Big Data - Ops vs Flops

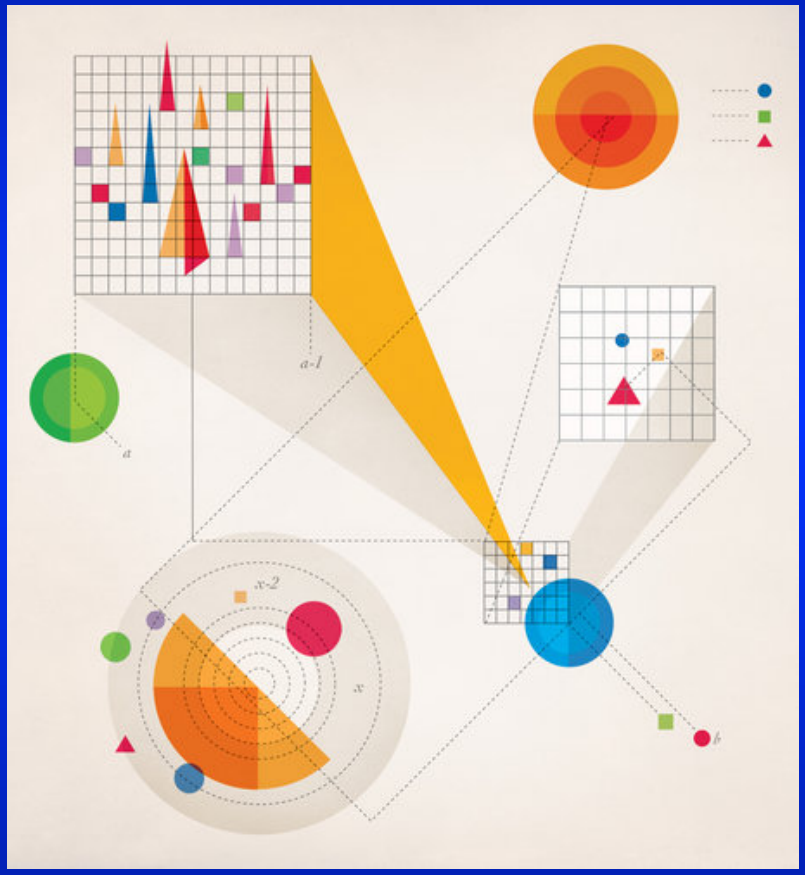
Steve Wallach
swallach”at”conveycomputer “dot”com
Convey <= Convex++

Why is this interesting?

- Big Data is on everyone's mind. It is in the NEWS.
  - HPC Classic (Flops) & Big Data (Ops)
- Power efficiency is on everyone's mind. It is in the NEWS.
- Are today's processor architectures a match for Exascale applications that are in the NEWS?
  - Ops vs Flops
    - Deep dive into some micro-architecture discussions
- A smarter computer for Exascale. It is in the NEWS

From a NEWS Perspective

- World Economic Forum
  - declared data a new class of economic asset, like currency or gold.
- growing at 50 percent a year, or more than doubling every two years, estimates IDC
- "data-driven decision making" achieved productivity gains that were 5 percent to 6 percent higher than other factors could explain.
- "false discoveries." "many bits of straw look like needles."



February, 11, 2012, NY Times, "The Age of Big Data", STEVE LOHR

CONVEY computer

Support from Washington DC

OBAMA ADMINISTRATION UNVEILS "BIG DATA" INITIATIVE: ANNOUNCES $200 MILLION IN NEW R&D INVESTMENTS

Office of Science and Technology Policy Executive Office of the President

New Executive Office Building

Washington, DC 20502

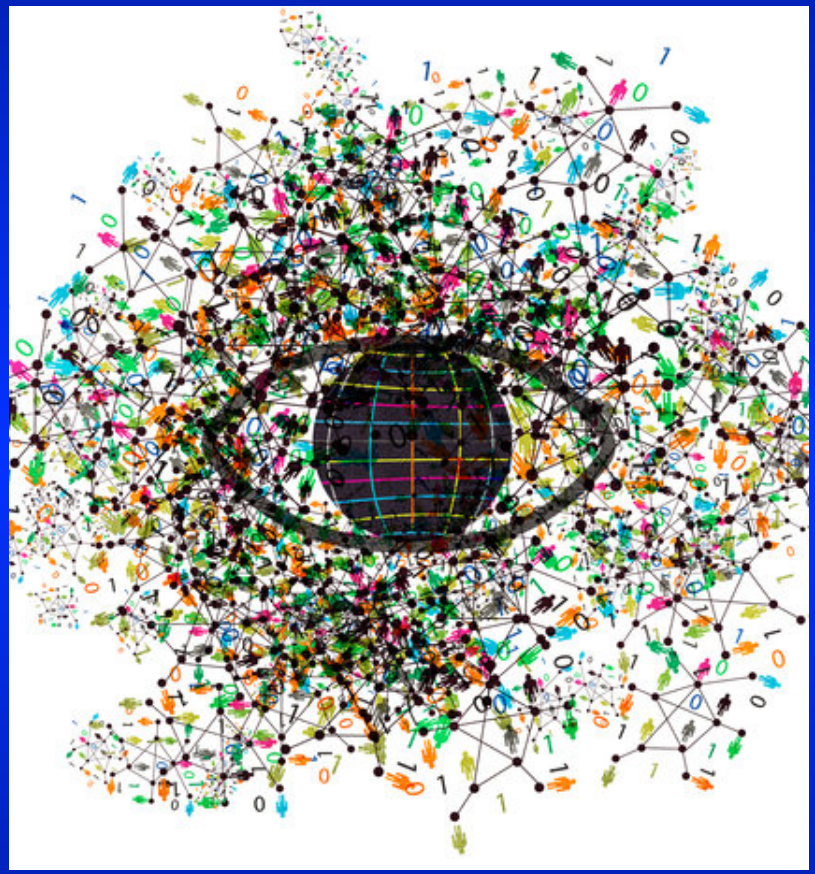
March 29, 2012

CONVEY computer

Perhaps another reason

Seriously

- "Computing may be on the cusp of another such wave."
- "The wave will be based on smarter machines and software that will automate more tasks and help people make better decisions."
- "The technological building blocks, both hardware and software, are falling into place, stirring optimism."



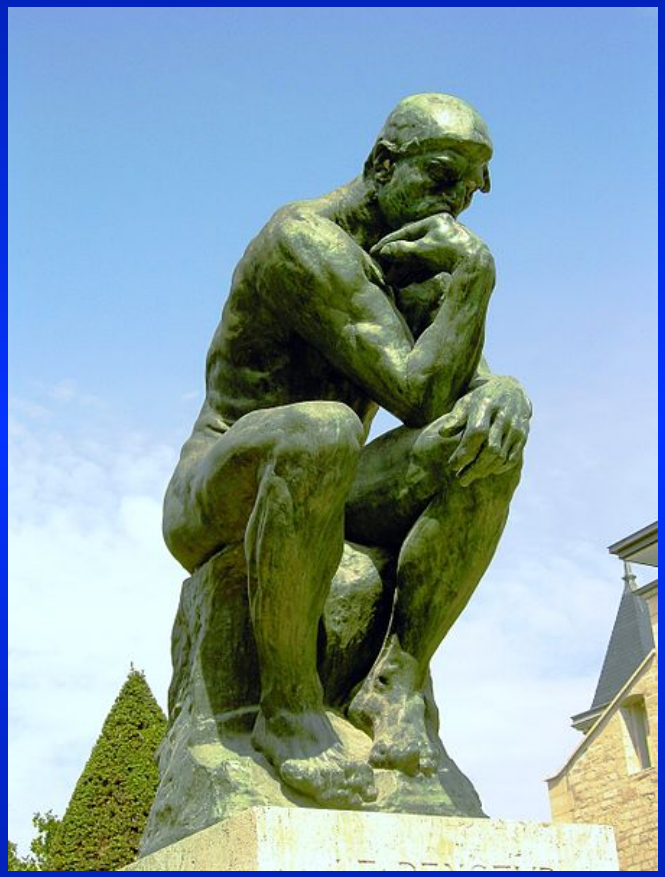
September 8, 2012, NY Times, "Tech's New Wave, Driven by Data"

STEVE LOHR

CONVEY computer

Thinking on Technology

- Flops and Ops are applicable to Exascale
- More items in common then initially obvious
- Examine in more detail
- The role of Time Sharing
- The role of cloud computing



Compare Flops with Ops

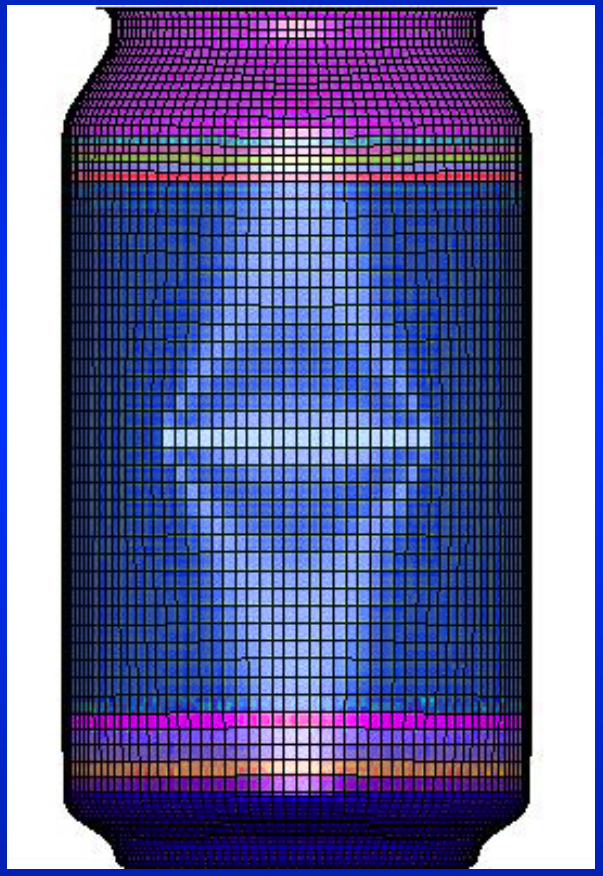
- ExaFlop (FP Intensive)
  - Historical Metrics/Features
    - Memory Capacity
      - Byte/Flop
    - Interconnect Bandwidth
      - .1 Bytes/Flop
    - Memory Bandwidth
      - 4 Bytes/sec/Flop
    - Linpack
  - Relatively static data set size
    - 3D … X,Y,Z - multiple points per cell
    - WaveFront Algorithms
  - Node level synchronization
    - Domain Decomposition
  - 64 bit memory references
    - 32 and 64 bit float
  - Cache Friendly (In general?)

- ExaOp (Data Intensive)
  - Transactions per sec
  - Transaction Latency
  - Graph edges per sec
  - YES/NO
  - Data Sets are dynamic
    - New data always being added
  - Never enough physical memory

    Memory/compute algorithms
    - Limited disk access
      - Flash memory/Memristor/Phase Change?
      - Memcached
  - Graphs, Trees, Bit and Byte strings
    - Fine grain synchronization
      - Threads/Transactions
    - Granularity of memory references
    - User defined strings
    - NORA (Non-Obvious Relationship Analysis)
  - Cache Unfriendly

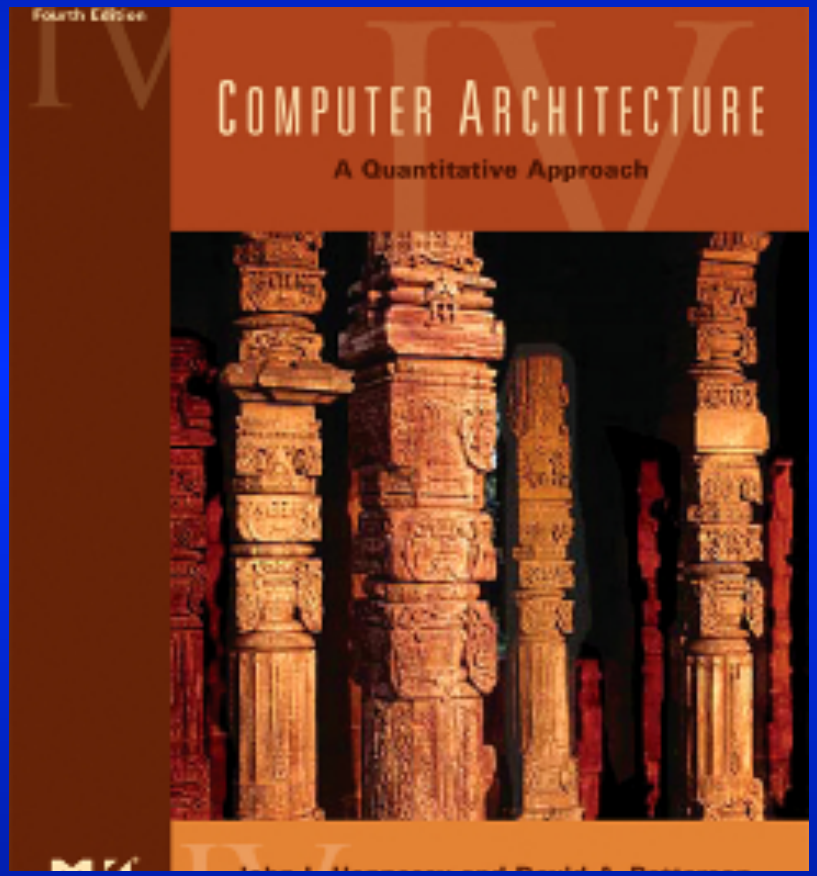
Solving Flops – HPC Classic

- Multi (Many) Core
- Multiple FMAC's
  - Floating Point
- Vector Accumulators
- Structured Data
- Attached accelerators
  - GPU, FPGA, …
- High Speed Interconnects
  - MPI Pings
- Open source numerical libraries
  - (e.g., LAPACK)
- Memory system
  - Classic – highly interleaved
  - Micro - multiple cache levels



Solving Ops
-An Architectural Perspective-

- Very Large Physical Memory
  - ExaOps ➔ ExaBytes
  - 64 bits of address is not enough
    - Run out by 2020
    - 1 to 1.5 bits increase per 1 to 1.5 years
- Single Level, Global Physical Memory
  - Simplifies Programming Model
  - Extensible over time
  - Updates in Real Time
  - Runtime binding
  - PGAS Languages
  - MAP REDUCE / HADOOP
- Multiple machine state models can exist
- Heterogeneous Processing
- Fine grain synchronization
  - Threads
  - Bit and Bytes
  - Adjacency Matrices



Now What

- Uniprocessor Performance has to be increased
  - Heterogeneous here to stay
  - The easiest to program will be the correct technology
- Smarter Memory Systems (PIM)
  - Synchronization
  - Complex Data Structures
  - Dram, NVRAM, Disk
- New HPC/Big Data Software must be developed.
  - SMARTER COMPILERS
  - ARCHITECTURALLY TRANSPARENT
  - Domain Specific Languages
- New algorithms
- New ways of referencing DATA at the processor level

Ops and Flops – Processor Memory System

- Caches create **in order** memory references for ALU's from out of order main memory references.
  - Cache block is in order
- Direct memory references tend to be out of order
  - Better for random references
  - Hundreds (thousands) of outstanding loads and stores
- Which memory reference approach is preferred?
- How does HMC (hybrid memory cube) change the approach?

Ops and Flops in one system?

```
Do I = 1,n
  A(i) = B(i) +C(i)
ENDDO
```

- For **in order** memory, a vector ISA is very efficient. For out of order, a thread model is more efficient (also hides latency better)
  - Threads execute independently but synchronized.
  - Supported by OpenMP

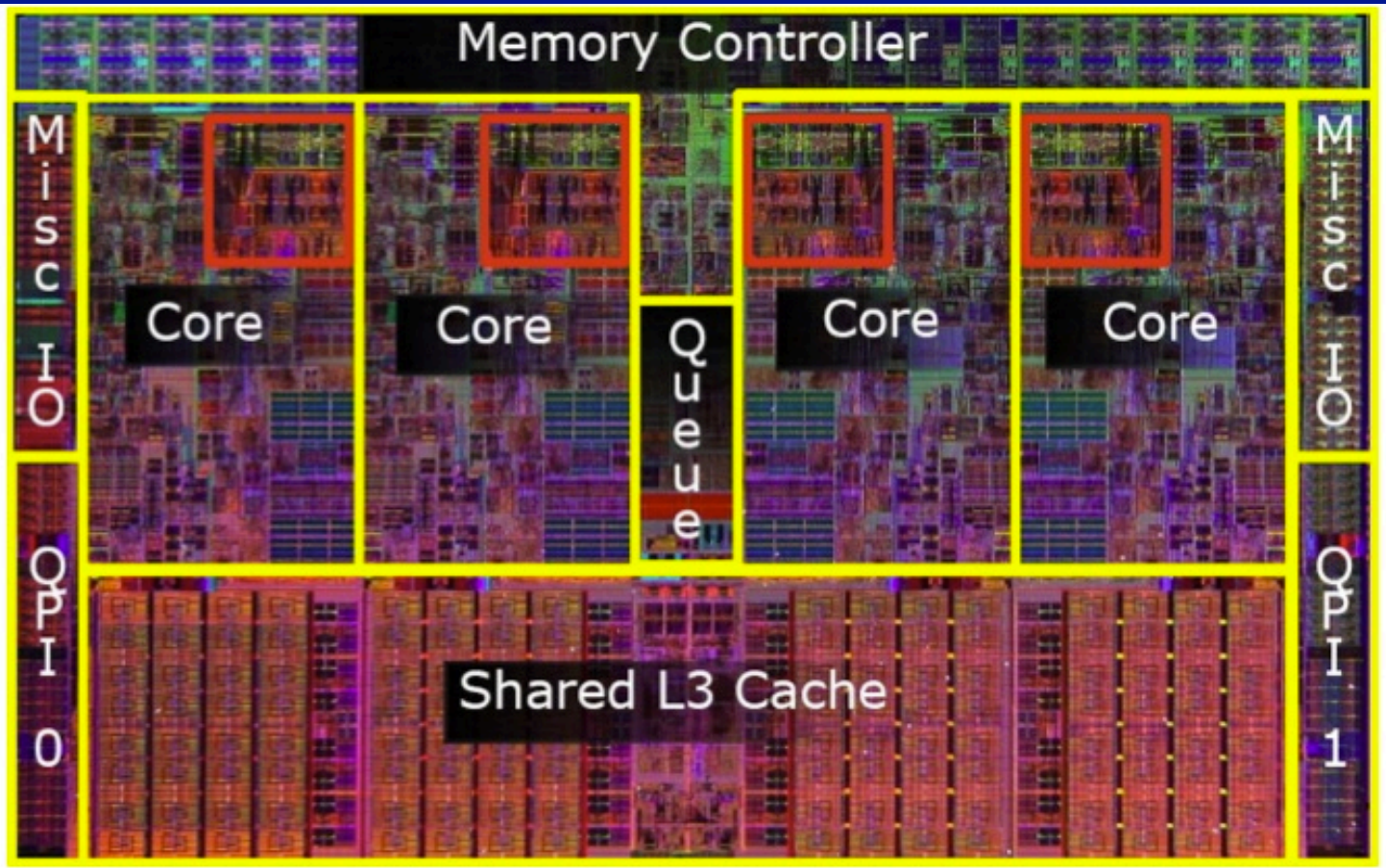


Figure 1. Intel's Core i7 processor (the chip code-named Bloomfield, based on the Nehalem microarchitecture) includes four CPU cores with simultaneous multithreading, 8MB of L3 cache, and on-chip DRAM controllers. Made with 45nm process technology, each chip has 731 million transistors and consumes up to 130W of thermal design power. Red outlines highlight the portion of each core occupied by execution units. (Source: Intel Corporation except red highlighting)

NVIDIA

Definition

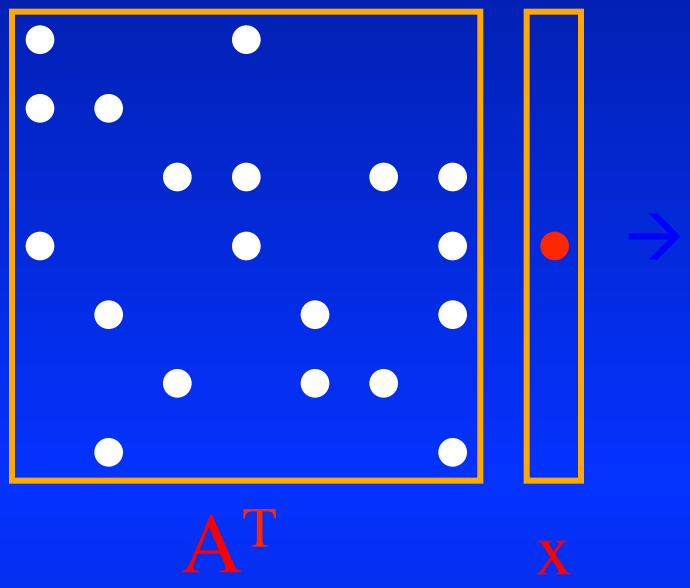
Boston, Hyannis, Martha's Vineyard, Nantucket, New Bedford, Providence, and Provincetown,

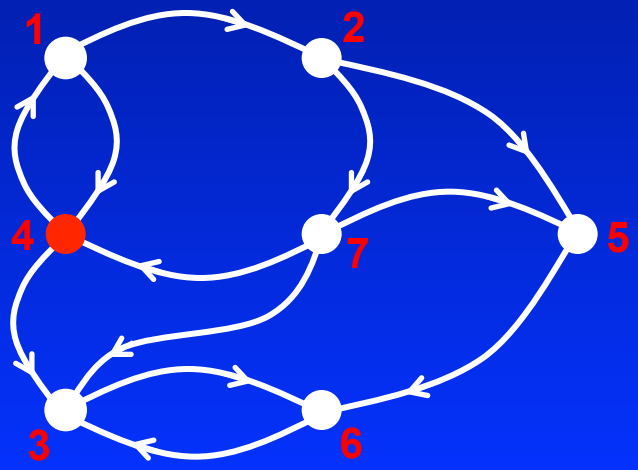
Figure 2: Northern Route Map for Cape Air – May

Adjacency Matrix

$$= \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

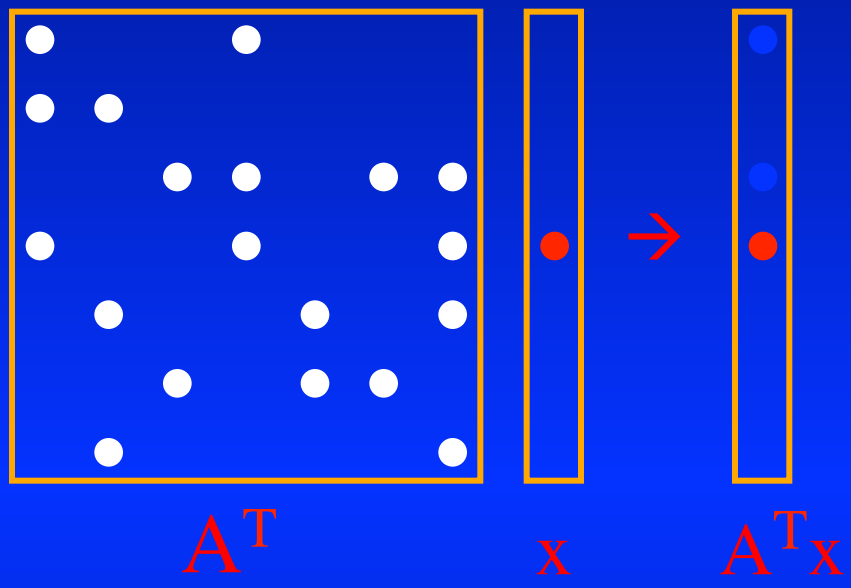
Breadth-First Search: Sparse mat * vec

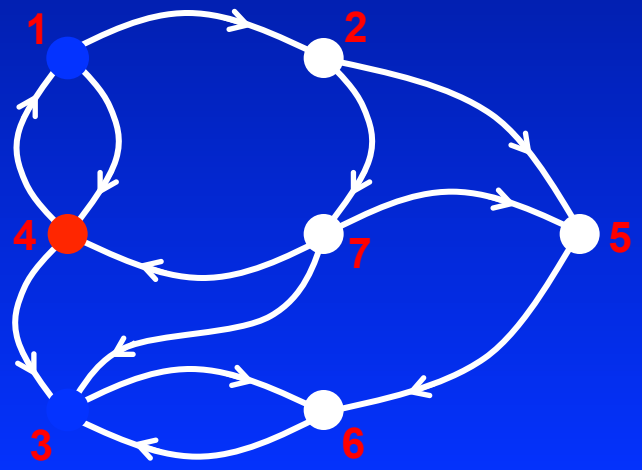




- Multiply by adjacency matrix → step to neighbor vertices
- Work-efficient implementation from sparse data structures
  - Ref: CS240a – UCSB – John Gilbert

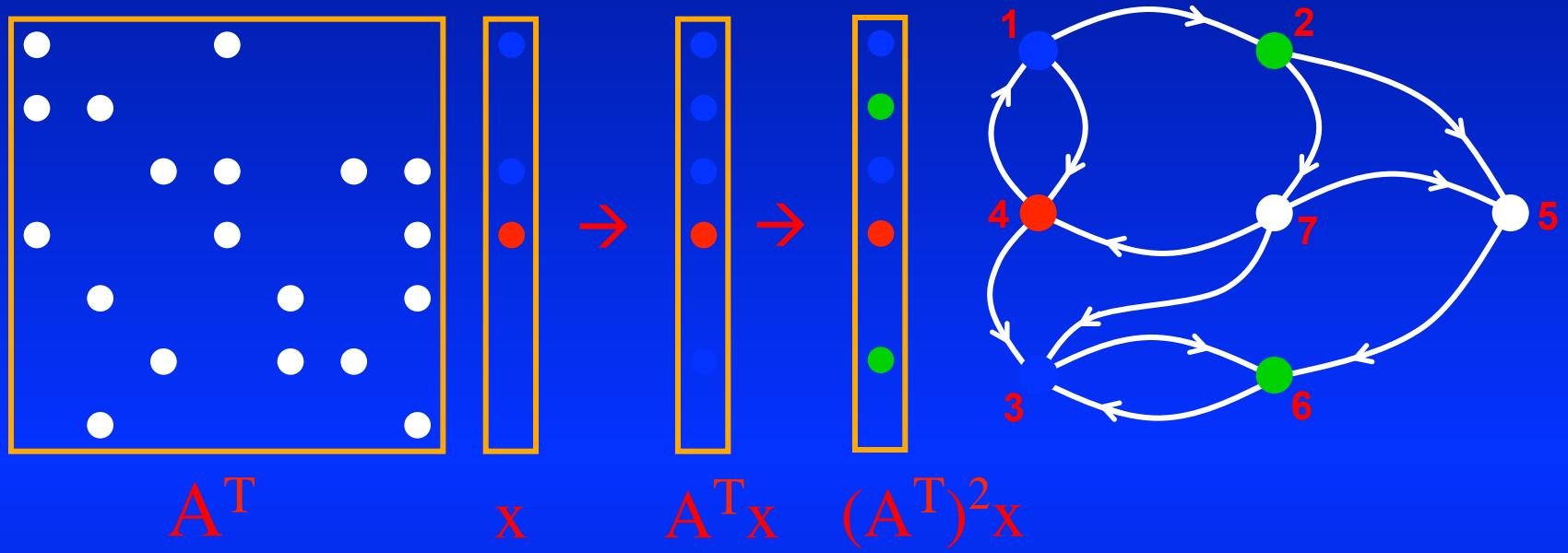
Breadth-First Search: Sparse mat * vec





- Multiply by adjacency matrix → step to neighbor vertices
- Work-efficient implementation from sparse data structures
  - Ref: CS240a – UCSB – John Gilbert

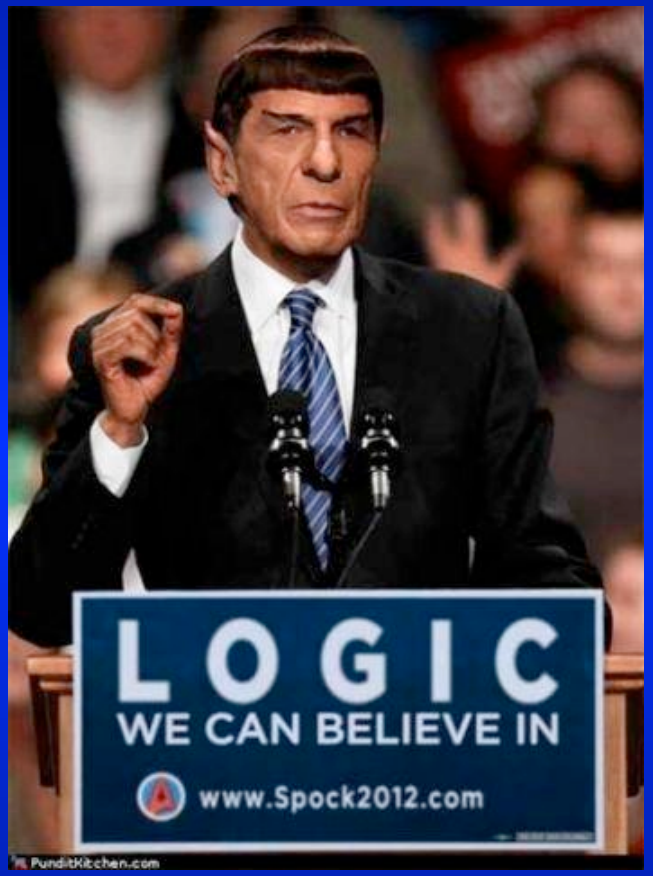
Breadth-First Search: Sparse mat * vec



- Multiply by adjacency matrix → step to neighbor vertices
- Work-efficient implementation from sparse data structures
  - Ref: CS240a – UCSB – John Gilbert
    - http://www.cs.ucsb.edu/~gilbert/cs240a/old/cs240aSpr2011/index.html

Problem Statement

- 4 Giga Names
  - Actual Names
  - IP address's
- 16k Meta Tags
- Discover Knowledge
- Adjacency Matrix
  - 32 Terabytes (if every bit is explicitly interpreted ("1" or "0"))



Problem Solution

- Need a way for dynamic selection of optimal solution
  - Threads
  - Bit Operations directly on Adjacency Matrix
- Just like sparse numerical solvers
  - Based on sparsity (dynamically chosen)
    - Operations Under Mask
    - Vector of Indices
    - Compress reduce/expand

Adjacency Matrices

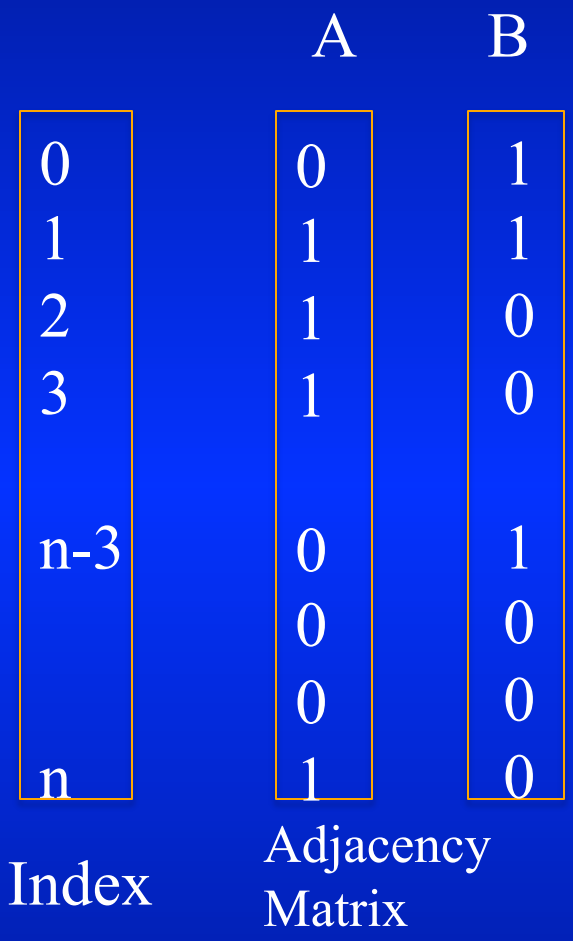
- Interesting properties in BIT domains
  - If compressed column, just having an index indicates a logical "1" or "0"
- Cardinality is trivial (POPCOUNT)
- Logical Operations are trivial
- Optimum Data Set structure is dynamic
  - http://www.ll.mit.edu/HPEC/agendas/proc10/Day2/S4_1335_Song_abstract.pdf
  - 3-D Graph Processor
  - William S. Song, Jeremy Kepner, et. aI., Lincoln Laboratory, Massachusetts Institute Technology, Lexington, MA 02420

Yellow Brick Road

ISA Definition

- Load/Store Bit streams
  - Indirect thru Dope vector of compressed columns or a dense bit stream
  - Load indices of columns (i)
    - If bit convert to index within processor
    - Block rows (like strip mining for parallelism)
  - Output indices that satisfy logical operation (general purpose N-adic)
  - PIM memory controllers interpret and create Dope Vectors
- Arithmetic's are Convolvers
  - N-Adic
  - Keep track of index

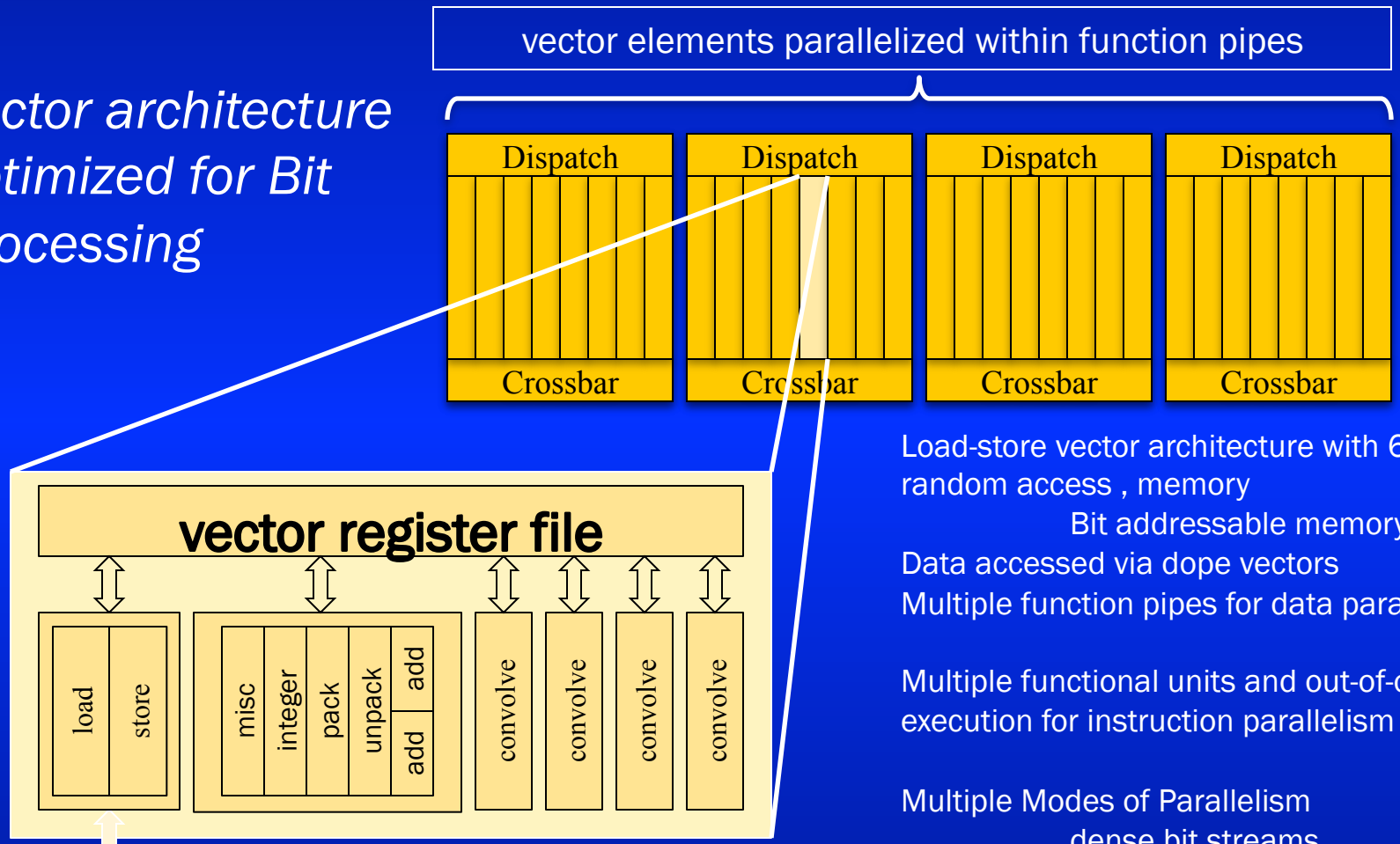
Dope Vector



- Column_(start, end bit)
  - Index_A (1,3)
  - Index_A (n: 1)
  - Index_B(0,2)
  - Index_B (n-3,1)
  - 32 or 64 bit index
  - Usually very sparse data set
  - Column Update – new dope vector

Node Logical Convolver

Vector architecture optimized for Bit Processing



Load-store vector architecture with 64 bit random access , memory

Bit addressable memory ?

Data accessed via dope vectors

Multiple function pipes for data parallelism

Multiple functional units and out-of-order execution for instruction parallelism

Multiple Modes of Parallelism

dense bit streams

64 bit wide logical ALU

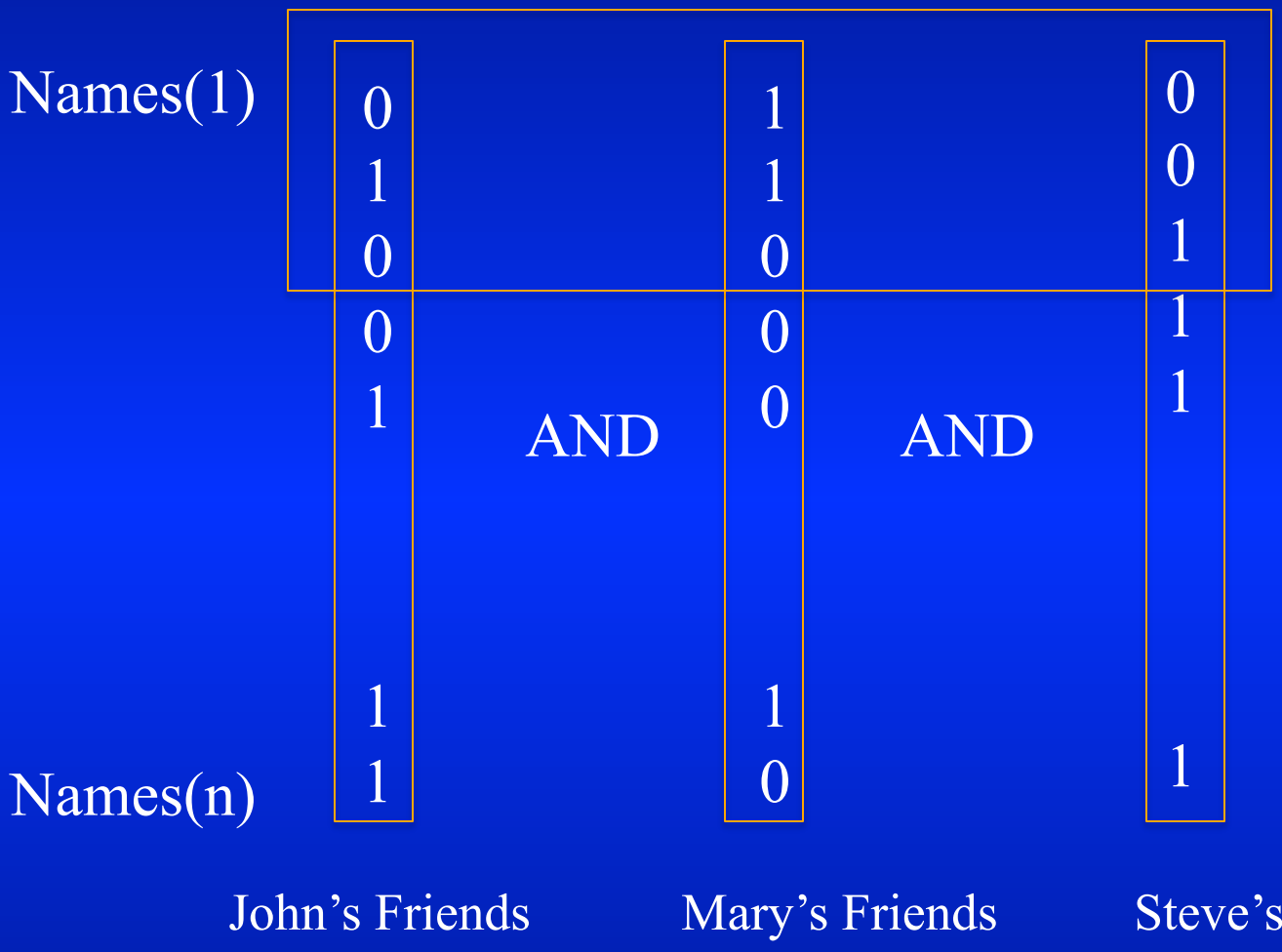
CONVEY computer

Example - Friends of Friends

- Dope Vectors friends of John
- Dope Vectors friends of Mary
- Dope Vectors friends of Steve
- Perform bitwise (index-wide 3-input AND)
  - Dope Vectors per Meta Tag Pre-computed as is sparse matrice's

- Benefits
  - Easy to understand
  - Easy to update
    - Add or Delete entries
    - No synch needed once operation begins
- Coding (via Instrinsic)
  - Z (i) = Convolve (AND, John, Mary, Steve)

Adjacency Matrix- Compute Model



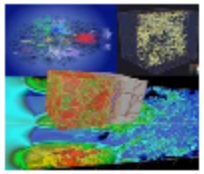
Parallelize block Rows – no sync Needed

Parallelize Columns

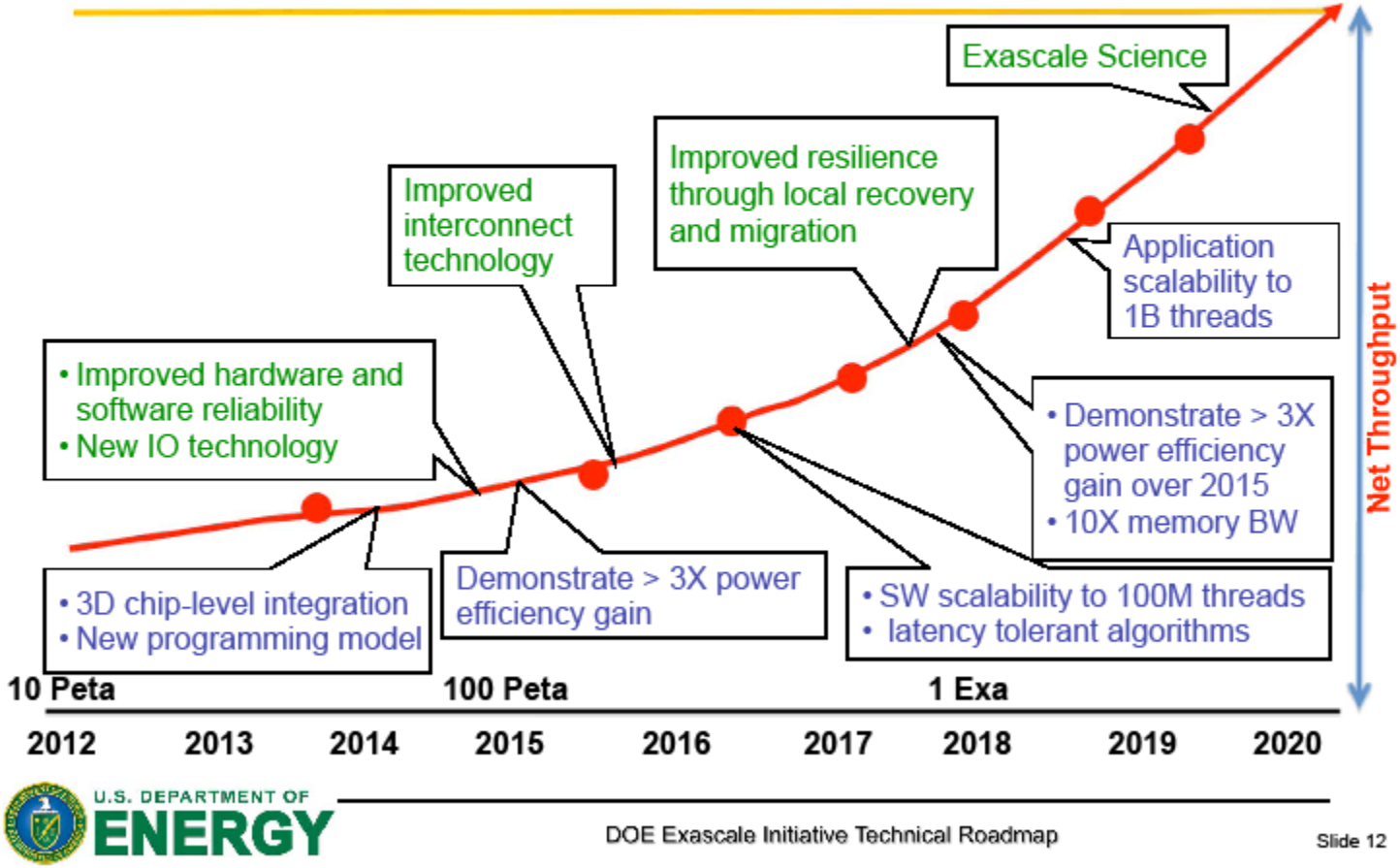
Run-Time Decision

Interesting Properties

- Can compute with threads if necessary
- Can define some interesting operators like
  - Majority (N out of K attributes)
  - Markov weighted edges
- The compute is straight forward, especially if the memory system is SMART
  - Probably a compute/time memory tradeoff
  - Tagged memory address's
  - Bit, Byte, Word Addressability (Deja Vu)
  - Short circuit logical operators
- The programming model is straightforward
  - From APL and MATLAB
    - **perspicuous**



Technology Roadmap



U.S. DEPARTMENT OF ENERGY

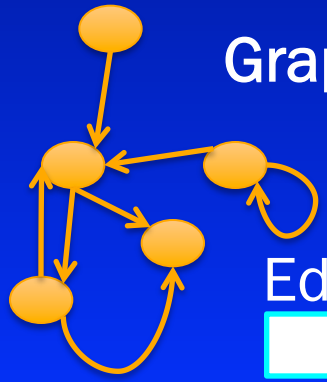
DOE Exascale Initiative Technical Roadmap

Slide 12

Exascale Workshop Dec 2009, San Diego

Development Frameworks for Multiple Architectural Models

- *CHOMP for Arrays of Customized Cores*
  - MIMD parallelism with many simple cores
  - OMP based shared memory programming model
  - Compile and run user model
- Convey Vectorizer for Custom SIMD
  - vector personalities optimized for specific data types
  - automatic vectorization of C/C++ and Fortran
  - support for customized instructions via intrinsics
- PDK for Algorithmic Personalities
  - specific routines in hardware
  - pipelining and replication for very high performance
  - PDK supports implementations in Verilog or with high level design tools



Graph Theory

Vertex Vector

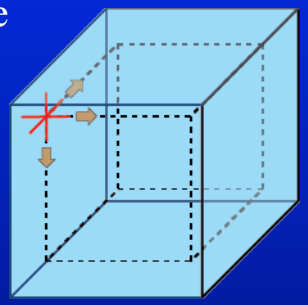
Edge Vector

, etc, etc

Genomics/Proteomics

ACTGTGACATGCTGACATGCTAG

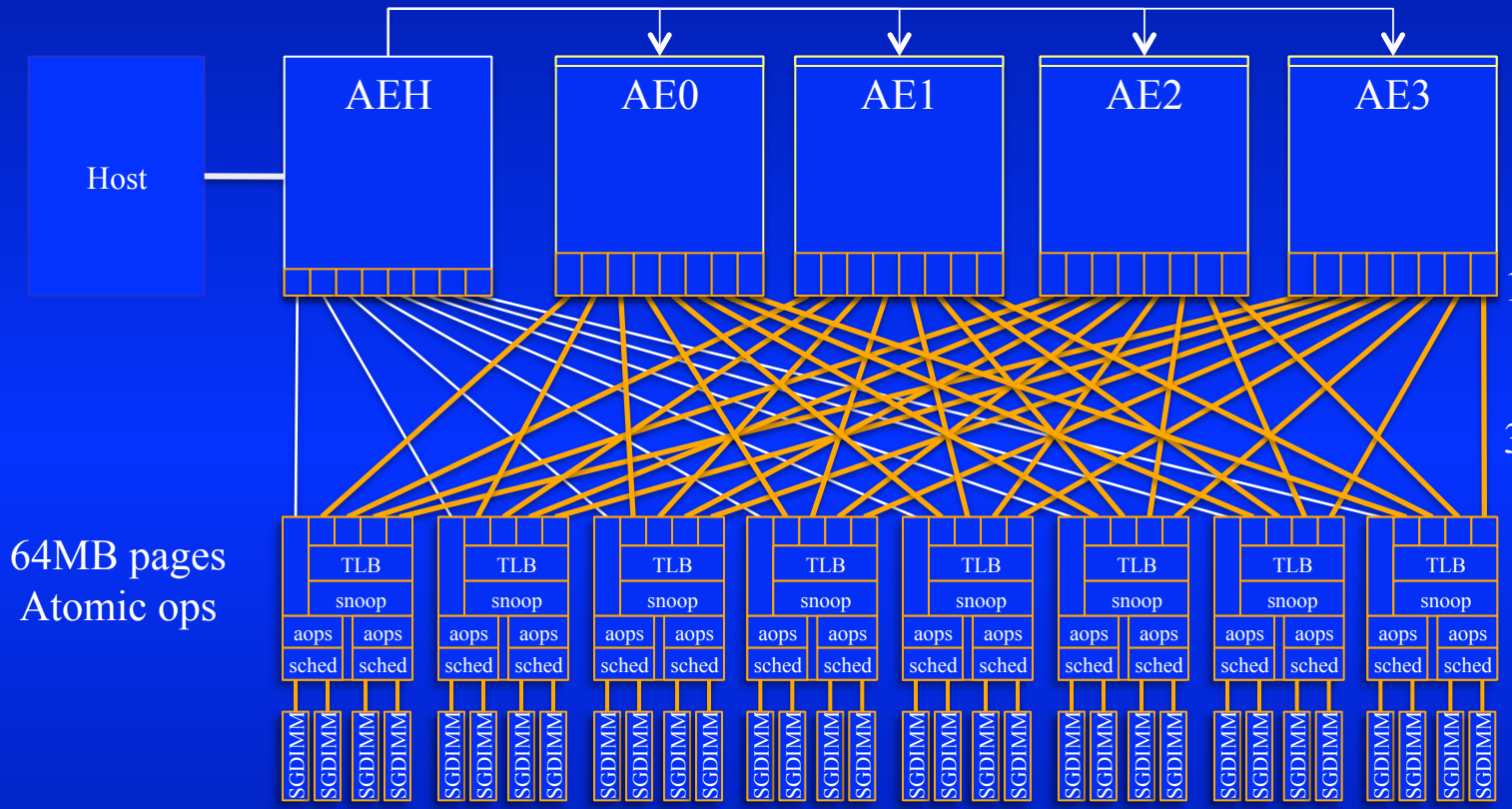
Stencils [FD/FV/FE]



```
X(I,J,K) = S0*Y(I  ,J  ,K  )
         + S1*Y(I-1,J  ,K  )
         + S2*Y(I+1,J  ,K  )
         + S3*Y(I  ,J-1,K  )
         + S4*Y(I  ,J+1,K  )
         + S5*Y(I  ,J  ,K-1)
         + S6*Y(I  ,J  ,K+1)
```

CONVEY computer

Fox Single-node Block Diagram



64MB pages
Atomic ops

1024 TIDs per link (32K OR)
32TB (45 bit) physical address space

32 DDR3 SG-DIMMs
HalfDIMM up to 4GB - 256GB system
FullDIMM up to 32GB - 1TB system

CONVEY computer

Graph 500 Results: Performance/ Power

Rank	Installation Site	Machine	Number of nodes	Number of cores	Problem scale	GTEPS
★	Convey Computer Corporation	fox6 / Convey MX-100	1	12	29	14.56
★	Convey Computer Corporation	thunder4 / Convey HC-2ex	1	12	27	11.47
★	Argonne Nat. Lab	HC-2ex, host-210	1	8	27	11.44
★	Univ. Calif. Riverside	yosemite / HC-2ex, host-210	1	8	27	11.44
★	Convey Computer Corp.	coconino / HC-2ex, host-210	1	8	27	11.44
25	Chuo University	Intel(R) Xeon(R) CPU E7- 4870 @ 2.4	1	40	26	8.15
27	Convey Computer Corporation	thunder4 / Convey HC-2ex	1	12	27	7.85
31	Convey Computer Corporation	Vortex / Convey HC-1ex	1	4	27	6.64
33	Virginia Bioinformatics Institute	convey-ex01/Convey HC-1ex	1	4	27	6.00
38	Chuo University	Intel Xeon E5-2690 2.90GHz (2 socke	1	16	29	4.22
40	Institute of Computing Technology, C	I840	1	32	30	3.69
41	Chuo university	AMD Opteron(tm) Processor 6174 2.	1	48	29	3.24
43	Chuo University	GraphCREST-M48 AMD Opteron 617	1	48	29	2.42
44	Chuo University	Intel(R) Xeon(R) CPU E7-4870 @ 2.4(	1	40	30	2.16
45	Imperial College London	Convey HC-1 Server	1	6	24	1.72
46	SNL	Dingus	1	4	28	1.72
46	SNL	Wingus	1	4	27	1.72
47	Convey Computer Corporation	Vortex	1	4	28	1.61
47	Bielefeld University, CeBiTec	Convey01	1	4	28	1.61
47	Convey Computer Corporation	Hc1-d	1	4	28	1.61
48	LBL/NERSC	Convey2	1	4	28	1.50
53	Chuo University, Tokyo Japan	GraphCREST-W12	1	12	0	1.18
63	Chuo University, Tokyo Japan	GraphCREST-M48	1	48	27	0.83
64	Convey Computer Corporation	Convey XC-1ex	1	4	27	0.76
65	Intel Research	Westmere E7-4870 2.4GHz	1	40	27	0.70
77	LLNL	Leviathan + FusionIO	1	40	36	0.05
77	LLNL	Appro	1	32	34	0.05

CONVEY computer

★= Convey

System Level - Trends

- Global Flat, Virtual Address Space
  - Common to Flops and Ops
  - Program using a PGAS Language
    - *Single Program Multiple Data* (SPMD)
  - 64 bits growing to 128 bits in 2020 and beyond

Control – Physical Memory Level
Dram – Flash – Disk
Node Field – PGAS model
Data Semantics – PIM Control
private vs. shared

4 KB – 1 MB Page Size

4 PetaBytes To 1 ExaBytes of Physical Address

Exascale Programming Model

Example 4.1-1: Matrix by Vector Multiply

```
1: #include<upc_relaxed.h>
2: #define N 200*THREADS
3: shared [N] double A[N][N];   NOTE: Thread is 16000
4: shared double b[N], x[N];
5: void main()
6: {
7: int i,j;
8: /* reading the elements of matrix A and the
9: vector x and initializing the vector b to zeros
10: */
11:    upc_forall(i=0;i<N;i++;i)
12:            for(j=0;j<N;j++)
13:          b[i]+=A[i][j]*x[j] ;
14: }
```

Finally

Flops/Ops /Watt

The which is the simplest to program will win. USER cycles are more important that CPU cycles