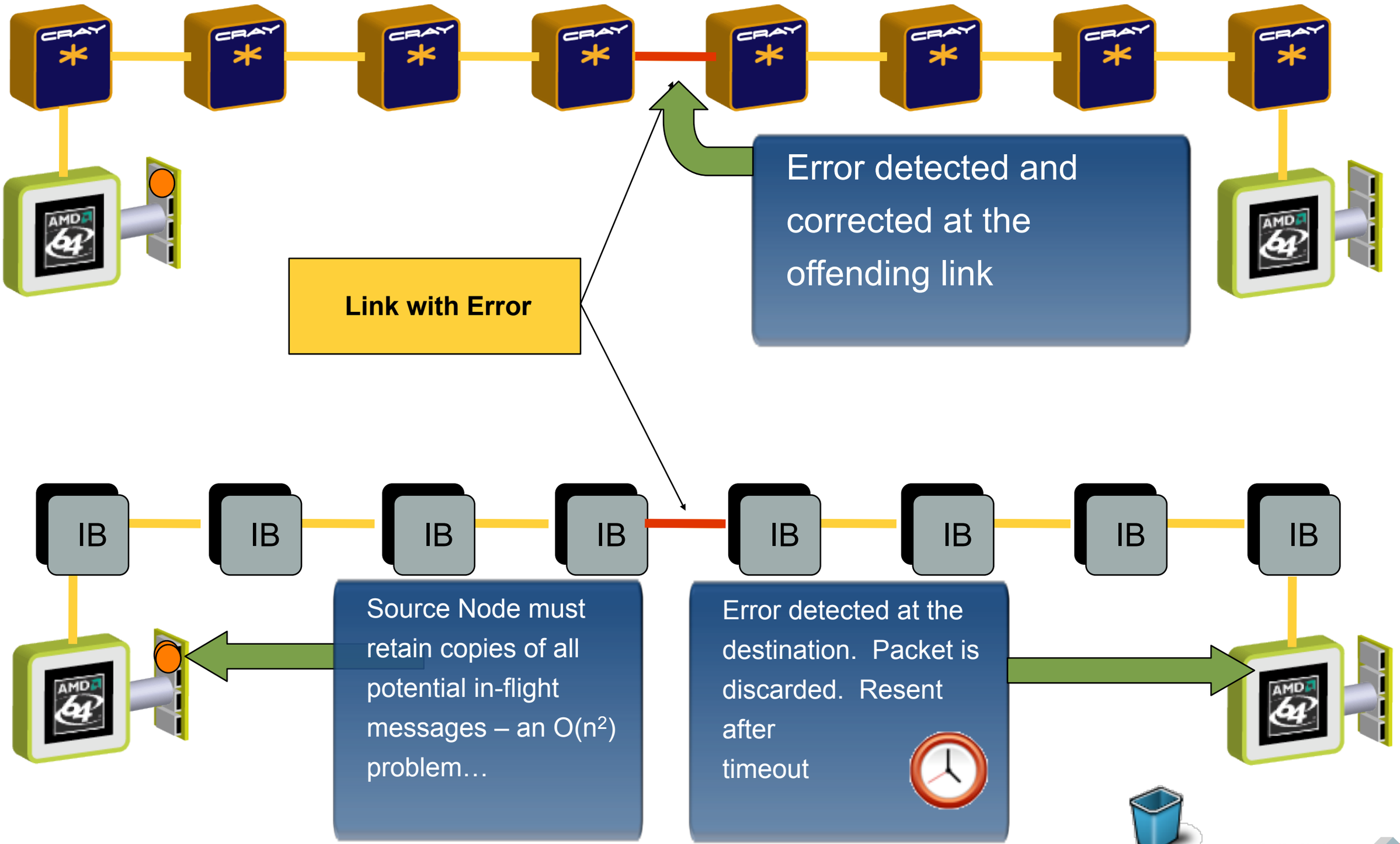
Source: www.top500.org

Interconnects in the TOP500 systems



LCI 2007

Complexity of interconnect



Getting ready for Quad Core

- Bytes/flops will decrease
  - XT3 – 5 GB/sec/2.6 GHZ* 2Flops/clock
    - 1 Byte/flop
  - XT4 (dual) – 6.25GB/sec/2.6 GHZ* 2Flops/clock/2 processors
    - ½ Byte/flop
  - XT4 (quad) – 8 GB/sec/2.2GHZ*4Flops/clock/4 processors
    - ¼ Byte/flop
- Interconnect Bytes/flop will decrease
  - XT3 – 2 GB/sec/2.6 GHZ* 2Flops/clock
    - 1/3 Bytes/flop
  - XT4 (dual) – 4 GB/sec/2.6 GHZ* 2Flops/clock/2 processors
    - 1/3 Bytes/flop
  - XT4 (quad) – 4 GB/sec/2.2GHZ*4Flops/clock/4 processors
    - 1/10 Byte/flop

HPC in the age of massively parallel processing (MPP) architectures: what does this really mean?

Evolution of the fastest sustained performance in real simulations

~1 Exaflop/s
~10^7 processing units

1.35 Petaflop/s
Cray XT5
1.5 10^5 processor cores

1.02 Teraflop/s
Cray T_{3D}
1.5 10^3 processors

1.5 Gigaflop/s
Cray YMP
0.8 10^1 processors

Summary: Brutal fact of modern HPC

- Mind boggling numbers of processing processing units
- Processor complexity (multi-core, heterogeneous)
- Interconnect is a non-trivial part of the HPC system
- Accessing memory is prohibitively expensive compared to the cost of floating point operations
  - 1960s: transistors were expensive, memory access was cheap
  - today: transitors are cheap, memory access is expensive

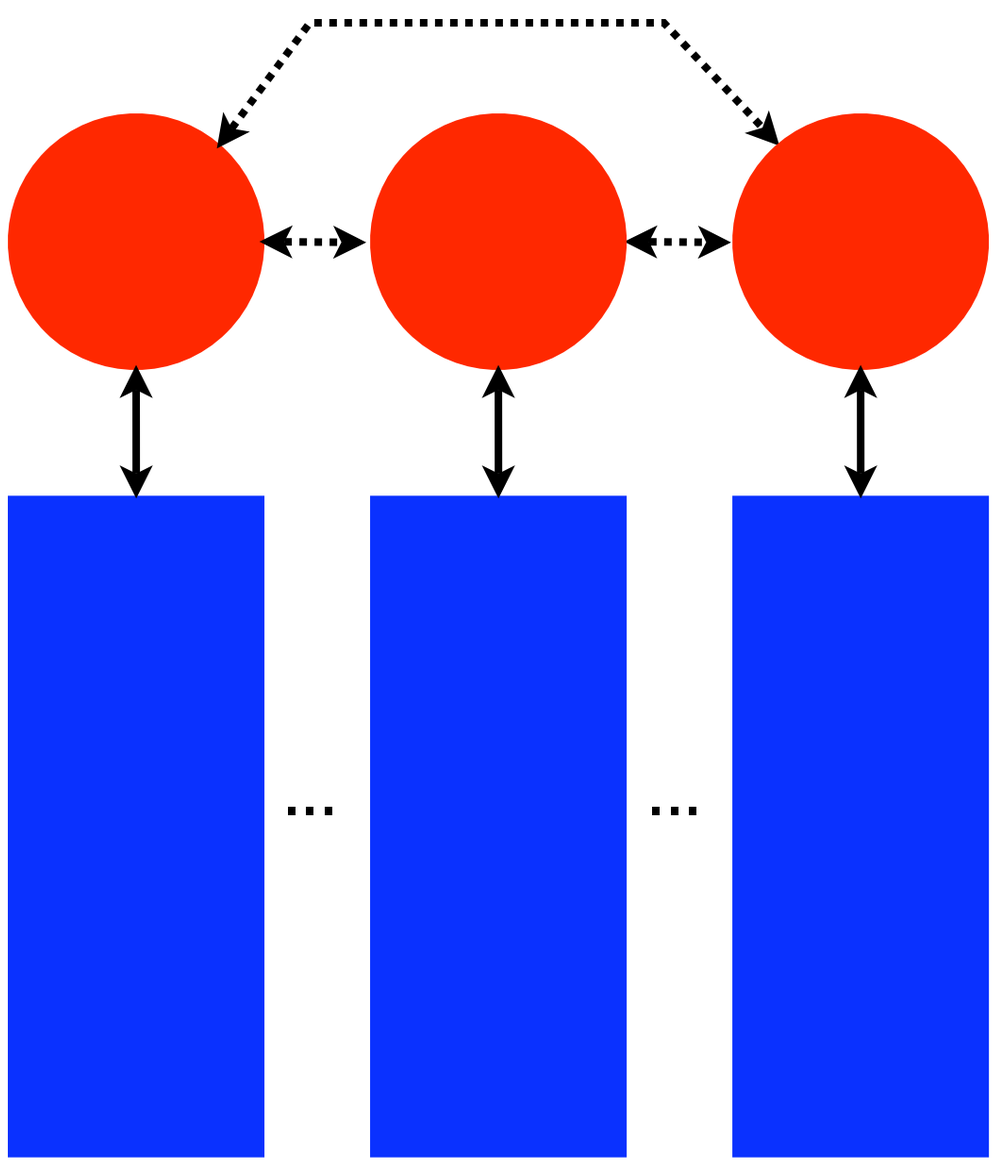
Key aspect of programming in HPC systems:

All about anaging resources

Outline / goals

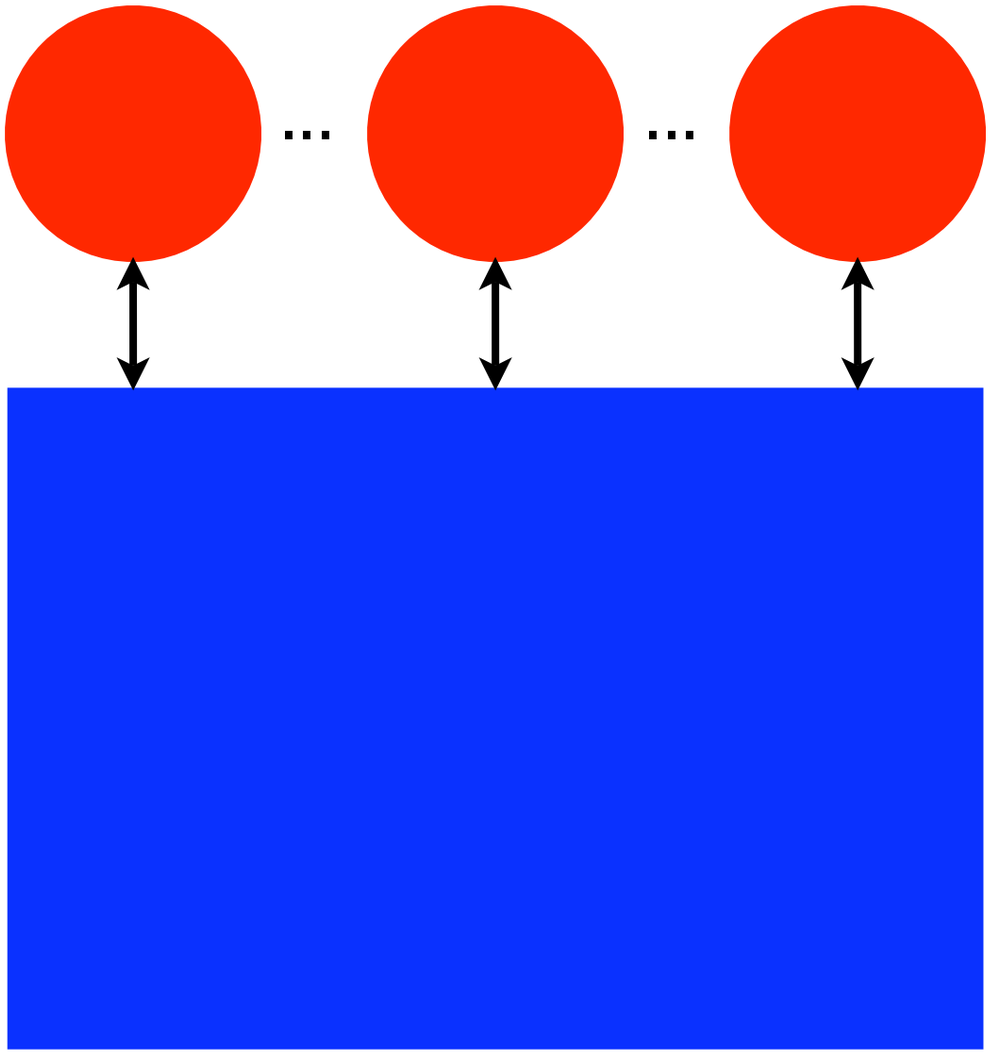
- History of scientific computing and HPC
- Overivew of typical HPC architectures
  - Important terminology used later
- Longterm development trends in HPC
  - Understand the brutal facts of HPC today
- Parallel programming models
  - There is more to parallel programming than MPI and OpenMP
- Mapping methods/algorithm onto hardware
  - Importance of computational methematics and computer science
- Example: running QMC/DCA at scale
  - Even simple things can be challenging on 150,000 processors

Programming models (I): message passing



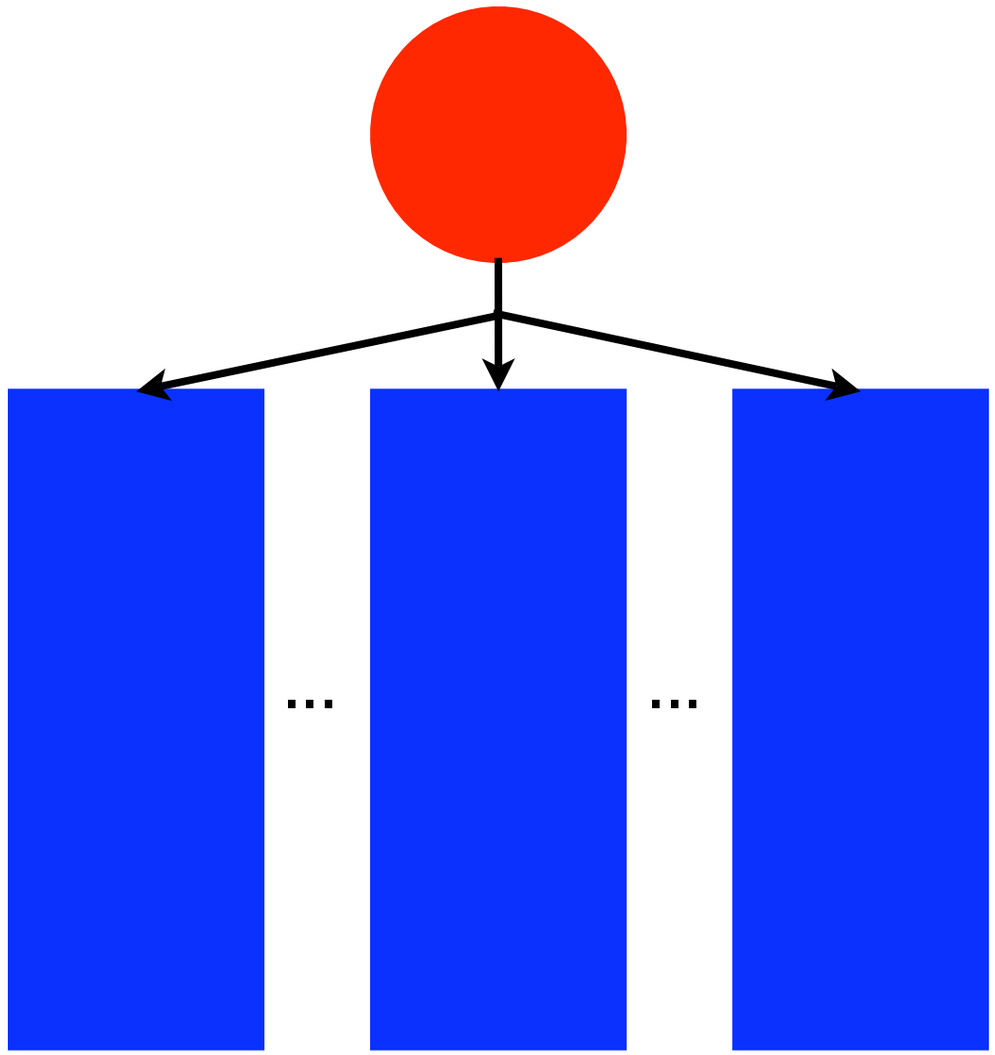
- Concurrent sequential processes cooperating on the same task
- Each process has own private space
- Communication is two-sided through send and receive
  - Large overhead!
- Lots of flexibility in decomposing large problems, however, provides only fragmented view of the problem
  - All burden is placed on the programmer to maintain global view
- Examples are message passing libraries like MPI or PVM

Programming models (II): shared memory



- Multiple independent threads operate on same shared address space
- Easy to use since there is only one type of memory access
  - One-sided remote access (low overhead)
- Application view remains integrated (global view)
- Shared memory hardware doesn't scale (local & remote memory access)
- It is difficult to exploit inherent data locality - degradation of performance!
- Examples are OpenMP or Pthreads
  - Compiler directive used with C, Fortran, ...

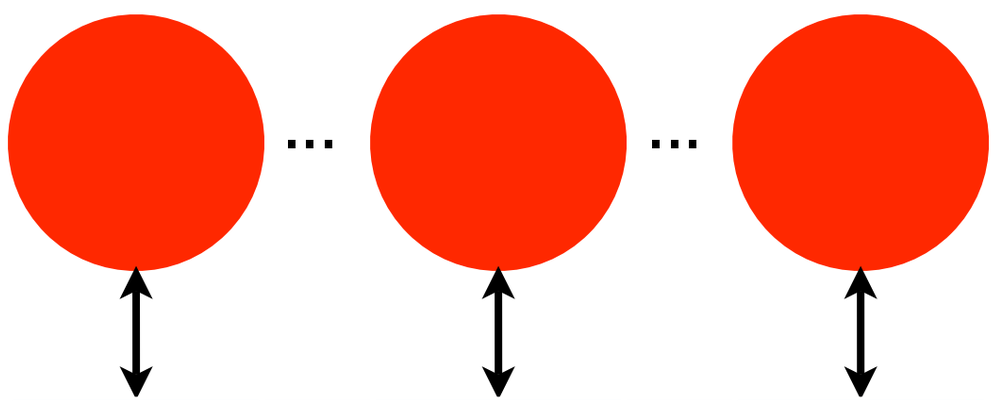
Programming models (III): data parallel

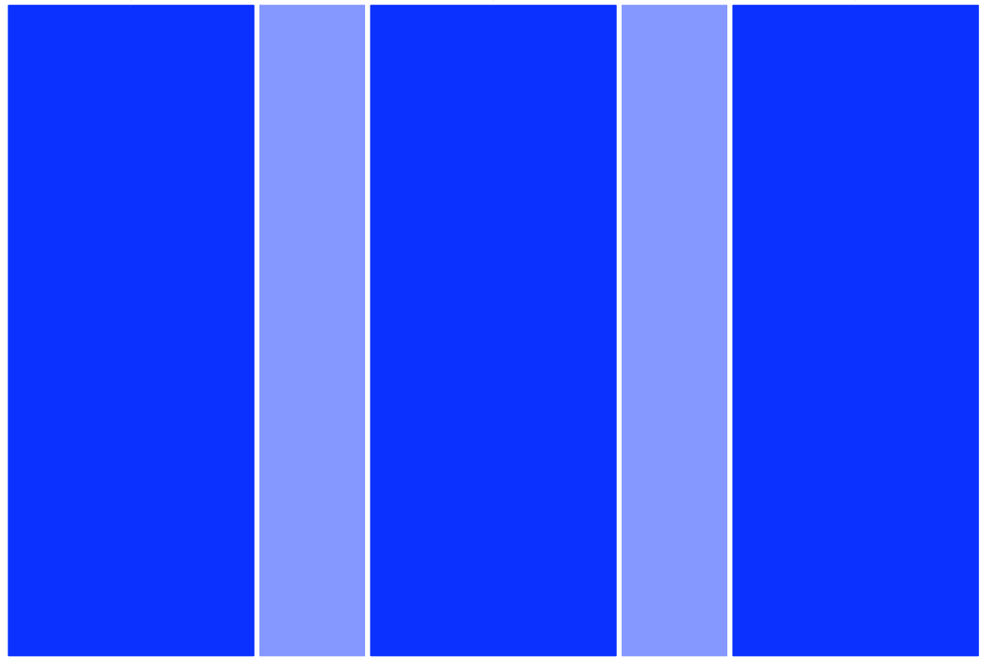


- Concurrent processing of many data elements in the same manner
- Executing only one process (on many processors)
- Major drawback: does not permit independent branching
  - Not good for problems that are rich in functional parallelism
- Popular examples are C* and HPF

Programming models (IV): distributed shared memory

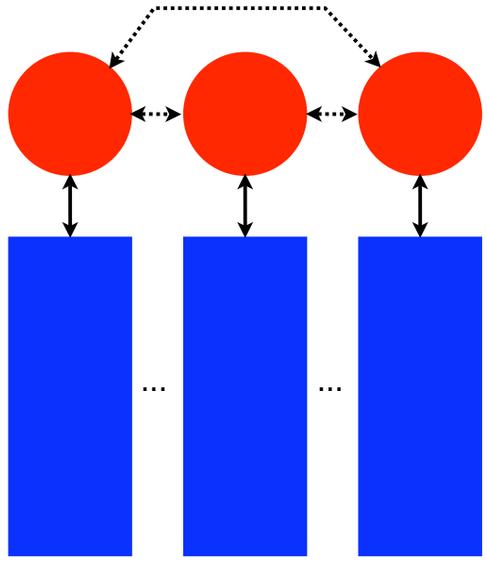
Also called partitioned global address space (PGAS) model

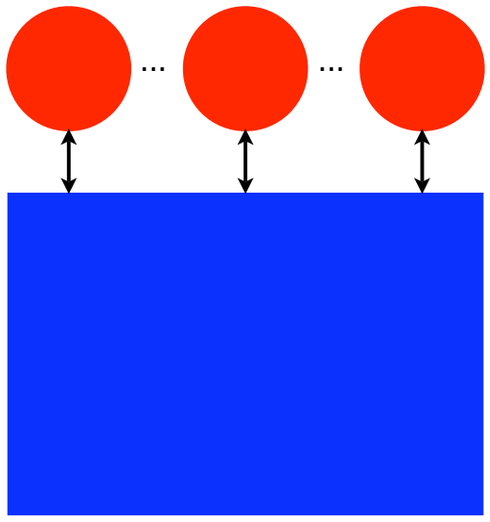


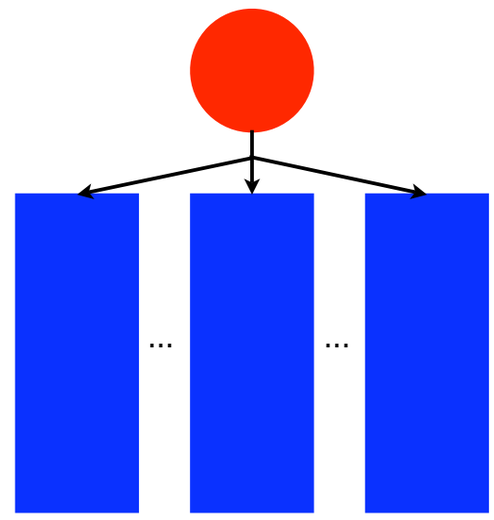


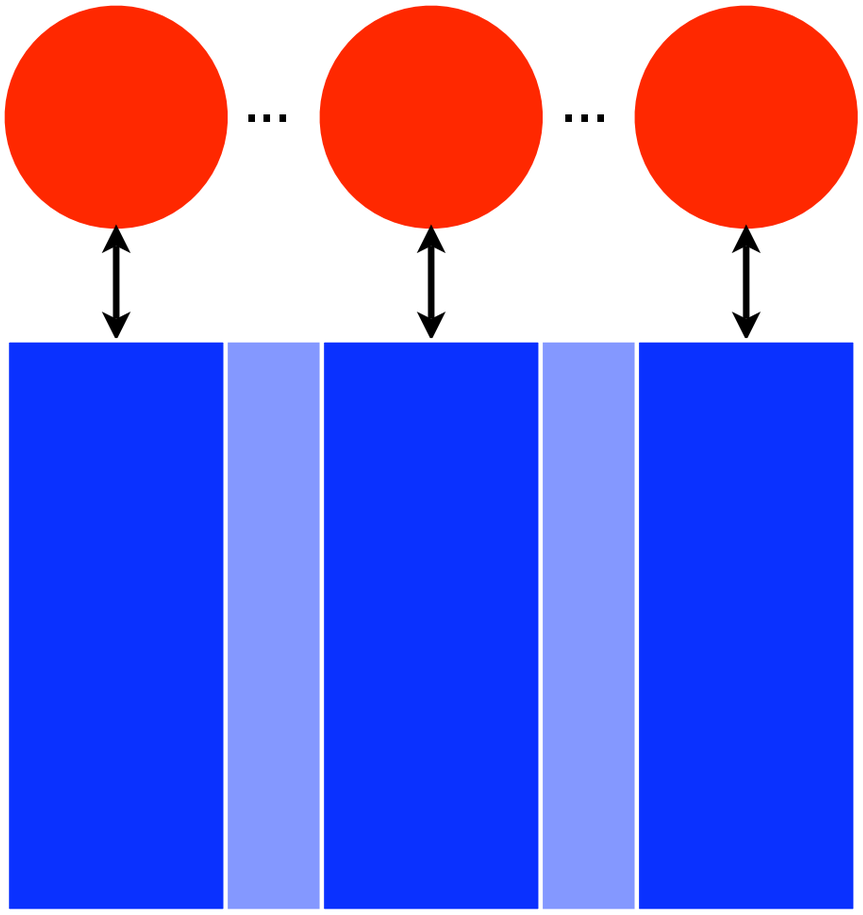
- Independed threads operate in shared memory space
  - preserve global view of program
- Shared space is locally partitioned among threads
  - allows exploiting data locality
- “Single program multiple data stream” (SPMD) execution
  - independent forking (functional parallelism)
- Popular examples: UPC and co-Array Fortran
- May still not have the same flexibility as Message Passing Model

Distributed shared memory or PGAS: keeping the best from all other models









Unified Parallel C (UPC)

see http://upc.lbl.gov

```
#include <upc.h>
#define SZ 20

main(){
  static shared int array[SZ];
  shared int *ar_ptr;
  static shared int step=10;
  int argument, i;

  ar_ptr = array + MYTHREAD;

  upc_forall (i=0; i<SZ; i++; i) {
    argument = step*i;
    *ar_ptr = some_function(argumet);
    ar_ptr += TREADS;
  }

  upc_barrier;

  if(MYTREAD==0) {
    ar_ptr = array;
    for (i=0; i<SZ; i++, ar_ptr++) {
      argument = step*i;
      printf ("%d \t %d \n", *ar_ptr, argument)
    }
  }
}
```

- Simple extension to ISO C to implement PGAS model in SPMD mode of programming
- Available on most systems (as extension to gcc or LBL UPC compiler)
- "Simpler" to program than MPI or OpenMP (!)
- Limitation: supports only one-dimensional decomposition of shared arrays
  - more complex decomposition are supported by DARPA HPCS languages (Chapel, X10, Fortress)

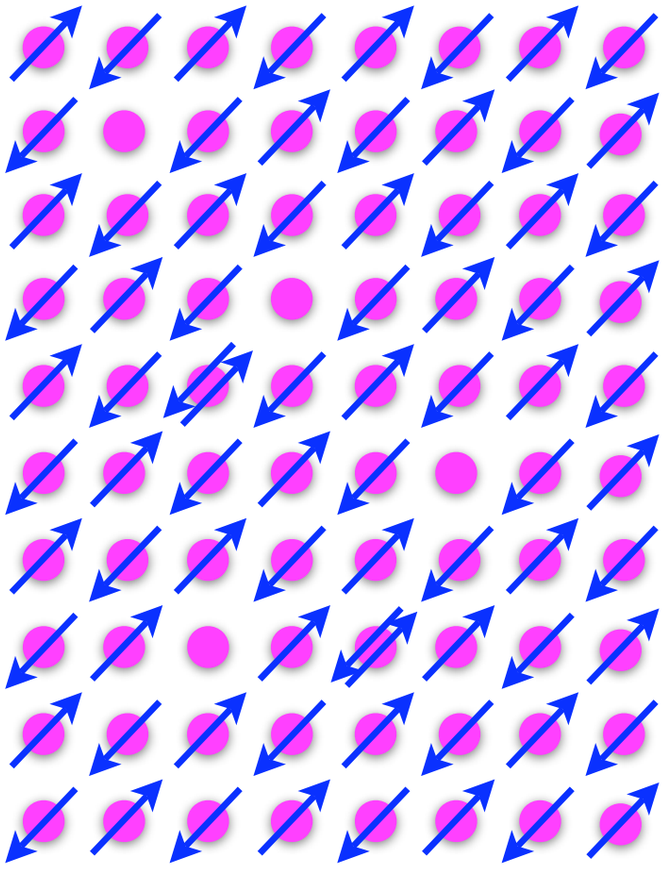
Example modified from Tarek El-Ghazawi, Wiliam Carlson, Thomas Sterling, and Katherine Yelick, "UPC Distributed Shared Memory Programming", WILEY 2005

Outline / goals

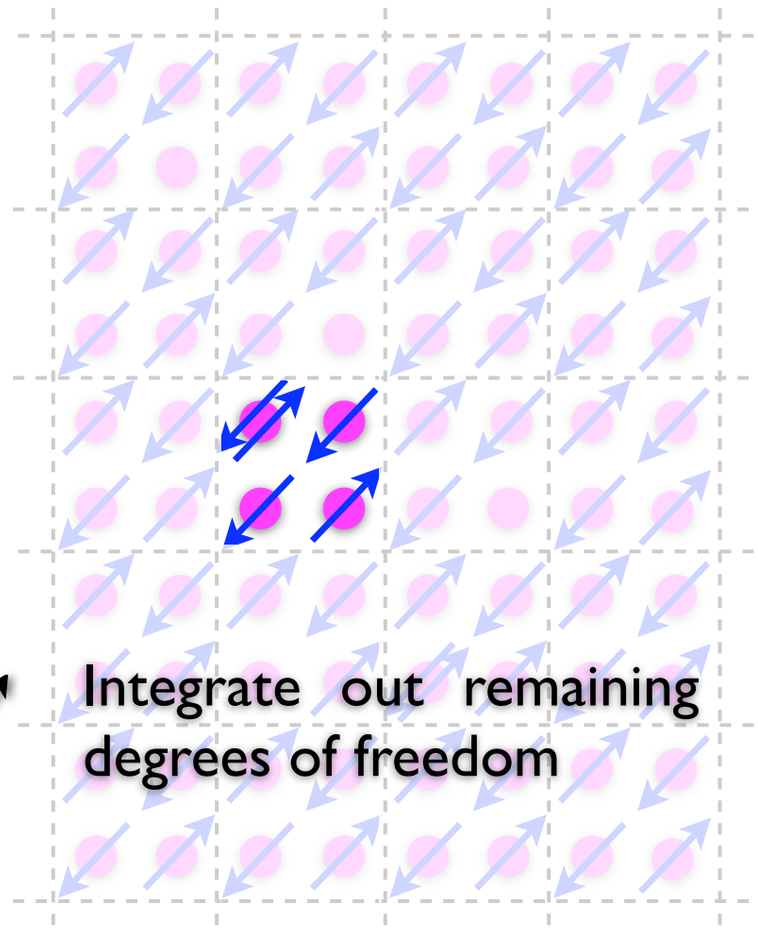
- History of scientific computing and HPC
- Overivew of typical HPC architectures
  - Important terminology used later
- Longterm development trends in HPC
  - Understand the brutal facts of HPC today
- Parallel programming models
  - There is more to parallel programming than MPI and OpenMP
- Mapping methods/algorithm onto hardware
  - Importance of computational methematics and computer science
- Example: running QMC/DCA at scale
  - Even simple things can be challenging on 150,000 processors

Sketch of the Dynamical Cluster Approximation

Bulk lattice

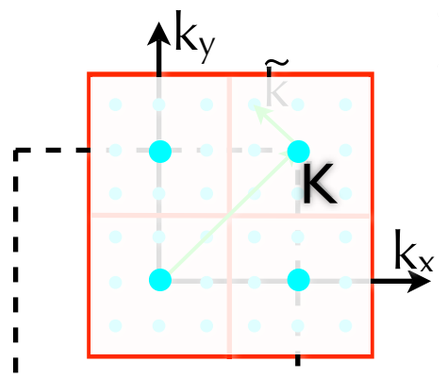


Size N_c clusters



Integrate out remaining degrees of freedom

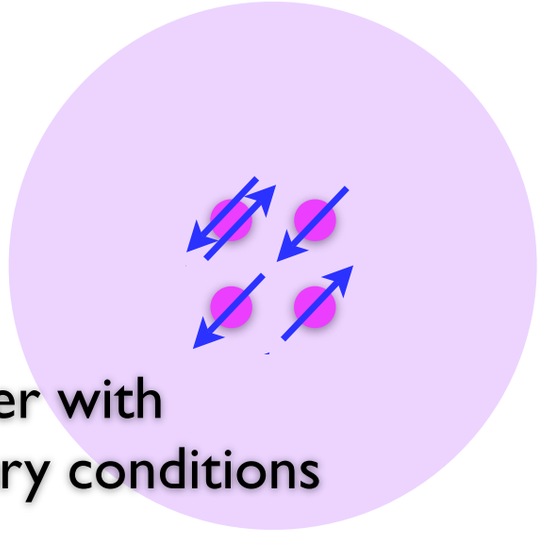
Reciprocal space



$\Sigma(z, k)$

$\Sigma(z, K)$

DCA

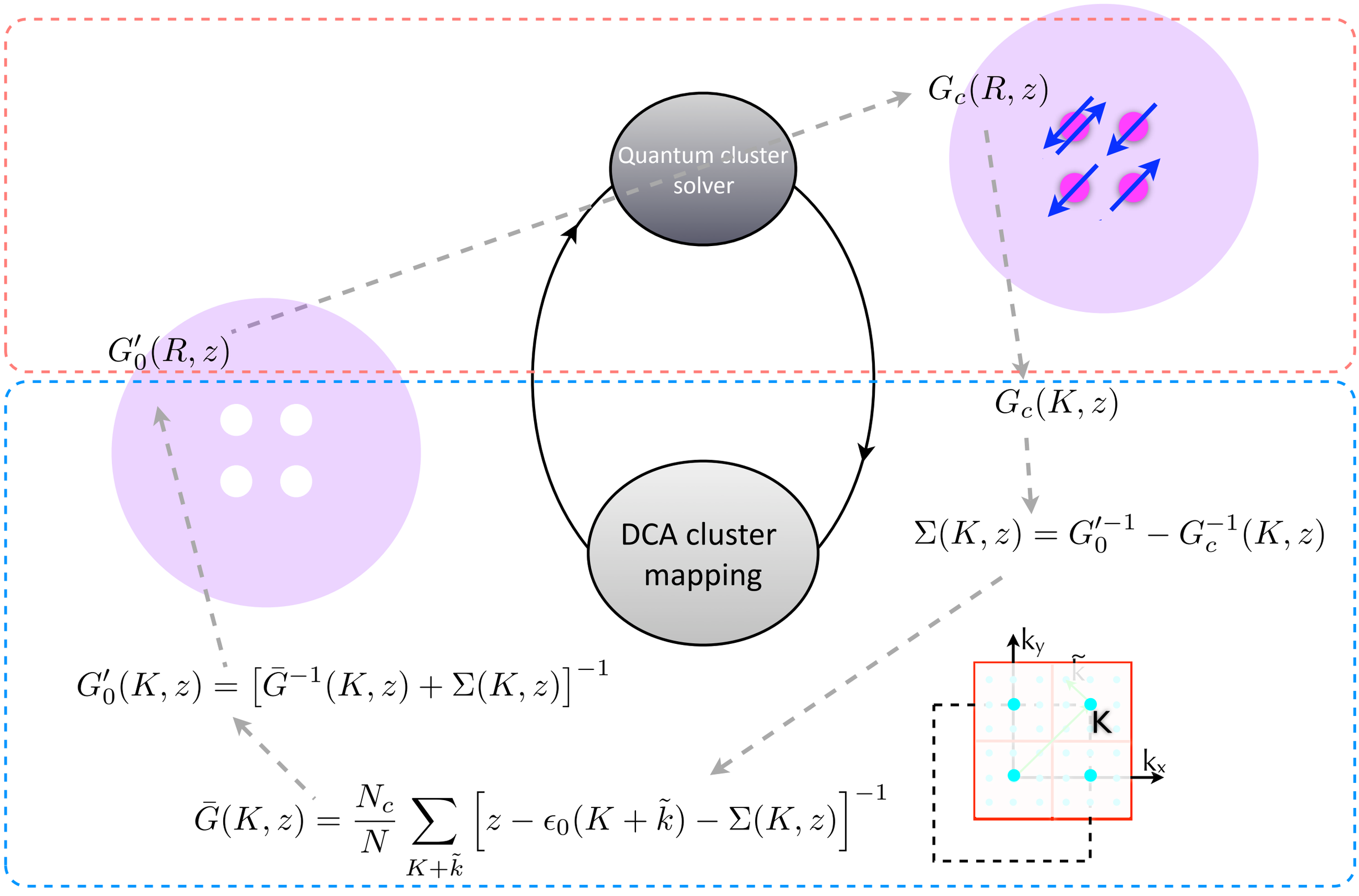


Embedded cluster with periodic boundary conditions

Solve many-body problem on cluster

➢Essential assumption: Correlations are short ranged

DCA method: self-consistently determine the "effective" medium



Hirsch-Fye Quantum Monte Carole (HF-QMC) for the quantum cluster solver

Hirsch & Fye, Phys. Rev. Lett. **56**, 2521 (1998)

Partition function & Metropolis Monte Carlo $Z = \int e^{-E[\mathbf{x}]/k_{\mathrm{B}}T} d\mathbf{x}$

Acceptance criterion for M-MC move: $\min\{1, e^{E[\mathbf{x}_k]-E[\mathbf{x}_{k+1}]}\}$

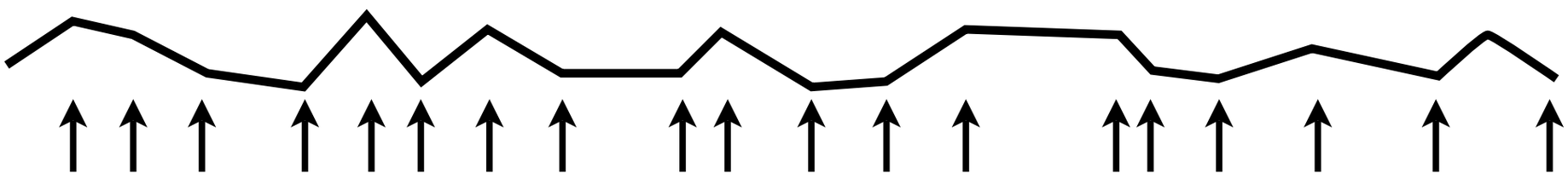
Partition function & HF-QMC: $Z \sim \sum_{s_i,l} \det[\mathbf{G}_c(s_i,l)^{-1}]$

N_c $N_l \approx 10^2$

matrix of dimensions $N_t \times N_t$

$N_t = N_c \times N_l \approx 2000$

Acceptance: $\min\{1, \det[\mathbf{G}_c(\{s_i,l\}_k)]/\det[\mathbf{G}_c(\{s_i,l\}_{k+1})]\}$

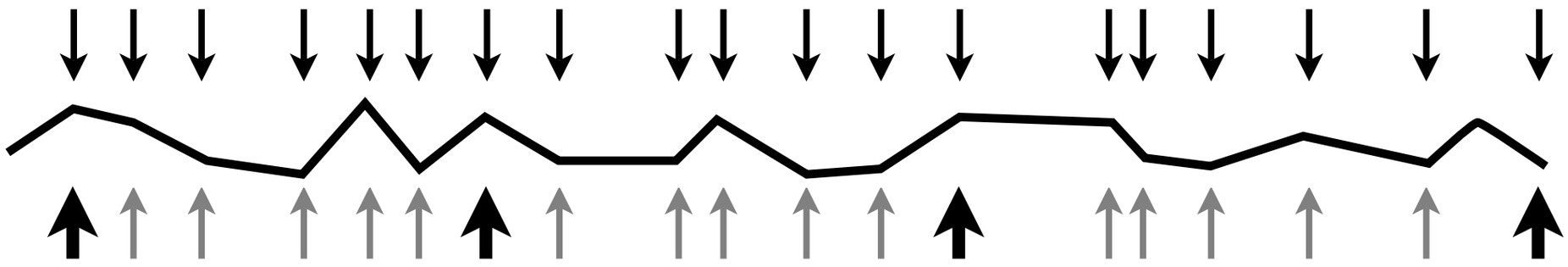


Update of accepted Green's function:

$$\mathbf{G}_c(\{s_i,l\}_{k+1}) = \mathbf{G}_c(\{s_i,l\}_k) + \mathbf{a}_k \times \mathbf{b}_k$$

HF-QMC with delayed updates (or Ed updates)

$$\mathbf{G}_c(\{s_i,l\}_{k+1}) = \mathbf{G}_c(\{s_i,l\}_k) + \mathbf{a}_k \times \mathbf{b}_k^t$$



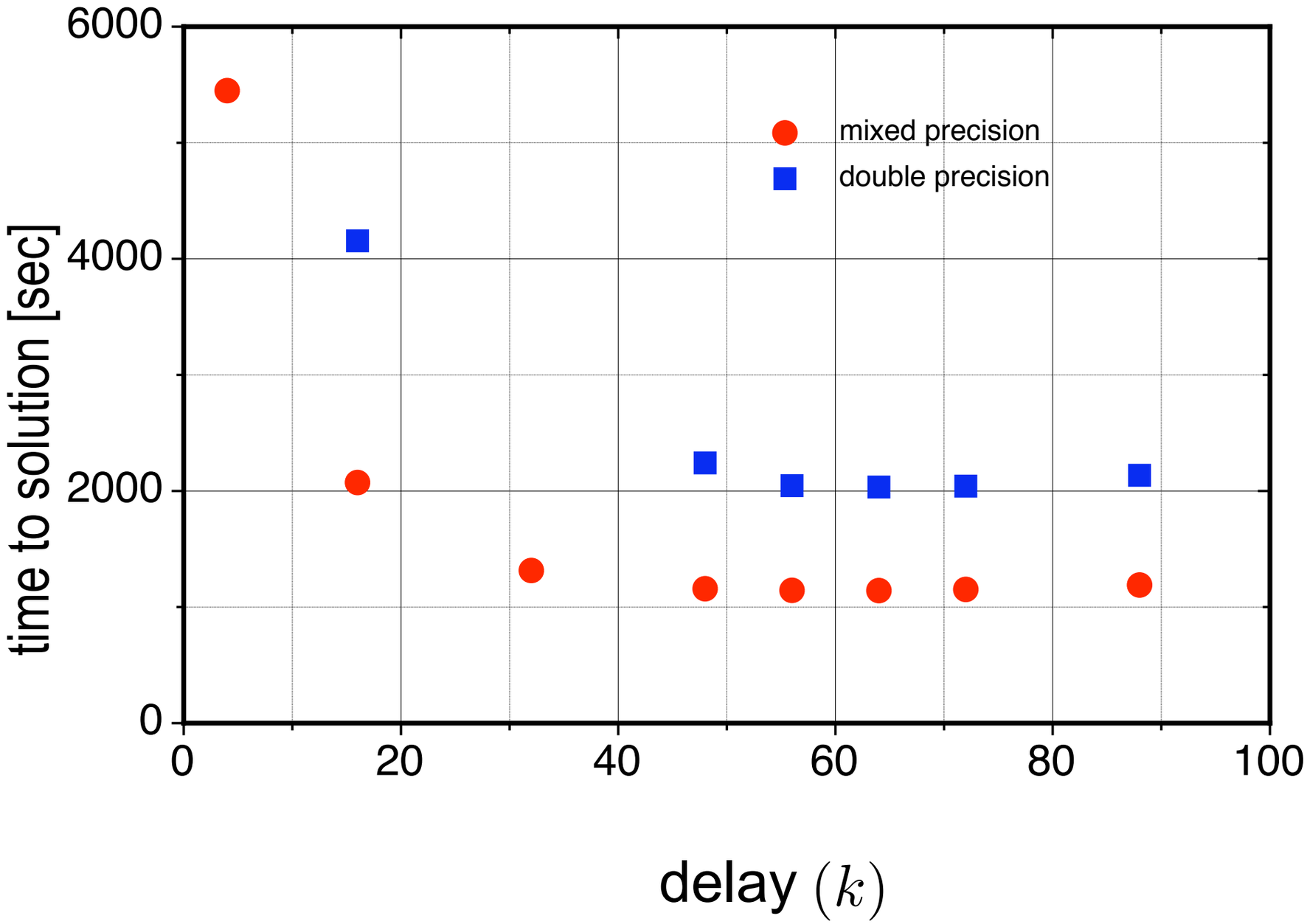
$$\mathbf{G}_c(\{s_i,l\}_{k+1}) = \mathbf{G}_c(\{s_i,l\}_0) + [\mathbf{a}_0|\mathbf{a}_1|...|\mathbf{a}_k] \times [\mathbf{b}_0|\mathbf{b}_1|...|\mathbf{b}_k]^t$$

Complexity for *k* updates remains $\mathcal{O}(kN_t^2)$

But we can replace *k* rank-1 updates with one matrix-matrix multiply plus some additional bookkeeping.

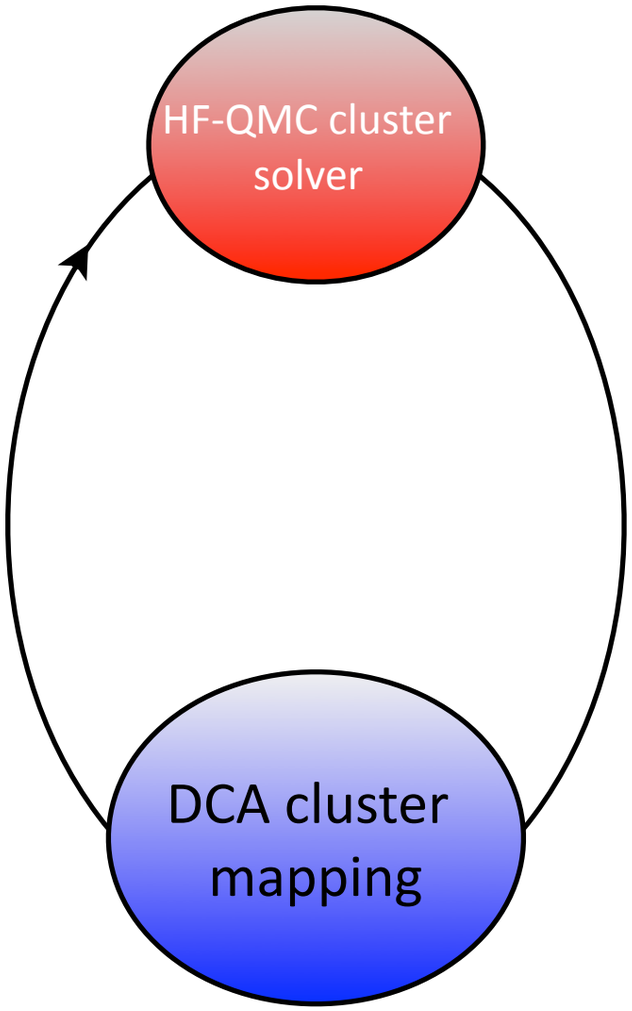
Performance improvement with delayed updates

$N_c = 16 \qquad N_l = 150 \qquad N_t = 2400$

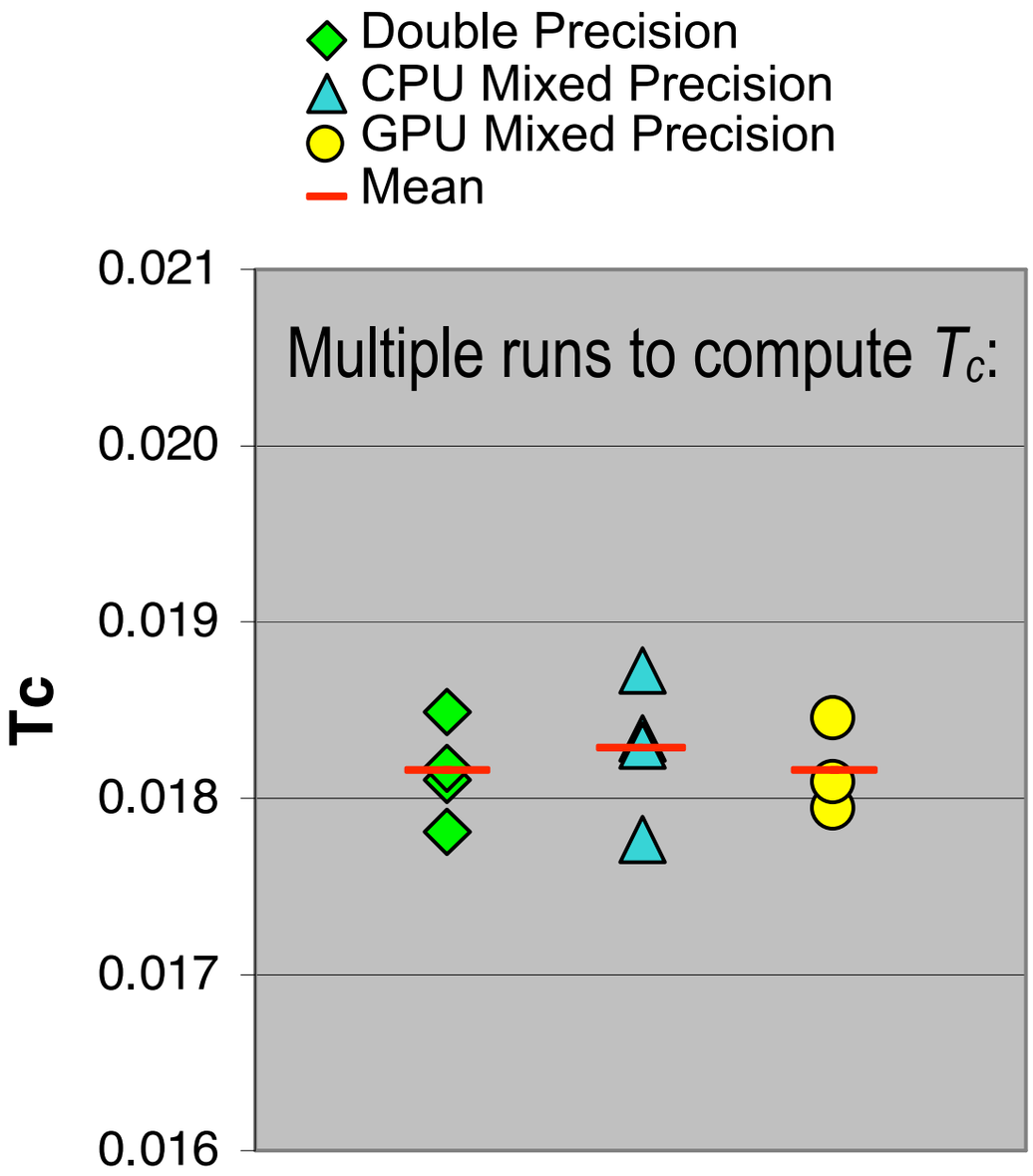


DCA++ with mixed precision

Run HF-QMC in single precision



Results for mixed and double precision runs are identical for same random number sequence!



Keep the rest of the code, in particular cluster mapping in double precision

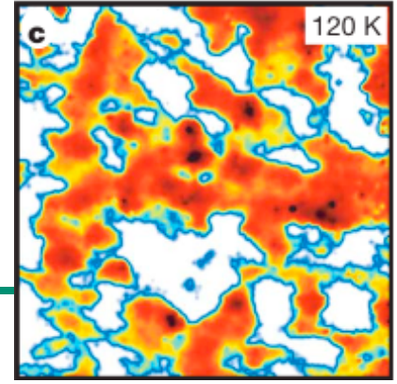
Speedup of HF-QMC updates (2GHz Opteron vs. NVIDIA 8800GTS GPU):

- 9x for offloading BLAS to GPU & transferring all data
- 13x for offloading BLAS to GPU & lazy data transfer
- 19x for full offload HF-updates & full lazy data transfer

Outline / goals

- History of scientific computing and HPC
- Overivew of typical HPC architectures
  - Important terminology used later
- Longterm development trends in HPC
  - Understand the brutal facts of HPC today
- Parallel programming models
  - There is more to parallel programming than MPI and OpenMP
- Mapping methods/algorithm onto hardware
  - Importance of computational methematics and computer science
- Example: running QMC/DCA at scale
  - Even simple things can be challenging on 150,000 processors

Disorder and inhomogeneities



Hubbard Model with random disorder (eg. in U)

$$H^{(\nu)} = -t \sum_{\langle ij \rangle, \sigma} c^{\dagger}_{i\sigma} c_{j\sigma} + \sum_{i} U^{(\nu)}_{i} n_{i\uparrow} n_{i\downarrow}$$

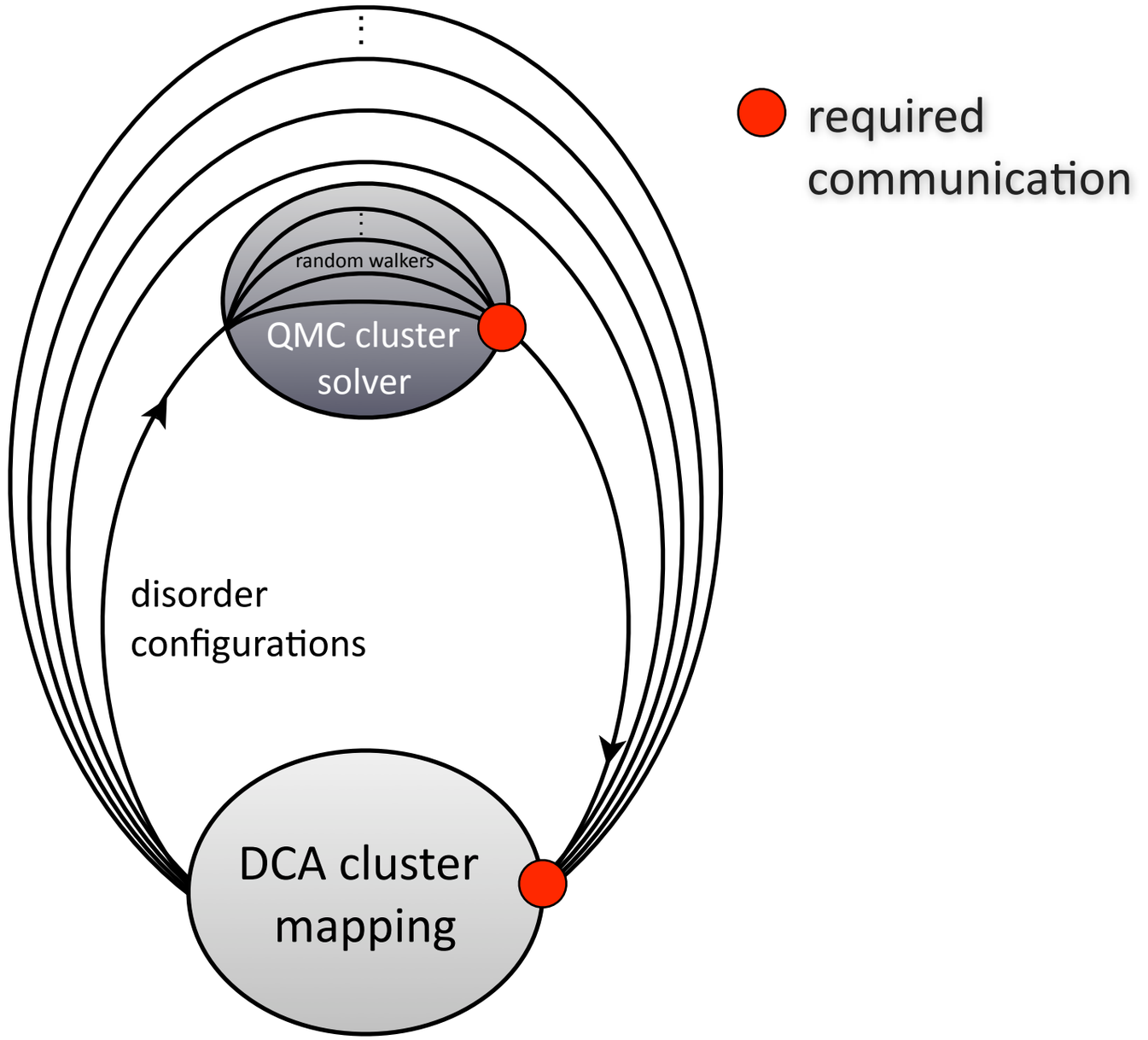
$$U^{(\nu)}_{i} \in \{U, 0\}; N_c = 16 \rightarrow N_d = 2^{16}$$

... need to disorder-average cluster Green function

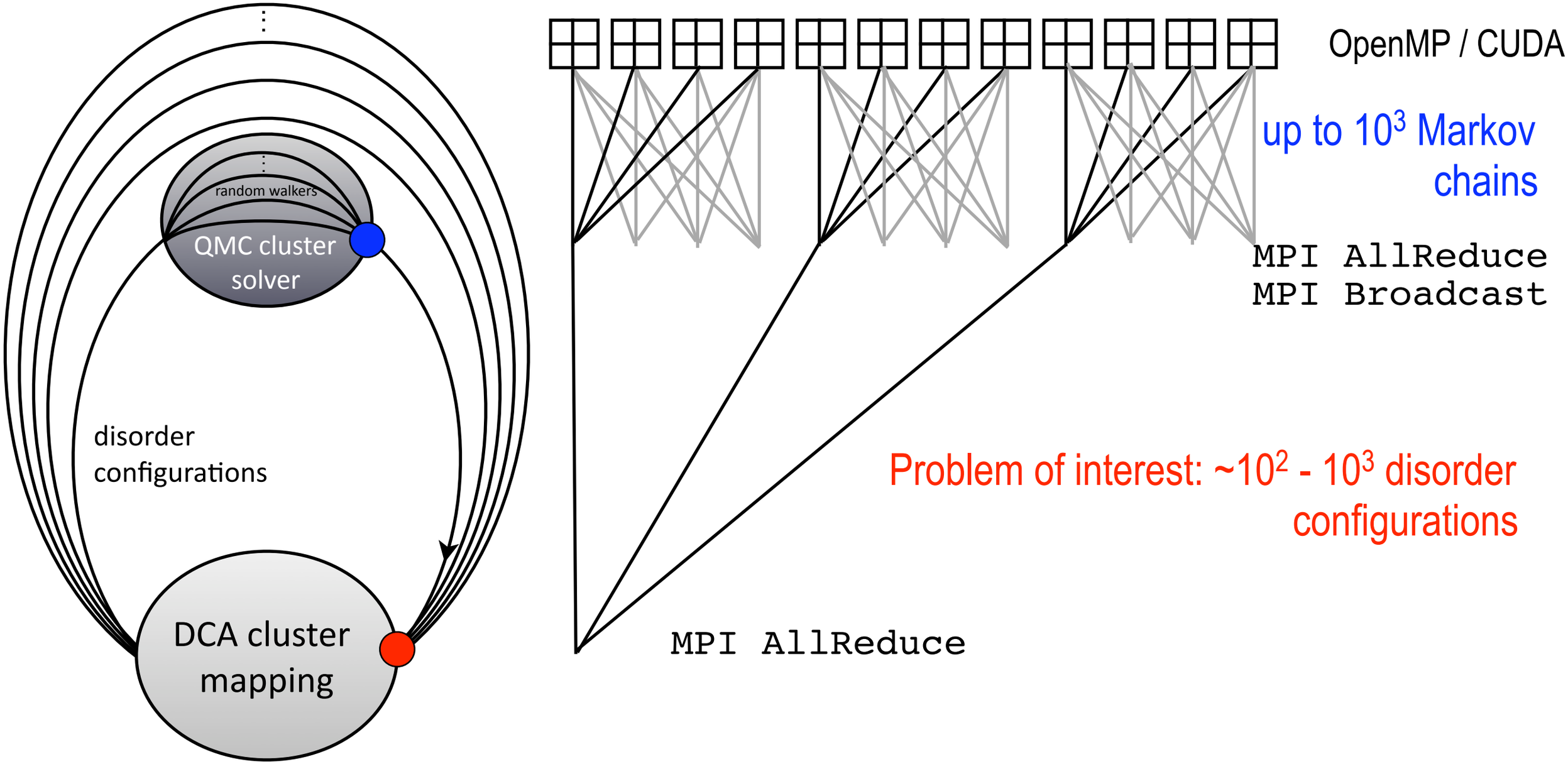
$$G_c(X_i - X_j, z) = \frac{1}{N_c} \sum_{\nu=1}^{N_d} G^{\nu}_c(X_i, X_j, z)$$

Algorithm 1 DCA/QMC Algorithm with QMC cluster solver (lines 5-10), disorder averaging (lines 4, 11-12), and DCA cluster mapping (line 3, 13)

```
1:  Set initial self-energy
2:  repeat
3:     Compute the coarse-grained Green Function
4:     for Every disorder configuration (in parallel) do
5:        Perform warm-up steps
6:        for Every Markov chain (in parallel) do
7:           Update auxiliary fields
8:           Measure Green Function and observables
9:        end for
10:       Accumulate measurements over Markov chains
11:    end for
12:    Accumulate measurements over disorder configurations.
13:    Re-compute the self-energy
14: until  self consistency is reached
```

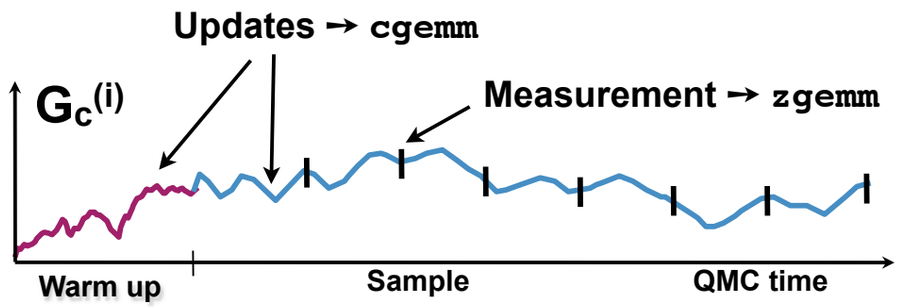


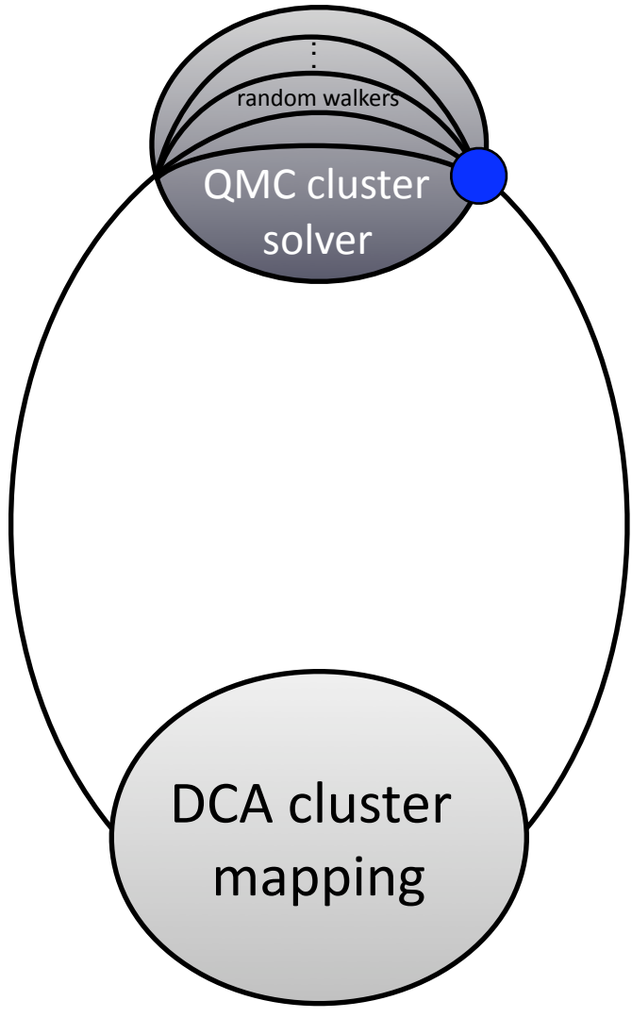
DCA++ code from a concurrency point of view

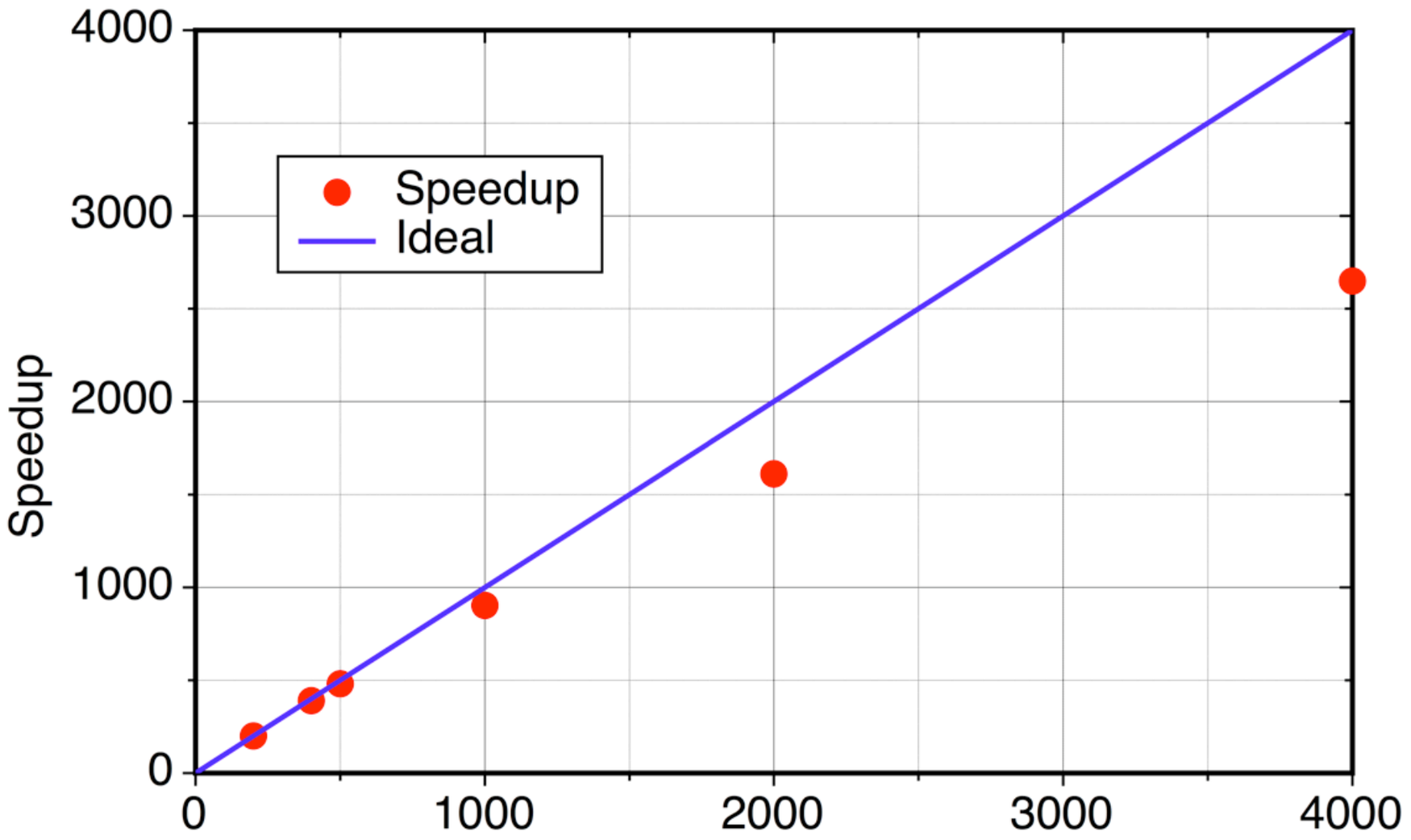


Problem of interest: ~10^2 - 10^3 disorder configurations

DCA++: strong scaling on HF-QMC

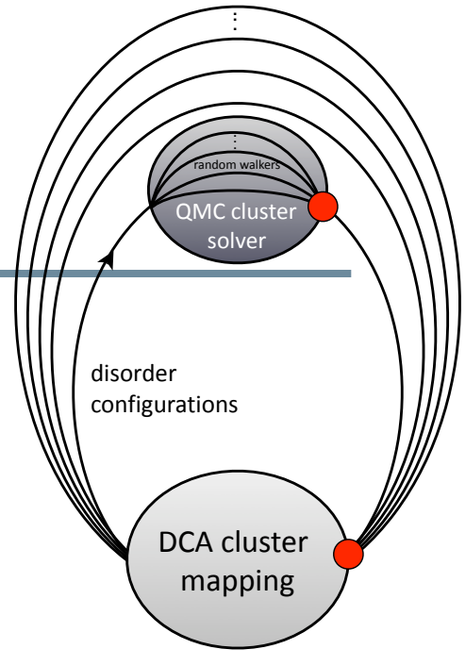


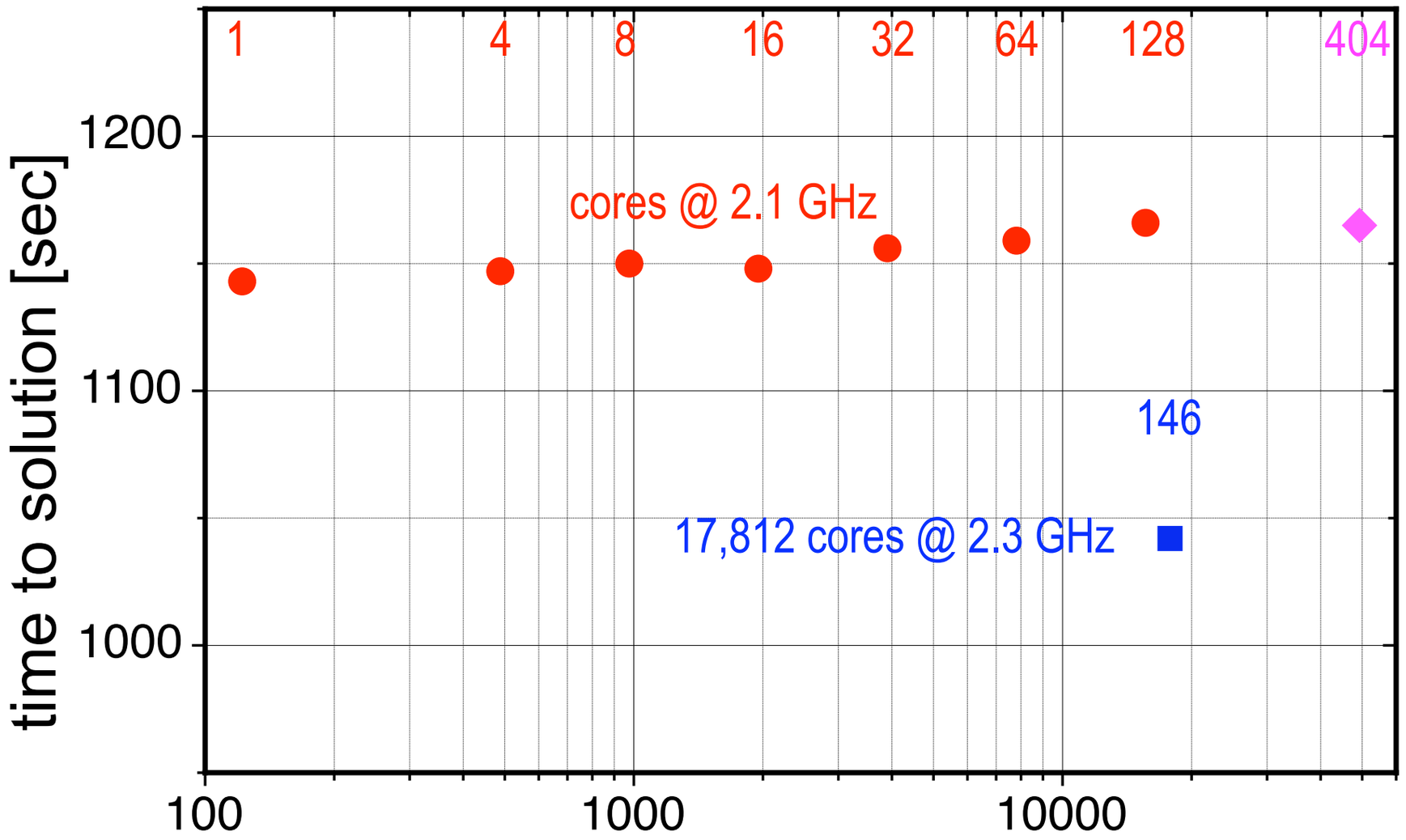




Weak scaling on Cray XT4

- HF-QMC: 122 Markov chains on 122 cores
- Weak scaling over disorder configurations





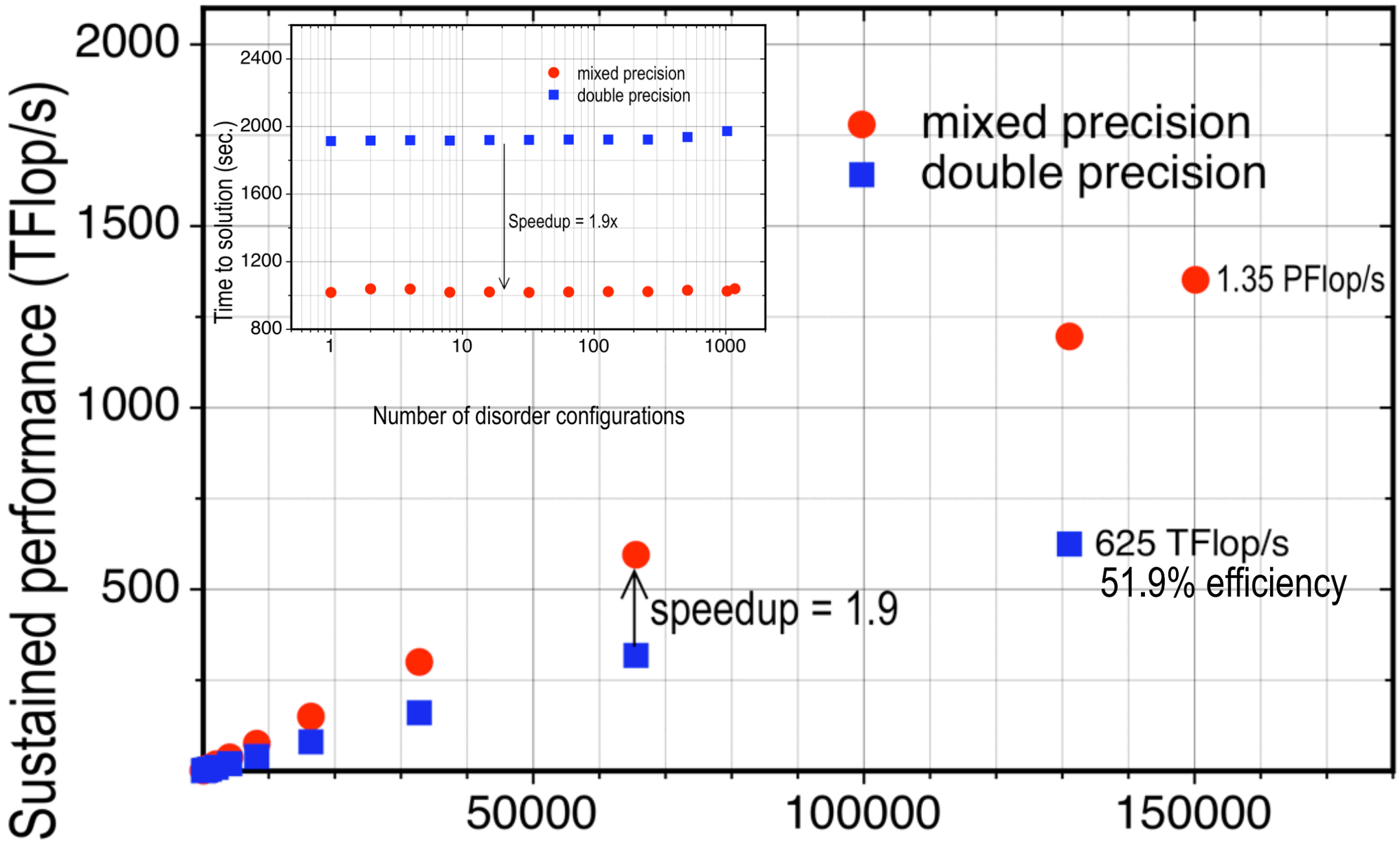
31,232 cores @ 2.1 GHz +
17,812 cores @ 2.3 GHz =
49,044-core chimera

Cray XT5 portion of Jaguar @ NCCS

Peak: 1.382 TF/s
Quad-Core AMD
Freq.: 2.3 GHz
150,176 cores
Memory: 300 TB
For more details, go to
www.nccs.gov

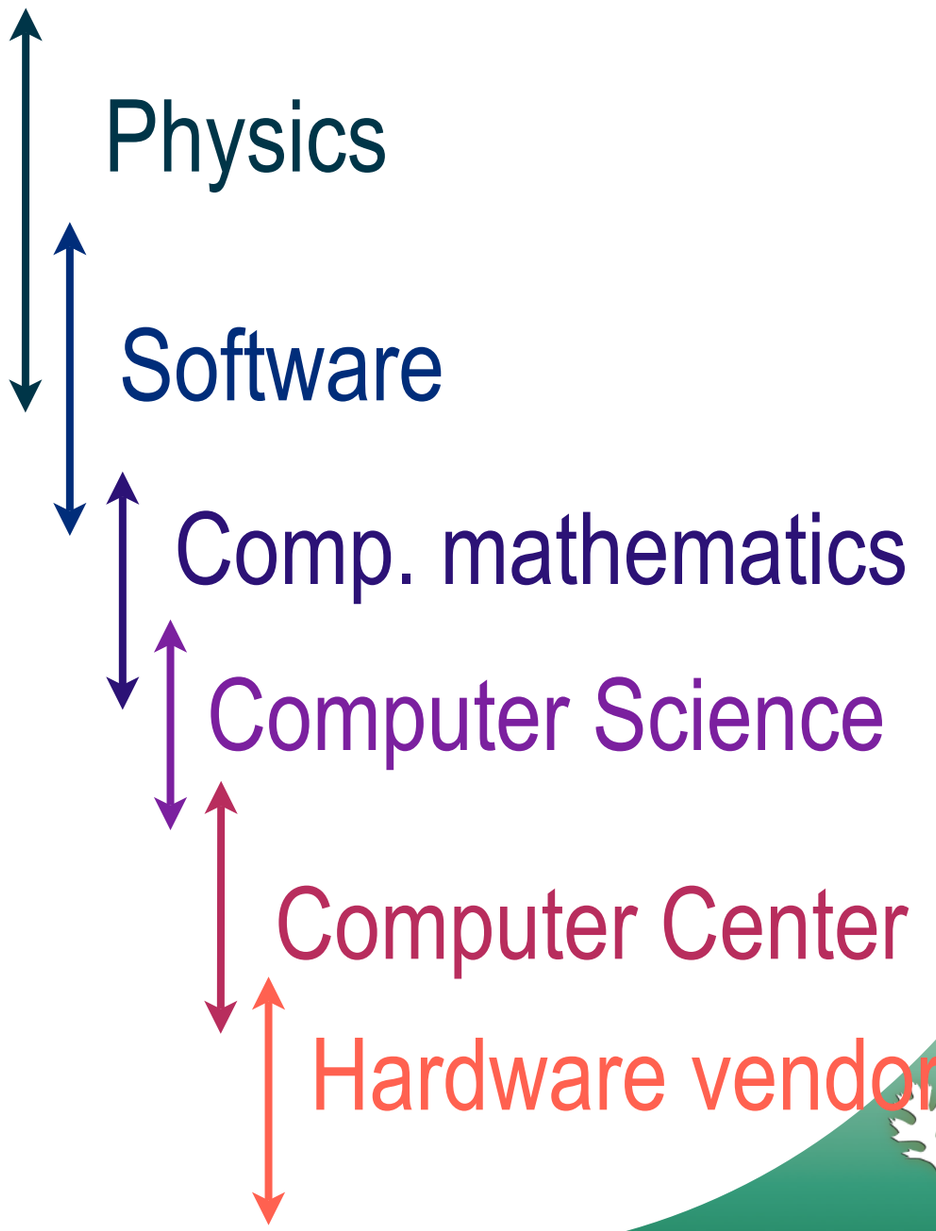
Sustained performance of DCA++ on Cray XT5

Weak scaling with number disorder configurations, each running on 128 Markov chains on 128 cores (16 nodes) - 16 site cluster and 150 time slides



New algorithm to enable 1+ PFlop/s sustained performance in simulations of disorder effects in high-T_c superconductors

T. A. Maier
P. R. C. Kent
T. C. Schulthess
G. Alvarez
M. S. Summers
E. F. D'Azevedo
J. S. Meredith
M. Eisenbach
D. E. Maxwell
J. M. Larkin
J. Levesque



OAK RIDGE National Laboratory