USC Viterbi
School of Engineering

Exascale: Can My Code Get From Here To There?

Information Sciences Institute

ISI
Information Sciences Institute
USC

22 June 2010
Bob Lucas
rflucas@isi.edu

Outline

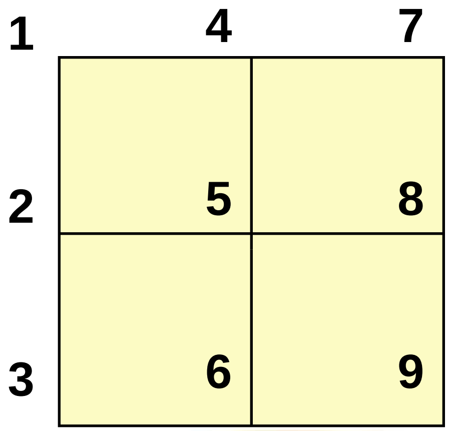
My Merely Terascale Sparse Solver

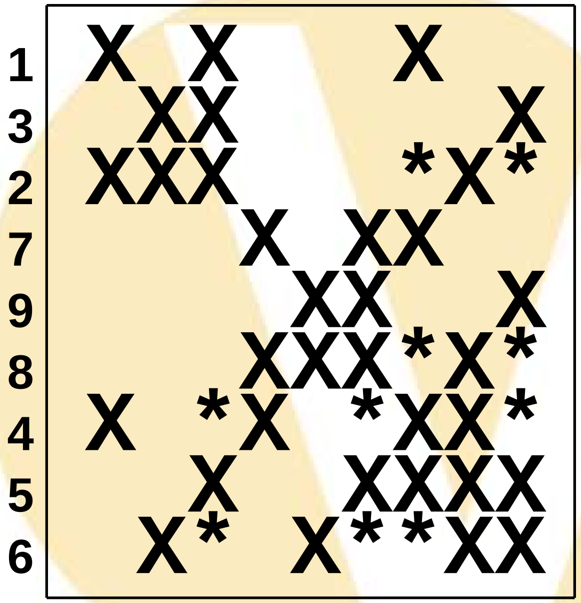
Today’s Execution Model (i.e., Here)

Exascale Expectations (i.e., There)

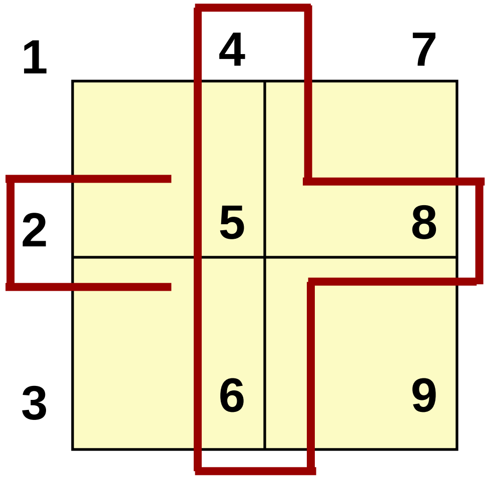
Gaussian Elimination
Toy Problem

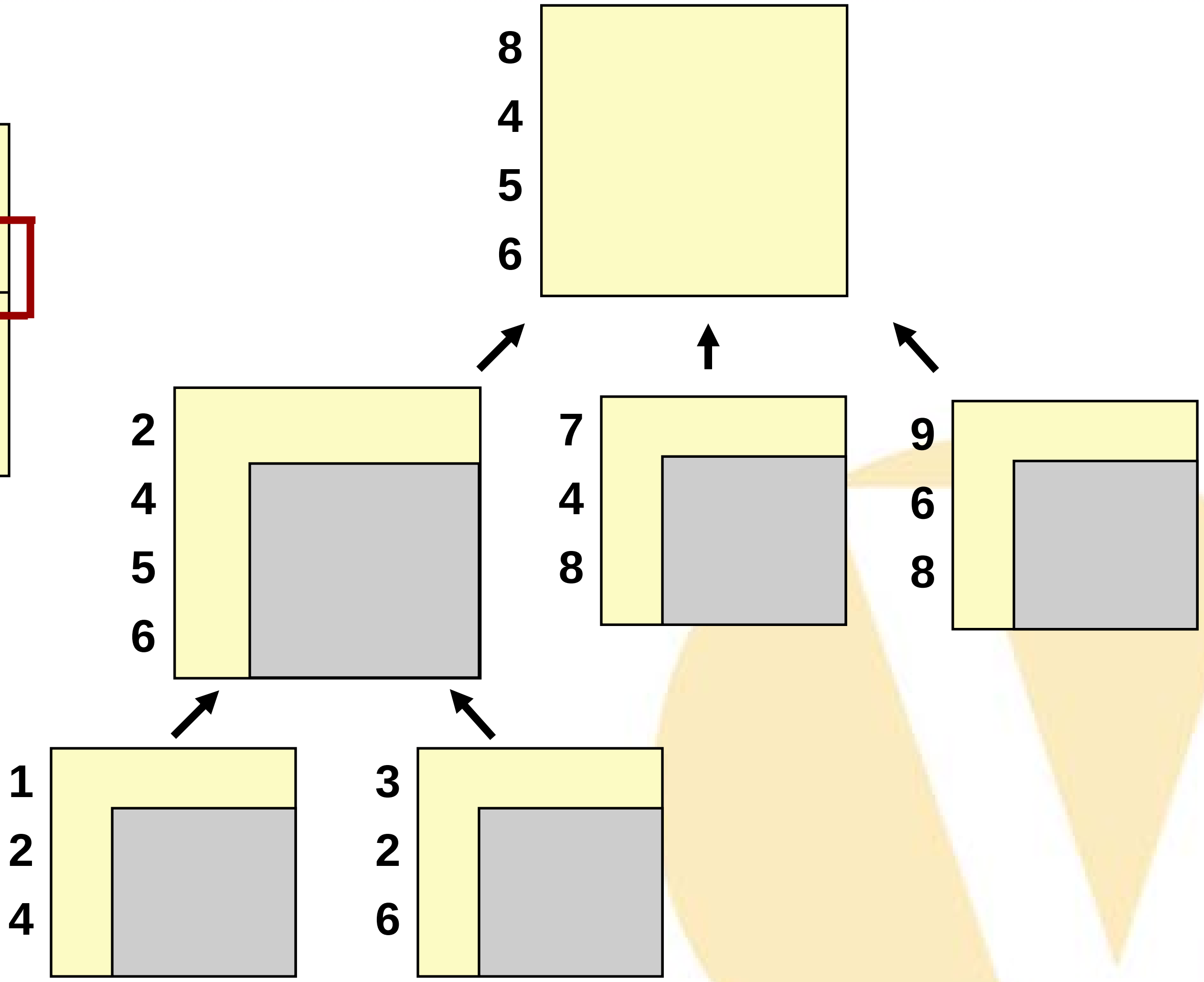
```
      do 4 k = 1, 9
        do 1 i = k + 1, 9
          a(i, k) = a(i,k) / a(k,k)
1       continue
        do 3 j = k + 1, 9
          do 2 i = k + 1, 9
             a(i,j) = a(i,j) –
     1                   a(i,k) *
     2                   a(k,j)
2         continue
3       continue
4     continue
```





Multifrontal View of the Toy Matrix

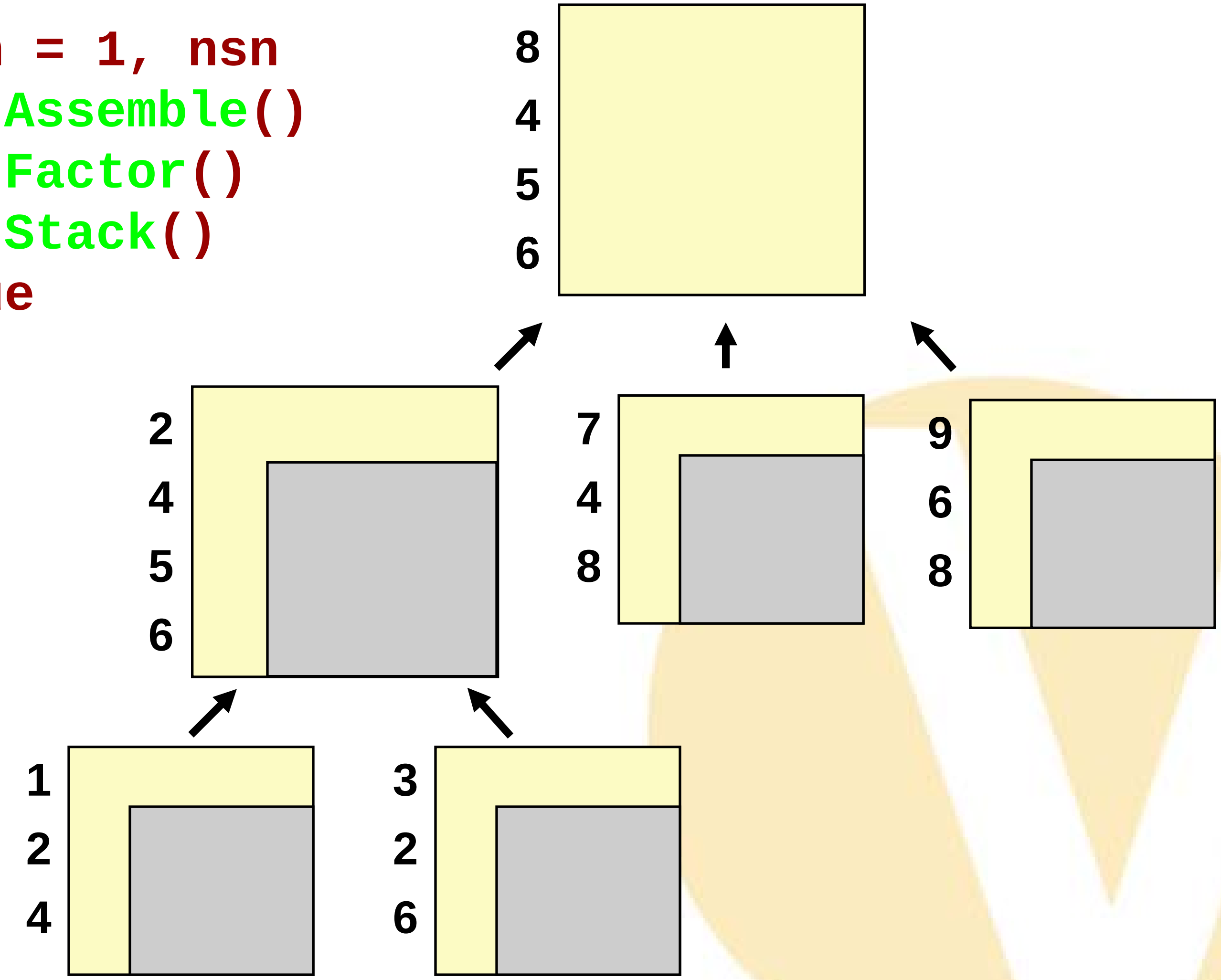




Duff and Reid, ACM TOMS 1983

Notional Control Flow

```
      do 1 sn = 1, nsn
         call Assemble()
         call Factor()
         call Stack()
1     continue
```

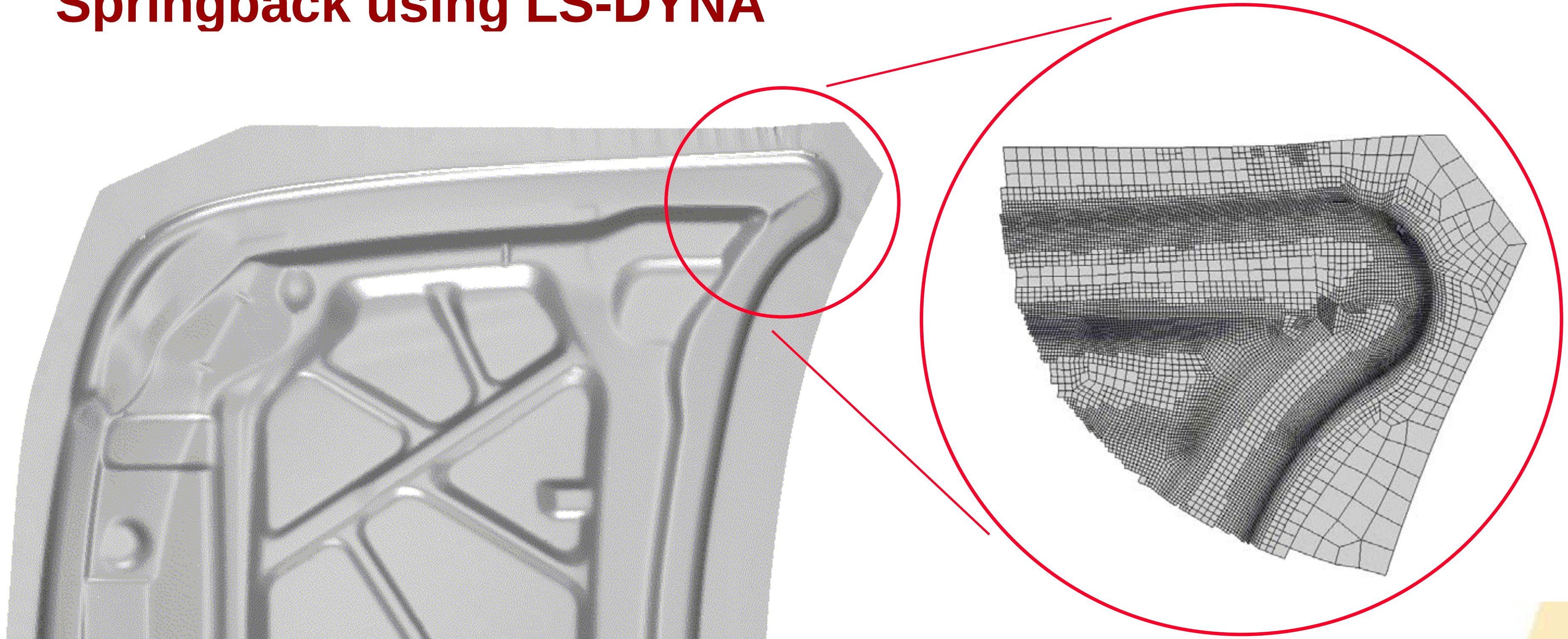


USC Viterbi
School of Engineering

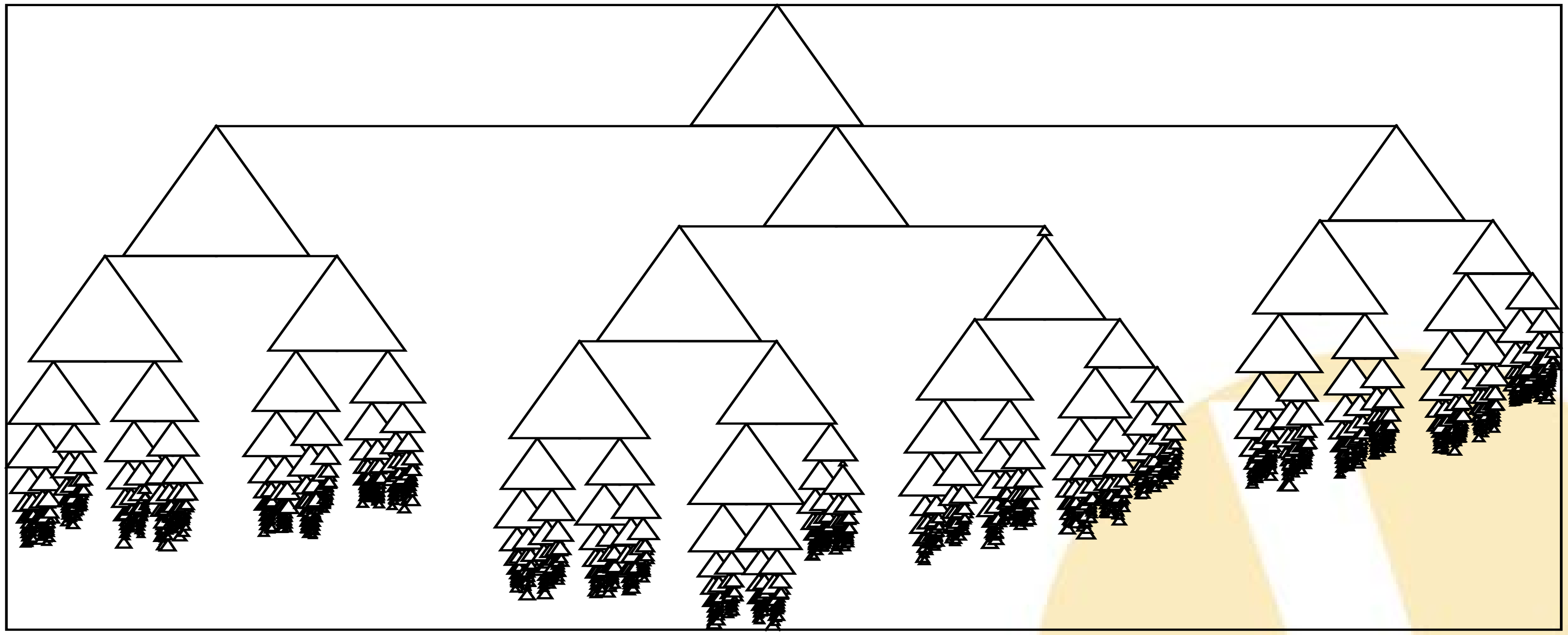
A Real Problem : “Hood”

ISI
Information Sciences Institute

Automotive Hood Inner Panel Springback using LS-DYNA



“Hood” Elimination Tree



Each frontal matrix’s triangle scaled by operations required to factor it.

Terascale Today

- Reordering is O(1%) Amdahl fraction
- I’m using sequential Metis
  - Memory bottleneck
  - Allocate all memory to one processor
- ParMetis and PT-Scotch
  - Modest parallel speedup
  - Lousy ordering inflates the other 99%
- Could get to Petascale given new reordering
  - 1994 MasPar version

Today’s Execution Model

Based on half a century of stability

Von Neuman CPUs	=> Fortran, C, C++, etc.

Evolutionary extensions

Distributed memory	=> MPI Library
SIMD extensions	=> SSE Directives
Multicore nodes	=> OpenMP Directives
Accelerators	=> CUDA, OpenCL

All of the above require user intervention

- Nothing comes for free anymore

Distributed Memory

- User has to explicitly manage
  - Data distribution
  - Synchronization and communication
- Portability via libraries
  - IEEE 1516
  - MPI
- High latency is major Amdahl problem
  - Most is software overhead
  - Anton’s point-to-point latency is 200ns

SIMD Extensions

- Originally multimedia extensions (MMX)
- Energy expended, per Bill Dally

Issue instruction in Pentium	~2000pJ
Issue instruction in Fermi	~200pJ
Perform floating point operation	~50pJ

- Amortize instruction issue over more ops.
- Requires:
  - Double-word data alignment (still?)
  - Padding of array leading dimension
  - Directives

Multicore Nodes

Dennard scaling has ended

Clock frequencies have plateaued

Moore’s Law continues unabated

Multiple cores per die

Coherent shared memory

Exploit with OpenMP (Pthreads, etc.)

Ideally simple and intuitive:

```
!$OMP PARALLEL DO
   do i = 1, dma_len
     front(p + i - 1) = front(p + i - 1) +  ltmp(i)
   end do
```

Multicore Nodes

- Dennard scaling has ended
  - Clock frequencies have plateaued
- Moore’s Law continues unabated
  - Multiple cores per die
  - Coherent shared memory
- Exploit with OpenMP (Pthreads, etc.)
- Ideally simple and intuitive:

```
!$OMP PARALLEL DO
      do i = 1, dma_len
        front(p + i - 1) = front(p + i - 1) +  ltmp(i)
      end do
```

Notional SMP Control Flow

```
      do 3 level = leaves, root
        if (sn_count(level) .gt. num_threads) then
c$omp paralleldo
          do 1 sn = ptr(level), ptr(level + 1) - 1
            call Seq_Assemble()
            call Seq_Factor()
1         continue
        else
          do 2 sn = ptr(level), ptr(level + 1) - 1
            call Seq_Assemble()
            call SMP_Factor()
2         continue
        end if
        call Storage_Recovery()
3     continue
```

Quickly Gets Ugly

```
#if 1
C_DOALL_PARALLEL
C_SHARED1 (wave,    jwave,   iwave,  l2D,     ln,     sp)
C_SHARED2 (tasks,   msglvl,  msgnum, indices, jv,     iv)
C_SHARED2 (l,       KObjPtr, KObjVal, alpha,  pvtTweak)
C_SHARED2 (Mexists, jm,      im,     m,       K_out)
C_SHARED2 (rs_num,  RS_out,  k_head, k_line,  k_num)
C_SHARED2 (L_out,   cleveX,  small,  sigma,   Ltrans)
C_SHARED2 (neq,     xl,      tmplen, l2Darray, my_err)
C_SHARED2 (nsn,     my_max,  my_min, my_lnz,  my_ops)
C_SHARED2 (my_clprt, my_mxd, hermtn, mom,     l2D_ptr)
C_SHARED2 (pvtThrsh, rs_head, rs_line, task_map, offset)
C_SHARED2 (sqz_prec, saunders, my_rv1, my_rv2)
C_PRIVATE (iw,      smp_sn,  tid,    s1,      s2)
C_PRIVATE (s3,      sp_tmp,  p1,     lerr,    sz,     dg)
C_PRIVATE (ld,      lp,      ip,     pp,      op,     rs)
C_PRIVATE (pi,      xtp,     xl2D,   alpha2)
C_PRIVATE (rip,     ibp,     sbp,    l2p)
C_DYNAMIC
#endif
```

Accelerators

- Long history in scientific computing
  - e.g., Floating Point Systems
- Now exploiting devices for gaming/graphics
  - Enhance end-user experience
  - Independent of scientific computing
- New architectures
  - Have to rethink algorithms
- New programming languages
  - Directives for standard languages

Fortran vs CUDA

```
do j = jl, jr
  do i = jr + 1, ld
    x = 0.0
    do k = jl, j - 1
      x = x + s(i, k) * s(k, j)
    end do
    s(i, j) = s(i, j) - x
  end do
end do
```

```
ip=0;
for (j = jl; j <= jr; j++) {
  if(ltid <= (j-1)-jl){
    gpulskj(ip+ltid) = s[IDXS(jl+ltid,j)];
    }
  ip = ip + (j - 1) – jl + 1;
  }

__syncthreads();

for (i  = jr + 1 + tid; i <= ld;
      i += GPUL_THREAD_COUNT) {
  for (j = jl; j <= jr; j++) {
    gpuls(j-jl,ltid) = s[IDXS(i,j)];
    }
  ip=0;
  for (j = jl; j <= jr; j++) {
    x = 0.0f;
    for (k = jl; k <= (j-1); k++) {
      x  = x  + gpuls(k-jl,ltid) * gpulskj(ip);
      ip = ip + 1;
      }
      gpuls(j-jl,ltid) -= x;
    }
  for (j = jl; j <= jr; j++) {
    s[IDXS(i,j)] = gpuls(j-jl,ltid);
    }
  }
```

Expectations for Exascale

Seminal DARPA study

- Peter M. Kogge (editor), "Exascale Computing Study: Technology Challenegs in Achieving Exascale Systems", Univ. of Notre Dame, CSE Detp. Tech. Report, TR-2008-13, Sept. 28, 2008

Principle challenges

Concurrency	O(1B ALUs)
Energy	Hundreds of MWs
Memory	Falling off Moore's Law
Resilience	Soft error rate skyrockets

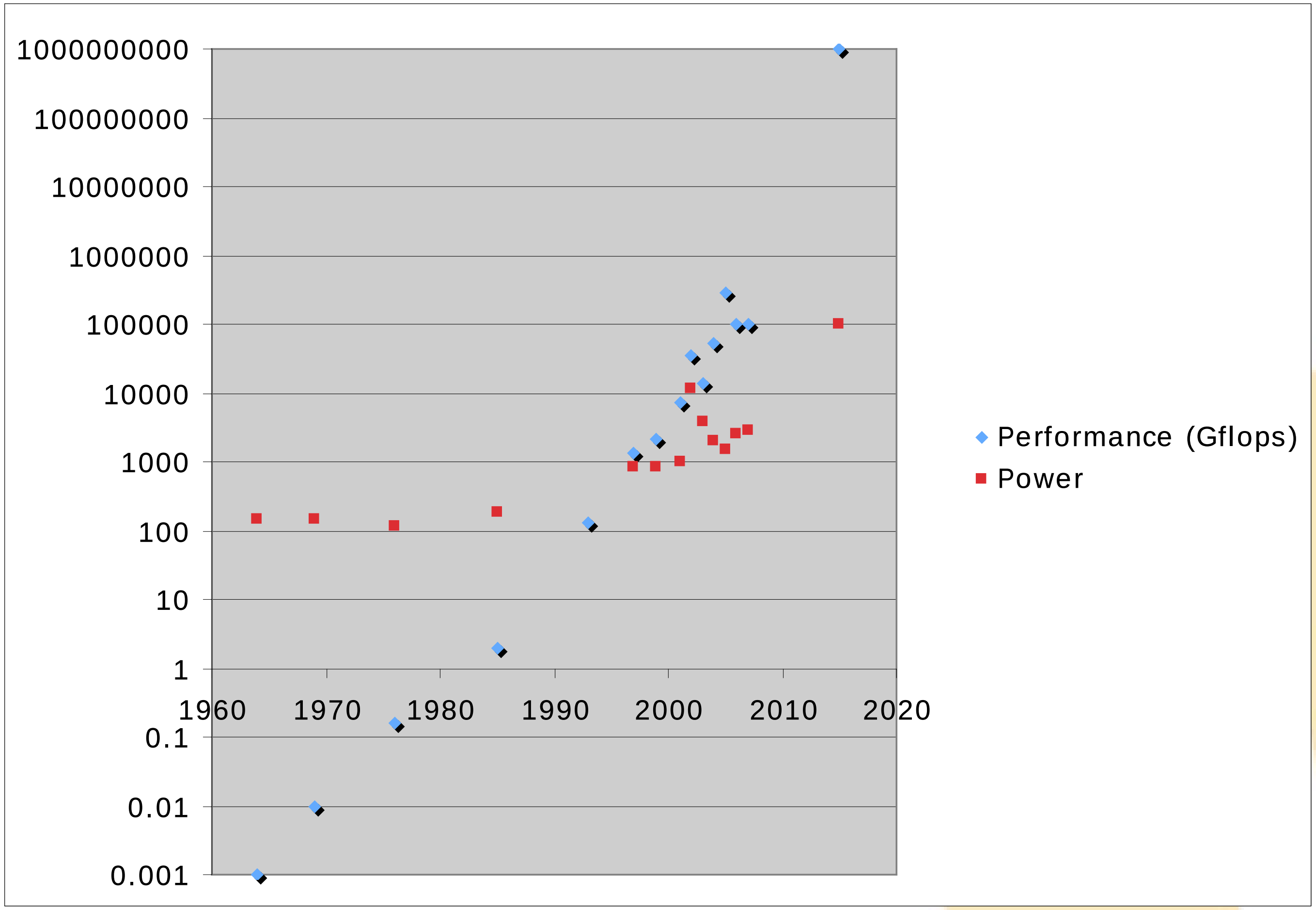
Concurrency

- Rate of growth accelerating
- With multithreading, it could reach billions
  - What's the Amdahl fraction of that?
  - May need to rediscover fine-grain SIMD
- Familiar synchronization will be prohibitive
  - Dot products in Krylov-space algorithms
  - Reductions for error state or time quanta
- We'll have to rethink a lot of mathematics
  - Somebody needs to invent a new reordering
  - Otherwise, I can't get there

Energy

- Data movement will dominate energy
  - What's the abstraction for this?
  - I expect explicit machine model
  - Will need to overlay with virtual model
- Heterogeneous processing nodes
  - SIMD nodes to minimize instruction issue
  - Low-latency nodes for Amdahl fractions
  - Only power up those cores you need
  - AMD's Fusion is just the beginning

Power Perspective



GigaFlop/s

KiloWatts

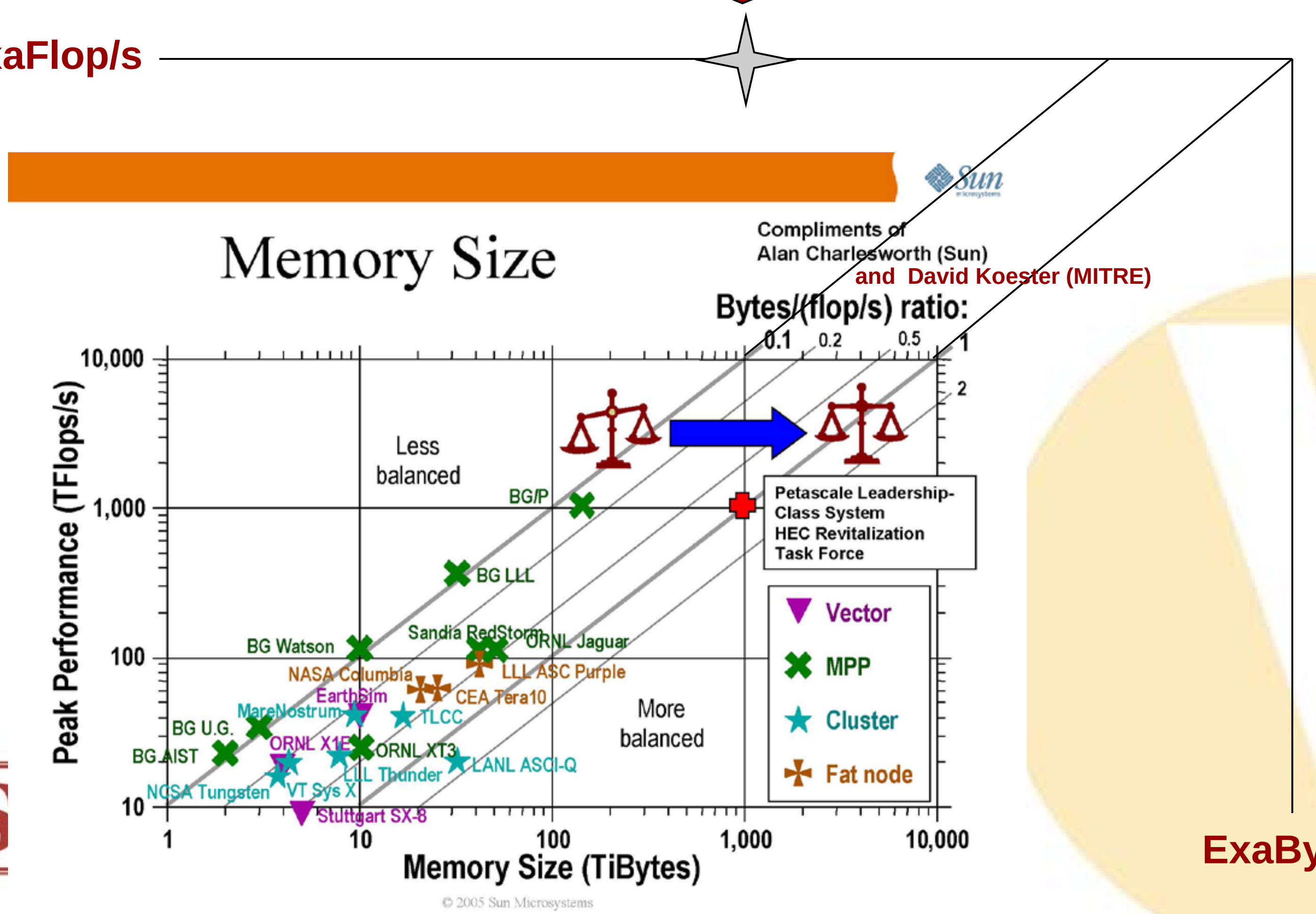
Memory

- End of Moore's Law for DRAM before logic
  - DRAM structures are 3D
- Won't have luxury of redundant data
  - Material properties tables
  - Executables
- Problem since shared data can't be local
  - Requires energy to move it
- I expect explicit memory hierarchies
  - Already seen it in Cray 2, Cell, & GPUs
  - What's the programming abstraction?

Memory Capacity Challenge

1,000,000 DRAM chips, circa 2014

ExaFlop/s



ExaBytes

Resilience

- Check pointing won’t be adequate any more
  - Error rates will grow faster than I/O B/W
- Memory and networks protected with ECC
- What about processors and arithmetic?
  - Can’t afford blanket use of redundancy
- Need a new programming abstraction here too
  - Ignore some errors (e.g., HPCS Random Access)
  - Correct others (e.g., Iterative Refinement)
  - Trade energy and/or performance for resilience.

Summary

- Some people will make it to Exascale
  - I'm literally betting on Malcomb Stocks
- I'm not sure if my solver will
  - First there's the reordering problem
  - Then load imbalance & critical path length
  - Then "whack a mole" with other bottlenecks
- I believe the programming model needs to evolve
  - Don't unnecessarily throw away working code
  - Performance programmers manage everything
  - I don't expect that to change