Industry Panel – Cray Inc.

What will be your company's HPC successes in 2012?

SOS 13
Hilton Head, SC
March 10, 2009

Steve Scott
Cray CTO

With users' relentless appetite for HPC we could expect systems with 100 PFlops peak soon after ~2012.

- *What type of systems do you think will first achieve this?*
- *Will they be general-purpose?*
- *How much electrical power will they consume?*
- *What will be the standard way of programming applications for these systems?*

First 100PF Systems (Steve's Guesstimate)

- Probably the first 100PF system will use "accelerators"
  - Recall MD Grape hit a PF in 2006, two years before Roadrunner and Jaguar
  - Could be FPGAs, more likely GP-GPUs
  - 2013, ~10-25 MW
  - *Maybe* could be called general purpose…
  - General purpose:
    - Automatic compilation: Drop on your existing parallel codes and go!
    - Needs to be broadly applicable: not just coarse-grained, localized codes
- First truly general purpose machine
  - 2014
  - 1-2 TF sockets (manycore with SIMD extensions)
  - 50-100K sockets (1-2x the size of Jaguar)
  - ~20-30 MW (depending on processors and memory)
- Programming
  - Still mostly MPI
  - Growing use of hierarchical (OpenMP under MPI)
  - Growing use of CAF, UPC, Global Arrays and Shmem (Chapel?)

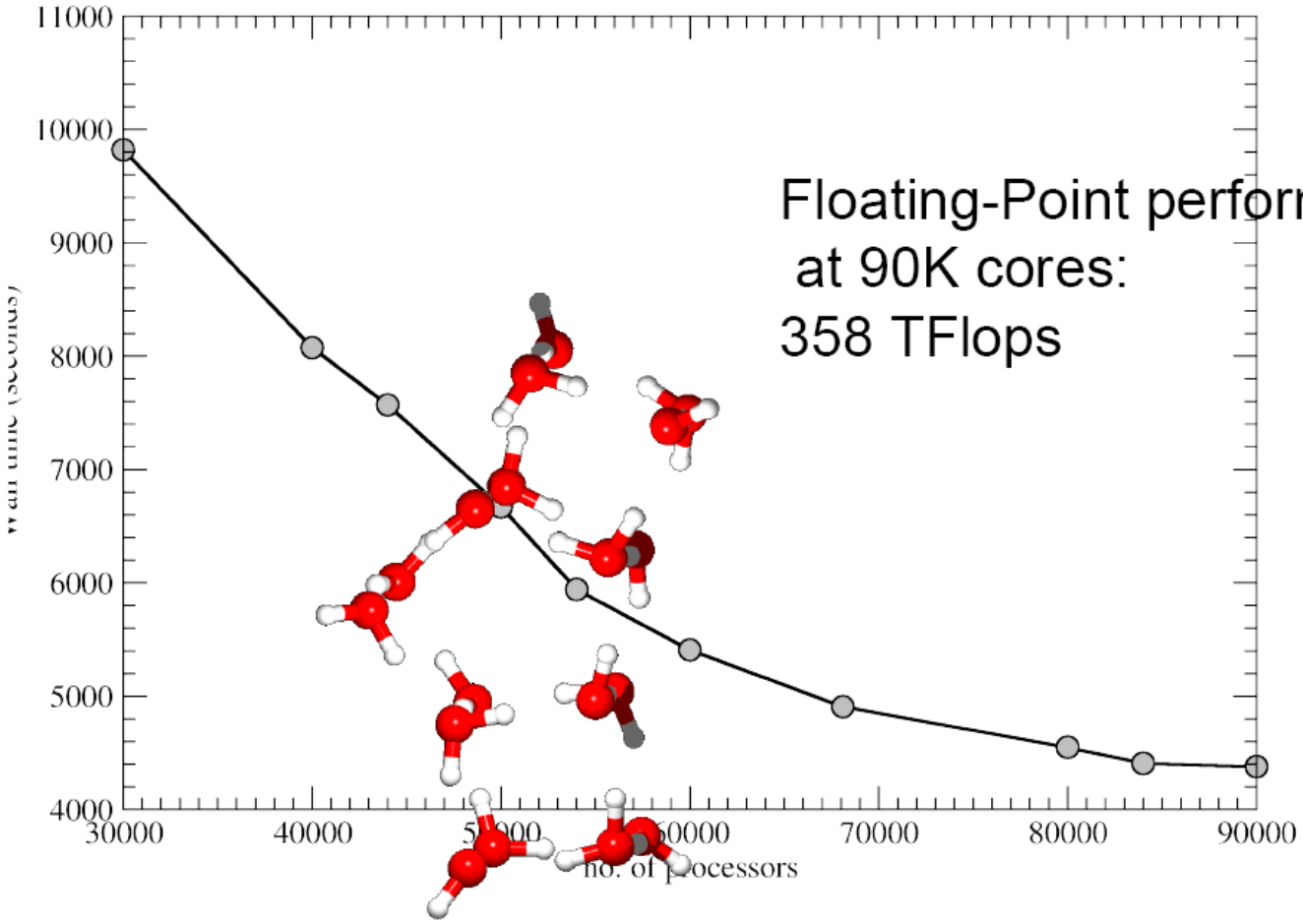
Petaflop Jaguar XT5 System

Eight Application World Records Set in First Week!

Science Area	Code	Contact	Cores	% of Peak	Total Perf	Scaling
Materials	DCA++	Schulthess	150144	97%	1.3 PF**	Weak
Materials	LSMS/WL	ORNL	149580	76.40%	1.05 PF	Weak
Seismology	SPECFEM3D	UCSD	149784	12.60%	165 TF	Weak
Weather	WRF	Michalakes	70000	5.60%	36 TF	Strong
Climate	POP	Jones	18000	3.00%	5 TF	Strong
Combustion	S3D	Chen	144000	6.00%	83 TF	Weak
Fusion	GTC	PPPL	102000		20 B particles pushed	Weak
Materials	LS3DF	Lin-Wang Wang	147456	32%	442 TF	Weak

** 2008 Gordon Bell prize winner! Mixed precision; 626 TF at 128K cores in 64 bit only.

CCSD(T) run on Cray XT5 : 18 water



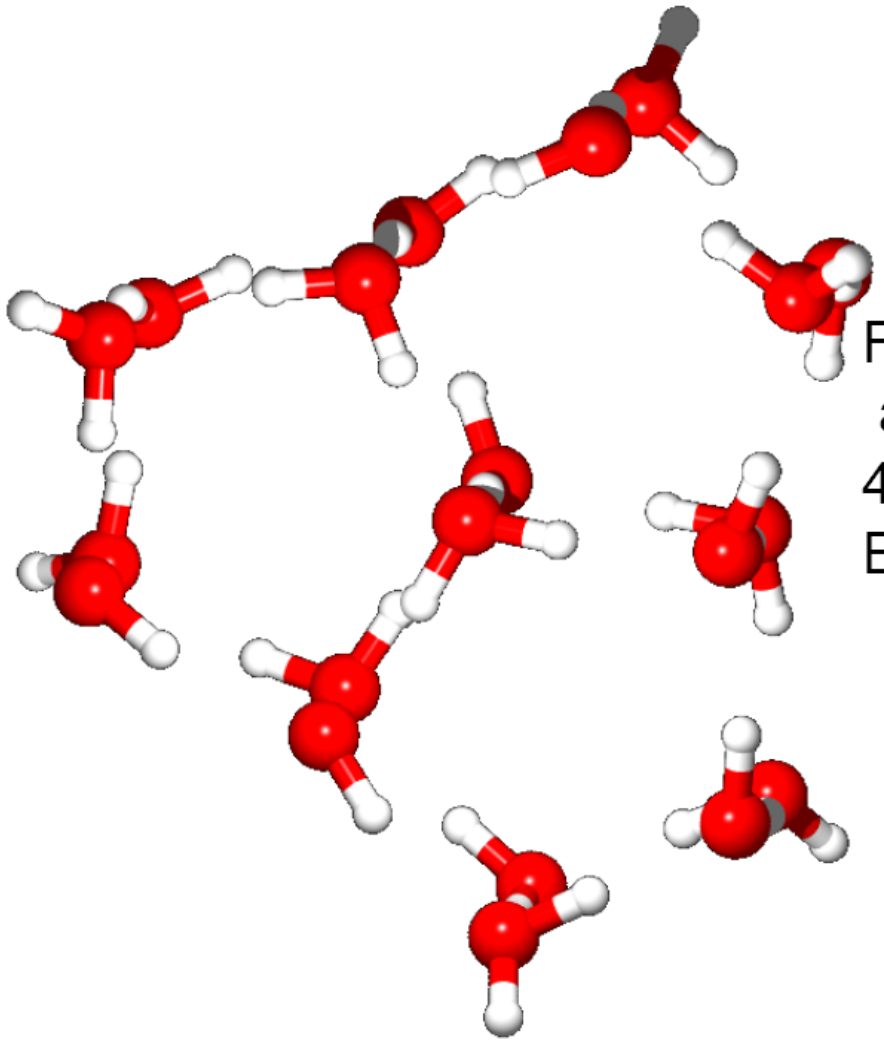
Floating-Point performance
at 90K cores:
358 TFlops

$(H_2O)_{18}$

54 atoms
918 basis functions
Cc-pvtz(-f) basis

Managed by UT-Battelle for the
Department of Energy

CCSD(T) run on Cray XT5 : 20 water



Floating-Point performance
at 92K cores:
475 TFlops
Efficiency > 50%

$(H_2O)_{20}$

60 atoms
1020 basis functions
Cc-pvtz(-f) basis

The HPC industry faces huge challenges including mounting power consumption, maintaining system availability with increasing component volumes, decreasing memory bandwidth per core, software to scale to millions of processors. Yet vendors seem able to re-invent solutions on a regular basis.

- *What paradigm shifts, if any, do you see occurring by 2012?*
- *Where might you look for new partners?*

Predict No Major Paradigm Shifts by 2012

- Minor paradigm shifts
  - Maybe some more use of accelerators
    - I believe that we can build compilers that will allow the use of standard programming models
    - Skeptical on breadth of applicability
    - Must *significantly* reduce synchronization and communication overhead
  - Maybe some better programming models
    - PGAS languages *are* more productive and more efficient.
    - Next year's Cray XT system will be a great PGAS machine
  - Parallel compilers and tools will continue to get better
    - Debugging at scale, automatic performance analysis, adaptive libraries with off-line optimization, etc.
    - Automatic parallelization at the node level: cores, threads, vectors
- Many areas of potential partnerships
  - Processors: New microarchitectures, accelerators
  - Optical signaling technology (AOC for now, directly off package by ~2015)
  - Local memory packaging: optics, stacking, ?
  - Tools: debuggers, compilers, etc.
  - Scalable file systems
  - *Community: Application resiliency, programming languages/models*

The current world economy is drastically different to any ever seen in the lifetime of the HPC industry. Credit may effectively disappear, funds may become more centrally controlled, hyperinflation may arise from capital injections, and markets may shrink. Yet HPC users benefit from healthy competition sustained by the current market size.

- *What do you think is your company's best strategy for survival?*
- *How do you think HPC customers can realistically help you?*

Surviving in HPC in the Global Recession

- 2008 was actually a great year for Cray
  - We don't focus on the consumer or business markets
  - Record revenue and gross profit
  - Nicely profitable except for non-cash write-down of goodwill in 4Q
  - Repurchased over half our long term debt
  - Diversified our business with the launch of the Cray CX1 and custom engineering business unit

- How can HPC customers realistically help us?
  - Keep buying systems – make the case to your management about the value of HPC
  - Procure systems based on sustained price-performance on representative workloads
  - Talk to us: what do you like and what don't you like about our systems?
  - Work with us to understand future application characteristics and needs
  - Work with us on new tool development
  - Work with us on scaling/testing of very large machines
  - Be open to new ways of programming where they can help (e.g.: PGAS)

A few closing thoughts….

Productivity Begins with the Architecture:
A Guidepost to Building Good Systems in 2012

- Global shared address space with one-sided data transfers
  - *So that code can easily reference and access objects held in remote nodes without involvement of code running on those nodes*
- High bandwidth, low granularity network
  - *So that programs can be written with far less concern about how and when communication takes place*
- Latency-tolerant processors
  - *So that compute capabilities do not go idle waiting for data, and programmers do not have to stage data and computation*
- Plentiful threading with efficient, lightweight synchronization
  - *So that parallelism can be dynamically exploited at multiple levels in the code, and programmers need to worry less about load balancing and synchronization*
- Adaptive processing capabilities
  - *So that idioms that would benefit from vectorization, streaming, fine-grain multithreading, or fast sequential processing can execute efficiently, and the programmer does not have to change the code to fit the paradigm*

Key Issues Over Next Five Years

- Power ('nuff said)
- System and application resiliency
  - 500 FIT processor has an expected lifetime of over 100 years
  - 100,000 processors → multiple failures per day (Al's "continuous failures")
  - Forget trying to make hardware MTBF acceptable (*deal* with failures)
  - "Easy" to make systems resilient; applications are the *really* hard part
- Local memory bandwidth technologies
  - Last 30 years: DRAM density has outpaced bandwidth by ~75 times
  - Memory bandwidth is limiting performance of future designs
  - Processors + DIMMs has to go
- Processor microarchitecture to exploit locality
  - Need a new microarchitecture and execution model ("co-design")
    - Much lower control overhead relative to computation
    - Much more aggressive exploitation of locality (explicit control of data movement?)
  - Must *not* burden the programmer with this
- Programming difficulty
  - Scale and concurrency
  - MPI is a low-productivity programming model (and *not* performace portable)
  - Time is right for a high productivity language
    - PGAS languages a good start, but prefer Chapel, X10, etc.

Chapel

A new parallel language developed by Cray for HPCS

Themes

- Raise level of abstraction, generality compared to SPMD approaches
- Support prototyping of parallel codes + evolution to production-grade
- Narrow gap between parallel and mainstream languages

Chapel's Productivity Goals

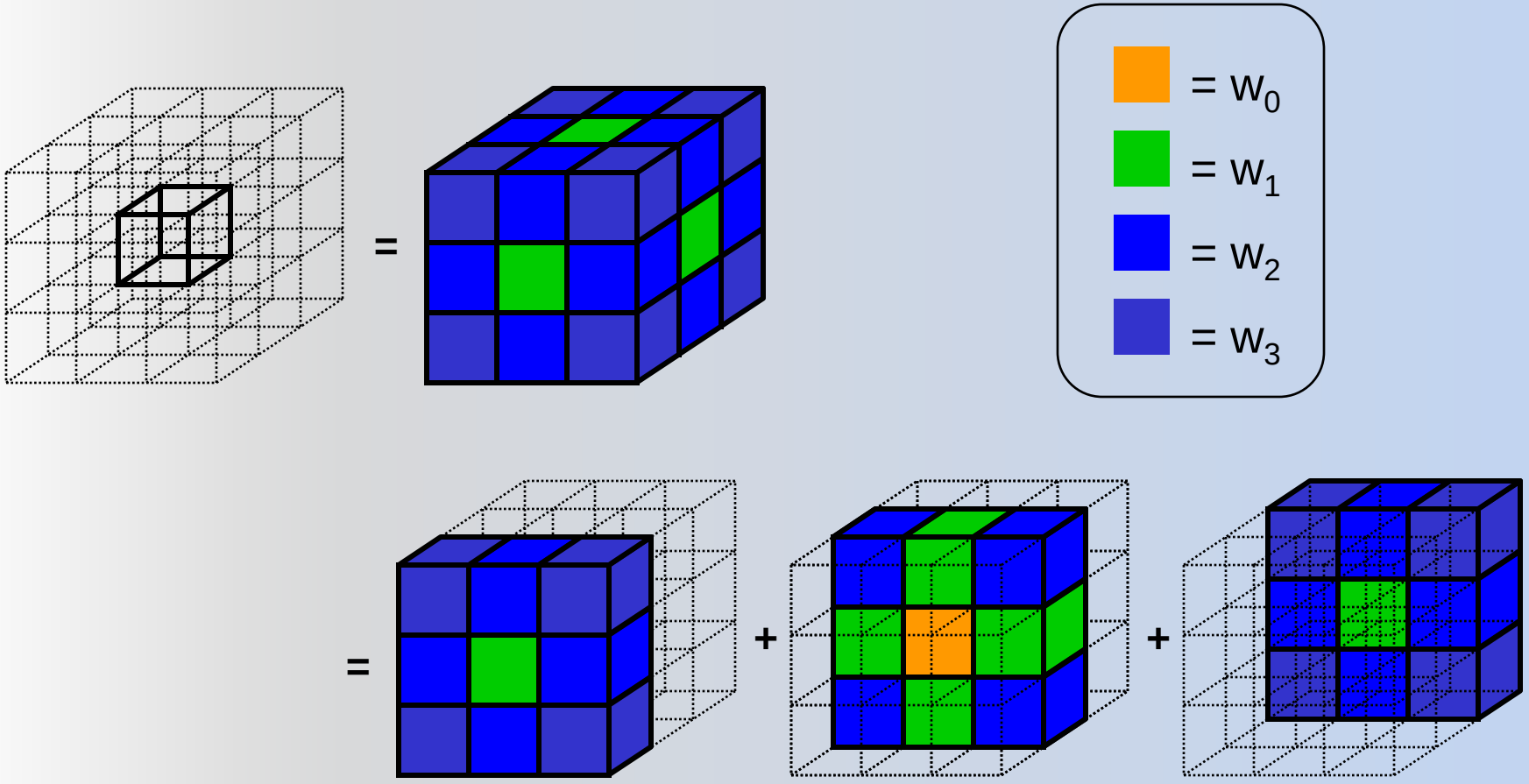
- Vastly improve **programmability** over current languages/models
- Support **performance** that matches or beats MPI
- Improve **portability** over current languages/models (actually better than MPI)
- Improve code **robustness** via better abstractions and semantics

Status

- Draft language specification available
- Portable prototype implementation underway
- Most effort to date has been focused on functionality and feature evaluation
- Early releases to ~90 users at ~30 sites (academic, government, industry)
- Public release planned for SC08

```
...
forall (_, r) in (Updates, RAStream()) do
  on T( r & indexMask ) do
    T( r & indexMask ) ^= r;
```

rprj3 Stencil from NAS MG



Fortran+MPI 3D 27-point Stencil (NAS MG *rprj3*)

```
      subroutine comm3(u,n1,n2,n3,kk)
      use caf_intrinsics

      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer n1, n2, n3, kk
      double precision u(n1,n2,n3)
      integer axis

      if( .not. dead(kk) )then
         do  axis = 1, 3
            if( nprocs .ne. 1) then
               call sync_all()
               call give3( axis, +1, u, n1, n2, n3, kk )
               call give3( axis, -1, u, n1, n2, n3, kk )
               call sync_all()
               call take3( axis, -1, u, n1, n2, n3 )
               call take3( axis, +1, u, n1, n2, n3 )
            else
               call comm1p( axis, u, n1, n2, n3, kk )
            endif
         enddo
      else
         do  axis = 1, 3
            call sync_all()
            call sync_all()
         enddo
         call zero3(u,n1,n2,n3)
      endif
      return
      end


      subroutine give3( axis, dir, u, n1, n2, n3, k )
      use caf_intrinsics


      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer axis, dir, n1, n2, n3, k, ierr
      double precision u( n1, n2, n3 )

      integer i3, i2, i1, buff_len,buff_id

      buff_id = 2 + dir
      buff_len = 0

      if( axis .eq.  1 )then
         if( dir .eq. -1 )then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  buff_len = buff_len + 1
                  buff(buff_len,buff_id ) = u( 2,  i2,i3)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
>           buff(1:buff_len,buff_id)

         else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( n1-1, i2,i3)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
>           buff(1:buff_len,buff_id)

         endif
      endif

      if( axis .eq.  2 )then
         if( dir .eq. -1 )then
            do  i3=2,n3-1
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( i1,  2,i3)
               enddo
            enddo

        buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
>       buff(1:buff_len,buff_id)
```

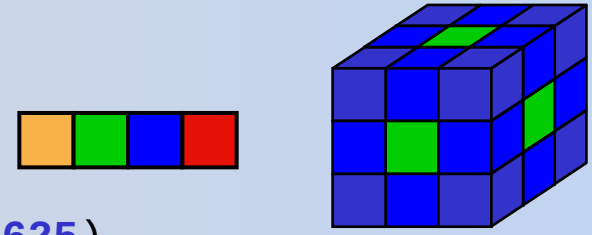
```
      else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len,  buff_id )= u( i1,n2-1,i3)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
>           buff(1:buff_len,buff_id)

         endif
      endif

      if( axis .eq.  3 )then
         if( dir .eq. -1 )then

            do  i2=1,n2
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( i1,i2,2)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
>           buff(1:buff_len,buff_id)

         else if( dir .eq. +1 ) then

            do  i2=1,n2
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( i1,i2,n3-1)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
>           buff(1:buff_len,buff_id)

         endif
      endif

      return
      end


      subroutine take3( axis, dir, u, n1, n2, n3 )
      use caf_intrinsics


      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer axis, dir, n1, n2, n3
      double precision u( n1, n2, n3 )

      integer buff_id, indx

      integer i3, i2, i1

      buff_id = 3 + dir
      indx = 0

      if( axis .eq.  1 )then
         if( dir .eq. -1 )then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  indx = indx + 1
                  u(n1,i2,i3) = buff(indx, buff_id )
               enddo
            enddo

         else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  indx = indx + 1
                  u(1,i2,i3) = buff(indx, buff_id )
               enddo
            enddo

         endif
      endif

      if( axis .eq.  2 )then
         if( dir .eq. -1 )then

            do  i3=2,n3-1
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,n2,i3) = buff(indx, buff_id )
               enddo
            enddo
```

```
      else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,1,i3) = buff(indx, buff_id )
               enddo
            enddo

         endif
      endif

      if( axis .eq.  3 )then
         if( dir .eq. -1 )then

            do  i2=1,n2
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,i2,n3) = buff(indx, buff_id )
               enddo
            enddo

         else if( dir .eq. +1 ) then

            do  i2=1,n2
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,i2,1) = buff(indx, buff_id )
               enddo
            enddo

         endif
      endif

      return
      end


      subroutine comm1p( axis, u, n1, n2, n3, kk )
      use caf_intrinsics


      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer axis, dir, n1, n2, n3
      double precision u( n1, n2, n3 )

      integer i3, i2, i1, buff_len,buff_id
      integer i, kk, indx

      dir = -1

      buff_id = 3 + dir
      buff_len = nm2

      do  i=1,nm2
         buff(i,buff_id) = 0.0D0
      enddo


      dir = +1

      buff_id = 3 + dir
      buff_len = nm2

      do  i=1,nm2
         buff(i,buff_id) = 0.0D0
      enddo

      dir = +1

      buff_id = 2 + dir
      buff_len = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               buff_len = buff_len + 1
               buff(buff_len,  buff_id )= u( n1-1,
     i2,i3)
            enddo
         enddo
      endif

      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len,  buff_id )= u( i1,n2-
     1,i3)
            enddo
         enddo
      endif
```

```
      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len, buff_id ) = u( i1,i2,n3-
     1)
            enddo
         enddo
      endif

      dir = -1

      buff_id = 2 + dir
      buff_len = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               buff_len = buff_len + 1
               buff(buff_len,buff_id ) = u( 2,  i2,i3)
            enddo
         enddo
      endif

      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len, buff_id ) = u( i1,
     2,i3)
            enddo
         enddo
      endif

      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len, buff_id ) = u( i1,i2,2)
            enddo
         enddo
      endif

      do  i=1,nm2
         buff(i,4) = buff(i,3)
         buff(i,2) = buff(i,1)
      enddo

      dir = -1

      buff_id = 3 + dir
      indx = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               indx = indx + 1
               u(i1,i2,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif

      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               indx = indx + 1
               u(i1,n2,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif

      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               indx = indx + 1
               u(i1,i2,n3) = buff(indx, buff_id )
            enddo
         enddo
      endif


      dir = +1

      buff_id = 3 + dir
      indx = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               indx = indx + 1
               u(1,i2,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif
```

```
      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               indx = indx + 1
               u(i1,1,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif

      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               indx = indx + 1
               u(i1,i2,1) = buff(indx, buff_id )
            enddo
         enddo
      endif

      return
      end

      subroutine rprj3(r,m1k,m2k,m3k,s,m1j,m2j,m3j,k)
      implicit none
      include 'cafnpb.h'
      include 'globals.h'

      integer m1k, m2k, m3k, m1j, m2j, m3j,k

      double precision r(m1k,m2k,m3k), s(m1j,m2j,m3j)
      integer j3, j2, j1, i3, i2, i1, d1, d2, d3, j
      double precision x1(m), y1(m), x2,y2

      if(m1k.eq.3)then
        d1 = 2
      else
        d1 = 1
      endif

      if(m2k.eq.3)then
        d2 = 2
      else
        d2 = 1
      endif

      if(m3k.eq.3)then
        d3 = 2
      else
        d3 = 1
      endif

      do  j3=2,m3j-1
        i3 = 2*j3-d3
        do  j2=2,m2j-1
          i2 = 2*j2-d2
          do j1=2,m1j
            i1 = 2*j1-d1
            x1(i1-1) = r(i1-1,i2-1,i3  ) + r(i1-1,i2+1,i3  )
     >               + r(i1-1,i2,  i3-1) + r(i1-1,i2,  i3+1)
            y1(i1-1) = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)
     >               + r(i1-1,i2+1,i3-1) + r(i1-1,i2+1,i3+1)
          enddo
          do  j1=2,m1j-1
            i1 = 2*j1-d1
            y2 = r(i1,  i2-1,i3-1) + r(i1,  i2-1,i3+1)
     >         + r(i1,  i2+1,i3-1) + r(i1,  i2+1,i3+1)
            x2 = r(i1,  i2-1,i3  ) + r(i1,  i2+1,i3  )
     >         + r(i1,  i2,  i3-1) + r(i1,  i2,  i3+1)
            s(j1,j2,j3) =
     >         0.5D0 * r(i1,i2,i3)
     >       + 0.25D0 * (r(i1-1,i2,i3) + r(i1+1,i2,i3) + x2)
     >       + 0.125D0 * ( x1(i1-1) + x1(i1+1) + y2)
     >       + 0.0625D0 * ( y1(i1-1) + y1(i1+1) )
          enddo
        enddo
        j = k-1
        call comm3(s,m1j,m2j,m3j,j)
        return
        end
```

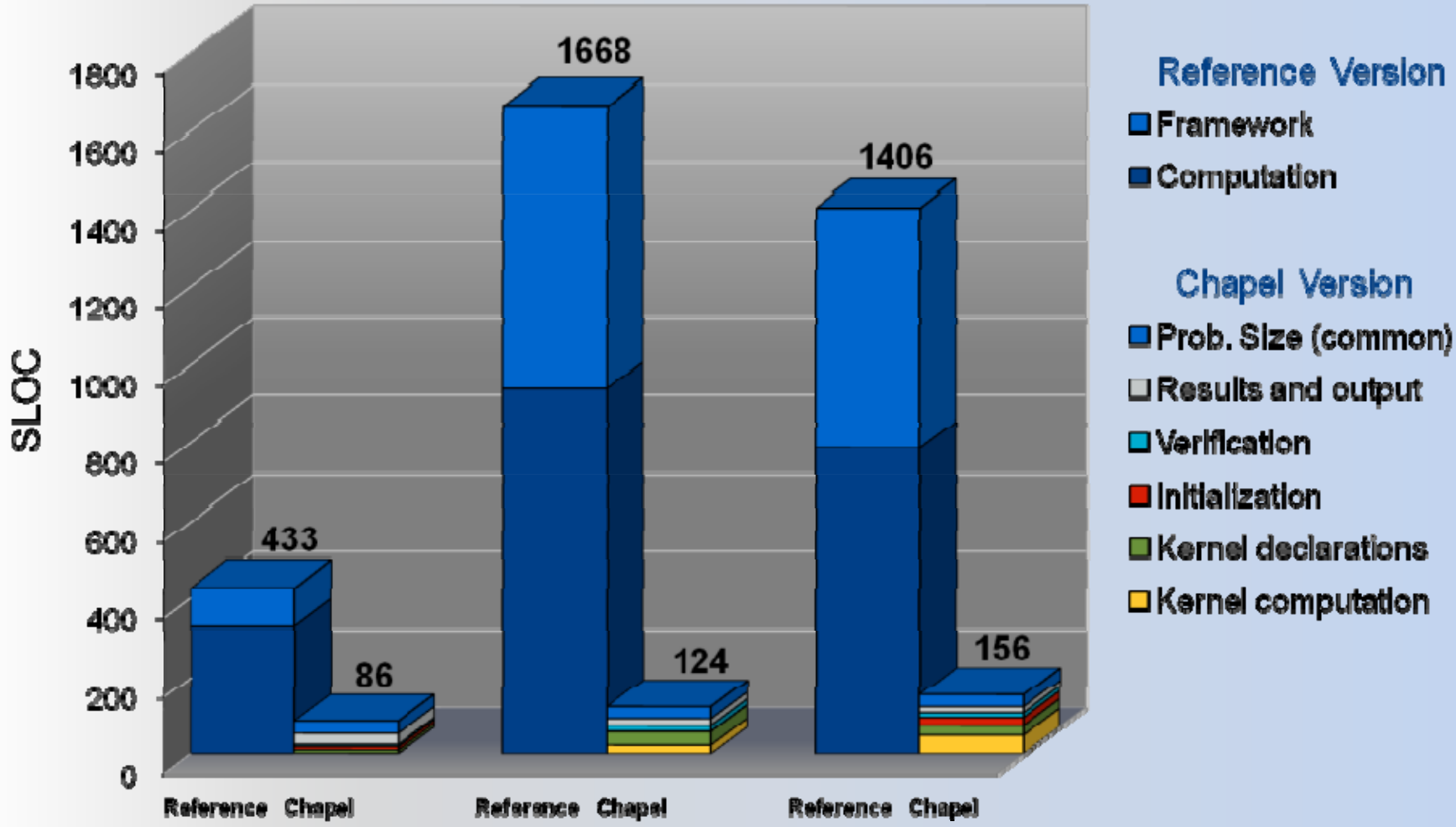
NAS MG *rprj3* Stencil in Chapel



```
def rprj3(S, R) {
  param Stencil = [-1..1, -1..1, -1..1],
        w: [0..3] real = (0.5, 0.25, 0.125, 0.0625),
        w3d = [(i,j,k) in Stencil] w((i!=0) + (j!=0) + (k!=0));

  forall ijk in S.domain do
    S(ijk) = + reduce [offset in Stencil]
                        (w3d(offset) * R(ijk + R.stride*offset));
}
```

Chapel Code Size Comparison

For HPC Challenge Benchmarks



STREAM Triad

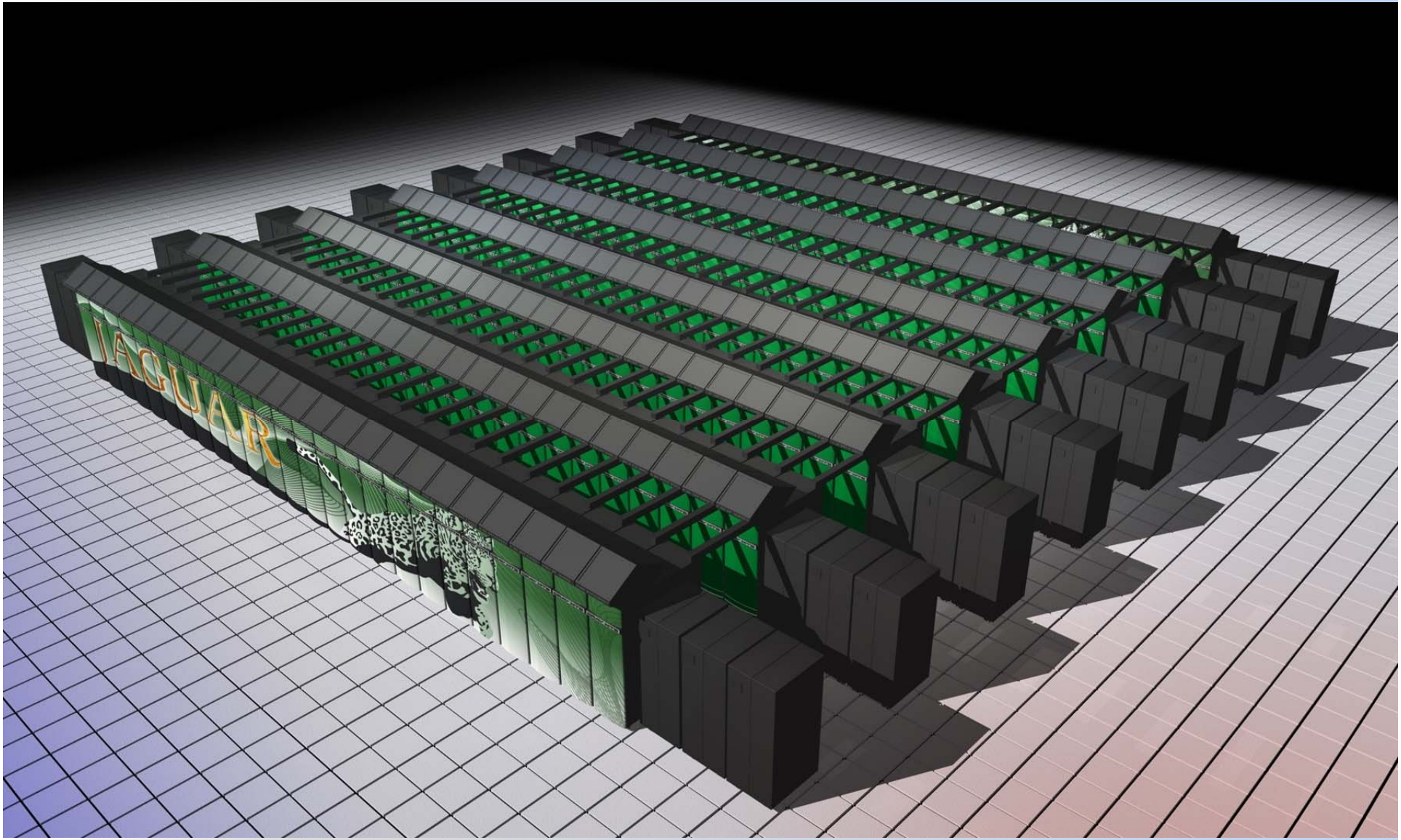
Random Access

FFT

Thank You!

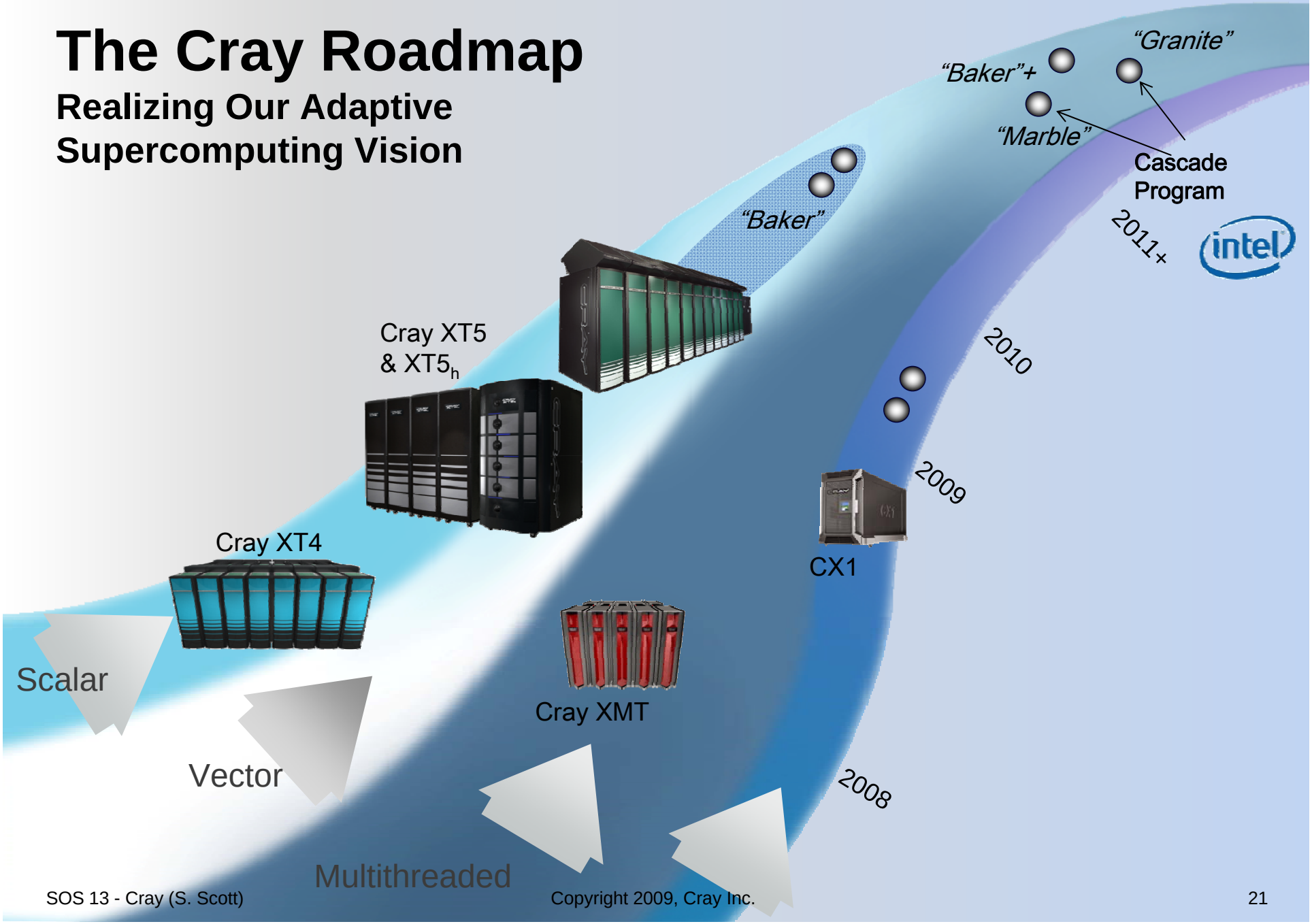
Questions?

ORNL Petaflop System



The Cray Roadmap

Realizing Our Adaptive Supercomputing Vision



Main Areas of Productivity Enhancement in Cray's Cascade System

- Compilers
  - Common user environment across Marble and Granite blades
  - Integrated support for CAF, UPC
  - Incremental compilation, runtime profiling, improved user feedback
  - Fully automatic multi-level parallelism (shared memory, multi-threading, vectorization)
- Programming Tools
  - Environment setup (modules)
  - Comparative debugging and dual-code debugging
  - Automatic performance analysis
- Scientific Libraries
  - Auto tuning (Cray Adaptive Sparse Kernels)
  - Adaptivity (Cray Adaptive FFT, Cray Adaptive Sparse Kernels)
- Programming Languages
  - Support for traditional languages
  - Integrated support for UPC, CAF, OpenMP
  - Chapel

Automatic Performance Analysis in Cascade

- Basic approach:
  - Intelligently collect and filter data
  - Distinguish between “similar” and “different” application behavior
  - Search data for inefficient execution patterns using performance models
- Automatically identify and expose performance anomalies
  - Load imbalance (MPI and OpenMP)
  - Communication / synchronization / I/O problems
  - Environment variables
  - Etc.
- Support includes:
  - Automatic profiling analysis
    - Automatically detects the most time consuming functions in the application
    - Feeds information back to the tool for further (focused) data collection
  - Recommendation infrastructure in CrayPat
    - E.g.: MPI rank placement suggestions (how ranks are mapped to cores)
    - Discrete Units of Help – visual performance hints to users
  - Scalable performance visualizer

Cray Adaptive Sparse Kernel

- The CASK Process
  1. Use a code generator to build all known variants of the algorithm for a set of optimizations (tens of thousands)
  2. Use extensive auto-tuned framework to benchmark performance for known matrix classes
  3. Use discrete optimization strategy to tune compile switches
  4. At runtime, analyze matrix to match to known category
  5. Use offline knowledge to assign a tuned kernel

- CASK will sit silently beneath PETSc and Trilinos

CRAFFT Library

- CRAFFT is designed with simple-to-use interfaces
  - Planning and execution stage can be combined into one subroutine call
  - Underneath the interfaces, CRAFFT calls the appropriate FFT kernel
- CRAFFT provides both offline and online tuning
  - Offline tuning
    - Which FFT kernel to use
    - Pre-computed PLANs for common-sized FFT
      - No expensive plan stages
  - Online tuning is performed as necessary at runtime as well
- At runtime, CRAFFT adaptively selects the best FFT kernel to use based on both offline and online testing (e.g. ACML, FFTW, Spiral, Custom FFT)