



Cray's Cascade Program

"We are almost at the top"

John Levesque

CTO Office

Director – Cray Supercomputing Center of Excellence





Are we acting a little like
Chicken Little?

Early 1970s – How are
We going to use 32KB
Of Memory

How are users going
To effectively use
Vectors

My God how are
We going to use
The CM

Any then came
MPPs

Isn't this just the next step in the evolution of HPC?

■ Things have changed

- ✿ Only a few vendors have control over the compiler
- ✿ Only a few vendors have control over the OS

■ Lets Vote

- ✿ A miscellaneous collection of Open Source Software will solve the productivity problem
 - ✿ A single vendors will be able to deliver the necessary software to have all the pieces work together in a way that the user can be productive
 - ✿ None of the Above
- I'm not saying that "Open Software" is not good, We use a tremendous amount of Open Software; however, we integrate it with our other software.

And we have some really smart people out there

- Robert Harrison's Madness code completely hides communication under his computation which is achieving 30-40% of peak on the XT
- Thomas Schulthess' has designed an application that uses Monte Carlo sampling of 1000s of smaller simulations that uses Cray's "Iterative Refinement Toolkit" and may achieve >100% of peak.
- Phil Jones has written POP to perfectly use Hybrid computing allowing the user to choose the number of blocks on the node, which can be equal to the number of MPI tasks on the node or number of threads on the node
- GTC is putting in more physics and can use as many flops as the designers can deliver
- S3D is putting in more chemistry into their combustion code and they are scaling perfectly on the XT to 32,000 cores
- Many of these applications (Not POP) can use "weak" scaling to extend to Millions of cores

And we are leaving a lot of performance on the table right now

- Very few people are doing blocking for Cache
- A majority of people have forgotten how to vectorize code
- A majority of people do not pre-post their MPI messages
- No one is really doing multi-threading in real applications
- Many people have C++ applications that get < 1% of peak
 - ✿ No C++ or Java compiler will ever vectorize the code.
 - ▶ Many have written kernels in Fortran

High Productivity Computing Systems

Goals:

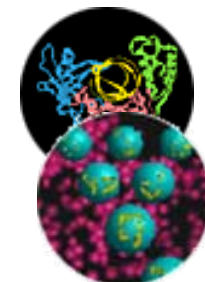
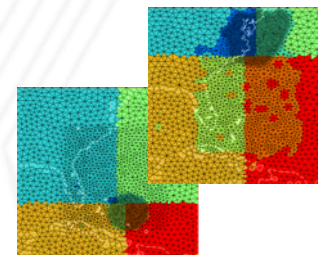
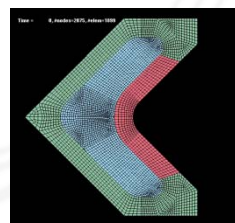
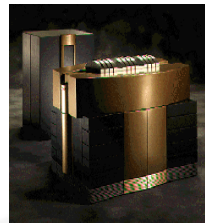
- Provide a new generation of economically viable high productivity computing systems for the national security and industrial user community (2007 – 2010)

Impact:

- Performance (efficiency): critical national security applications by a factor of 10X to 40X
- Productivity (time-to-solution)
- Portability (transparency): insulate research and operational application software from system
- Robustness (reliability): apply all known techniques to protect against outside attacks, hardware faults, & programming errors



HPCS Program Focus Areas



Applications:

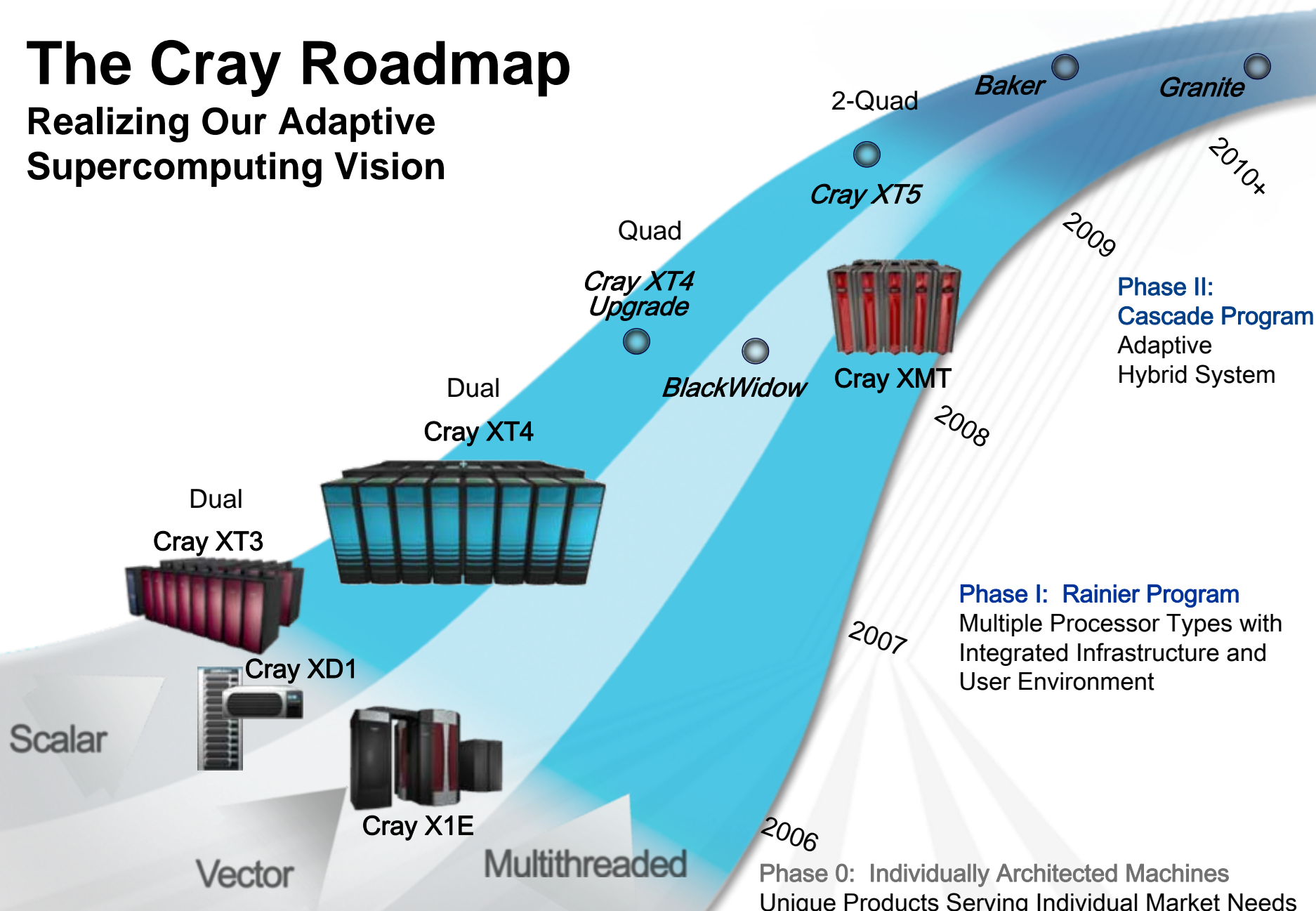
- Intelligence/surveillance, reconnaissance, cryptanalysis, weapons analysis, airborne contaminant modeling and biotechnology

Fill the Critical Technology and Capability Gap

Today (late 80's HPC technology).....to.....Future (Quantum/Bio Computing)

The Cray Roadmap

Realizing Our Adaptive Supercomputing Vision



Scalar

Vector

Multithreaded



Quad
Cray XT4 Upgrade



BlackWidow



Cray XMT

2-Quad



Cray XT5

Baker



Granite



2009

Phase II: Cascade Program
Adaptive Hybrid System

2010+

2007

Phase I: Rainier Program
Multiple Processor Types with Integrated Infrastructure and User Environment

2006

Phase 0: Individually Architected Machines
Unique Products Serving Individual Market Needs

Example: 6 Cabinet Cray XT5 System

SPECIFICATIONS

Compute cabinets: 6 (18 chassis)

Processors: 1112

Peak: 43 Tflops

Memory: 8.5-17 TBytes

Topology: 6 x 12 x 8

Floor space: 7 Sq Meters

System power: ~250 kW

Power and floor space do not include
IO & storage units



=



!

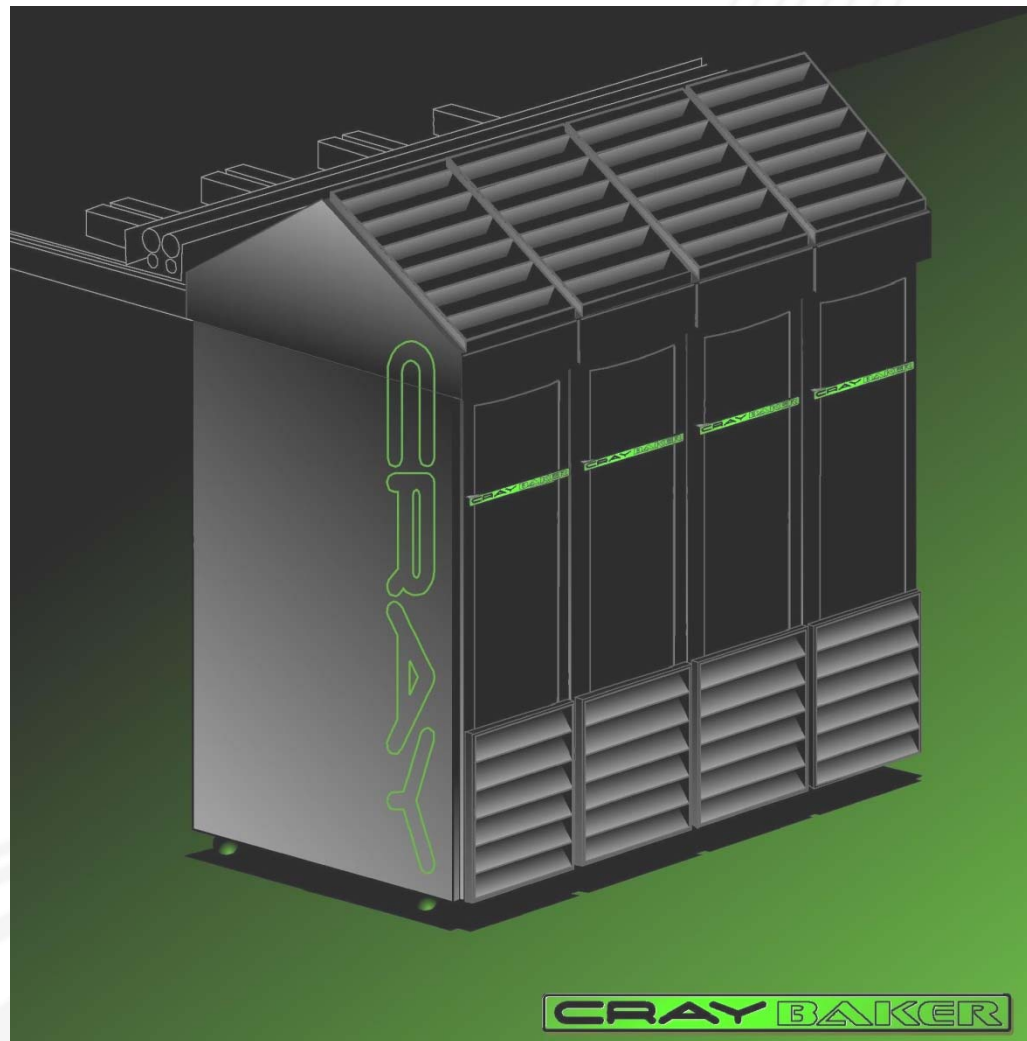
Baker

- Recently, some changes were made within the Baker program
- Baker is now compatible with the Cray XT packaging
- This eliminated a strong discontinuity in our product roadmap and allows for key technologies to be introduced separately (and earlier)
- Three new pieces of technology will be delivered
 - 1) **The new “High Efficiency” Baker cabinet with liquid cooling option**
 - 2) **The Baker G3 Compute Blade**
 - 3) **The Gemini Mezzanine Card and interconnect**



Cray HE Cabinet – 4Q 08

- Capable of removing 60kW at 20C ambient air temps
- Can operate with 18C (65F) water
- Liquid-Cooled kit is an option and can be retrofitted in the field
- HEUs can be interleaved to provide some redundancy



The Road to Cascade

	2004	2007	2009	Cascade
Processor	Single Core	Dual Core	Quad Core	More Core
# cores	5000	20000	50000	Hybrid
Interconnect	Seastar 1	Seastar 2	Seastar 2+	Gemini/Aries
Compiler	PGI	PGI/Pathscale	PGI/Pathscale	Cray/PGI/Path
Programming	SSE	SSE	SSE	SSE/Vector/MT
Communication	MPI	MPI/OpenMP	MPI/OpenMP	MPI/PGAS/OMP SHMEM

Hardware changes

- Marble/XT/Baker - Multi-core chips of 2010
- Granite
 - ⚙ Vector hardware to generate more flops/clock
 - ⚙ Multi-threading
 - ⚙ Addressing hardware to improve indirect addressing performance and global memory accesses
 - ⚙ Special instructions to do special things
- Inter-connect
 - ⚙ Lower latency (~1 microsecond)
 - ⚙ Higher bandwidth
 - ⚙ Synchronization mechanisms, including atomic memory ops, barriers and collectives
 - ⚙ Ability to GET from and PUT to remote memory without interrupting remote processor
- Heterogeneous
 - ⚙ Baker is the same proven XT technology MPP system with improved inter-connect
 - ⚙ Ability to strictly use Granite
 - ⚙ Ability to combine XT and Granite

Multi-cores

- Everything has already been said before this talk
 - ⚙ Pressure on Injection bandwidth
 - ⚙ Pressure on memory bandwidth
 - ⚙ Pressure on the USER
- When does Hybrid help?

But it hasn't – no one has really talked about the Interconnect impact of increasing the FLOP rate on the node

Impact of multi-core on inter-connect

- There is the normal bytes/flops of interconnect bandwidth
 - ⚙ However – Shouldn't it be Bytes/"Sustained Flops"
- The real issue is number of messages per second that the interconnect can handle.
 - ⚙ On XT5, the global sum is not longer the limiting factor on POP performance.
 - ▶ It is the number of messages coming off the 8 cores within the node during the halo update.
 - ⚙ One of the biggest problems with MPI is the message matching required to do the handshake between the processors
 - ⚙ OpenMP could really help this situation

Granite

- Can't really say anything definitive yet
 - ✿ Will be lots of vectors and/or threads
 - ✿ Compiler will systematically generate efficient code
 - ▶ Systematic means “With Directives and/or Profile Feedback”
 - ✿ Has to handle indirect addressing efficiently
 - ✿ Has to have reasonable Byte/flop ratio to memory
 - ✿ Has to have reasonable Byte/flop ratio to interconnect

Look at last year's talk

Interconnect

- In the next generation interconnect we will finally have the T3F
 - ✿ Excellent MPI machine with 100 times the MPI messages/second than on Seastar
 - ✿ Excellent PGAS machine with hardware assist for remote GETS/PUTS
 - ✿ Excellent Global Arrays machine
 - ▶ Given this feature, the Chemistry people will totally saturate the system

Finally Co-arrays and UPC to the rescue

■ How does Co-Arrays help

- ⚙️ Allows for lots of small messages in flight at the same time
- ⚙️ Applies to whatever memory is being addressed, either on-node or off node
 - ▶ Simplifies
 - Hybrid – Shared and distributed Parallel computing
 - » Its all shared
 - Heterogeneous processing
 - » It is so much easier to put into the remote processors memory than packing up messages and sending them and then unpacking messages at the receiver.

Pointers in Derived Types

```
TYPE P4
  integer len1
  real(REAL8),dimension(:), POINTER :: p_send_low
END TYPE P4

TYPE R4
  integer len2
  real(REAL8),dimension(:), POINTER :: p_send_scratch
END TYPE R4

TYPE S4
  integer len3
  integer, dimension(:),POINTER :: p_rsend_index
END TYPE S4

TYPE(P4) :: send_low[*]
TYPE(R4) :: send_scratch[*]
TYPE(S4) :: rsend_index[*]

! set Co- array pointer to location of output array
send_scratch%p_send_scratch => input(1:length)
rsend_index%p_rsend_index => send_index(1:length)
send_low%p_send_low => send_lo(0:maxpe)
```

Must Barrier before using pointer

And then use them

```
do while(all(alldone(1:recv_num)))
  do n=1,recv_num
    pe = recv_pe(n)
    if(send_ready(1,pe+1))then
      ll = rlow_send[pe+1]%p_rlow_send(mype+1)
      do l=recv_lo(n),recv_lo(n)+recv_length(n)-1
        rindex = rsend_index[pe+1]%p_rsend_index(ll)
        output(recv_index(l))=send_scratch[pe+1]%p_send_scratch(rindex)
        ll = ll + 1
      enddo ! l
      alldone(n)=.true.
      recv_ready(1,me)[pe+1]=.true
      send_ready(1,pe+1) = .false..
    endif
  enddo ! n
enddo ! while
```

Having a compiler that understands PGAS

- When the compiler understands PGAS
 - ✿ Can do pre-fetching of remote data
 - ✿ Can vectorized a fetch – to get/put a larger chunk of data
 - ✿ Will not be calling messaging interface like GASNET, do direct RMA through the interconnect
 - ▶ Either on node or off the node – user doesn't need to know
- Kathy's Yelick's examples would show a better improvement, if the compiler could do direct memory accesses

MPI is still going to be important

■ Latest version of MPI on the XT

- ✿ Able to have multiple transports
 - ▶ Transport to Gemini/Aries API
 - ▶ Transport to Shared Memory
 - Optimized to use memory copies
 - Collectives are optimized to use combined distributed/shared memory
- ✿ As number of cores get larger, Cray MPI will be able to effectively run MPI across all of the cores.
 - ▶ Will still be contention for interconnect bandwidth and message/second to be handled
 - ▶ MPI will have a more significant impact on memory utilization
 - Buffers mean more memory copies

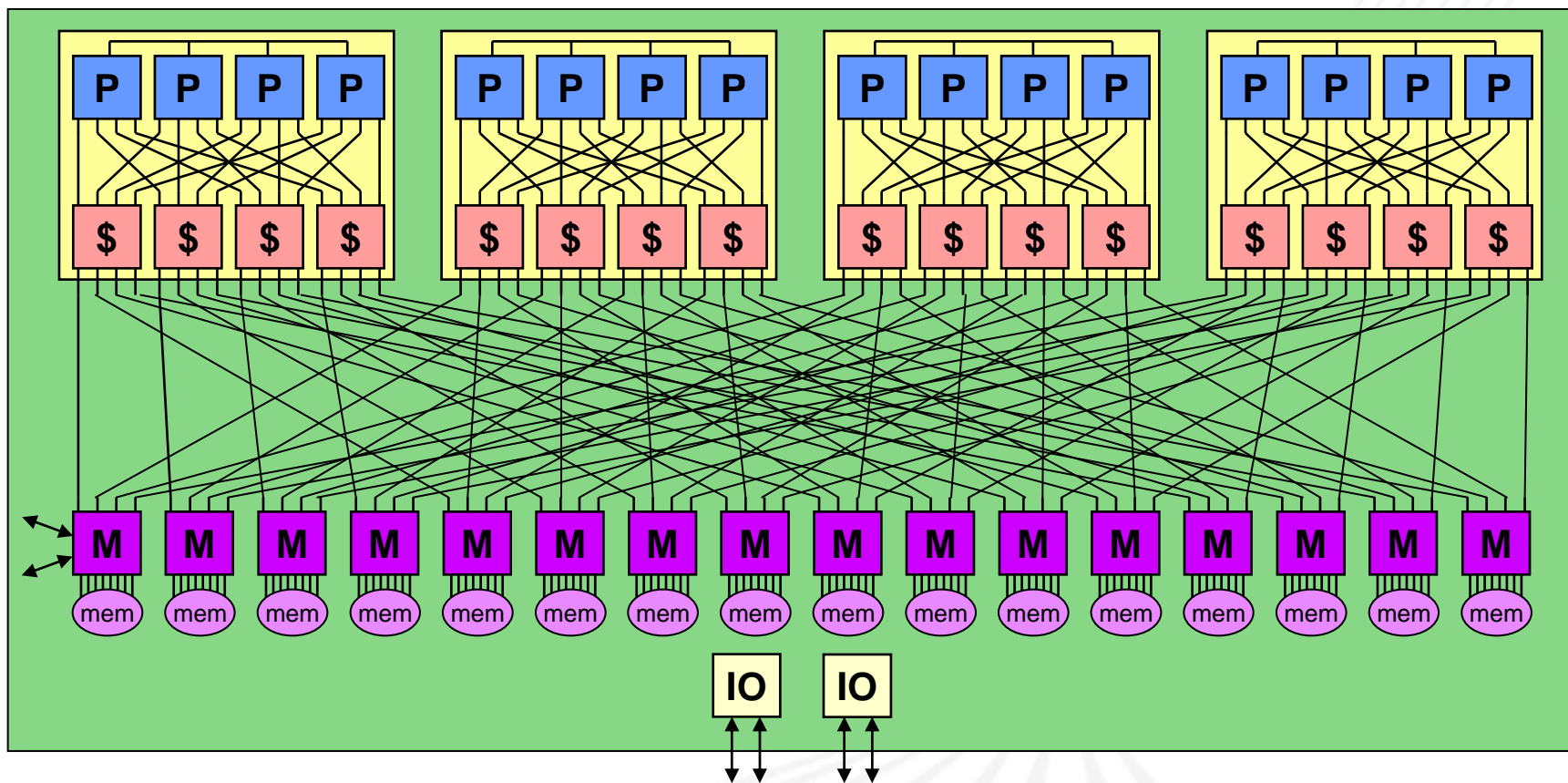
Cray's PDGCS Compiler

- Will be generating code for Granite
 - ✱ Best Vectorization in HPC
 - ✱ Best Automatic parallelization in HPC
 - ✱ Early results show very respectable scalar performance
- Coupled with the Cray Tools package to perform profile directed optimization
- Coupled with the debuggers to perform much better debugging
- Coupled with interconnect to perform the best Co-Arrays and UPC – no subroutine calls

And why is PDGCS so good?

An abstract graphic consisting of several thin, light gray lines that originate from the bottom right corner and converge towards the top right corner, creating a sense of depth and perspective. The lines are set against a white background with a green header bar at the top.

Cray X1 Node (a T932 on a board)



Inter node network:

2 ports per M chip
 1.6 GB/s full duplex per link

I/O connections:

4 ports per node
 1.2 GB/s full duplex per link

Local memory:

200 GB/s peak bw
 8-32 GB per node

Heterogeneous Computing

- You asked for it, now you will get it
 - ✿ Mixed nodes with multi-core (Marble) on some nodes and accelerators (Granite) on others
 - ▶ From a coupled application, different binaries for different nodes
 - ▶ From the same executable, using MPI and/or PGAS between disparate nodes
 - MPI, OpenMP, Vector and multi-threading on the XT multi-core
 - MPI, OpenMP, Vector and multi-threading on the Granite
 - Now combine the above two – wow what a programming nightmare

Motivation for Cascade

Why are HPC machines unproductive?

- Difficult to *write* parallel code
 - ⚙ Major burden for computational scientists
- Lack of programming tools to *understand* program behavior
 - ⚙ Conventional models break with scale and complexity
- Time spent trying to modify code to fit *machine's* characteristics
 - ⚙ For example, cluster machines have relatively low bandwidth between processors, and can't directly access global memory...
 - ⚙ As a result, programmers try hard to reduce communication, and have to bundle communication up in messages instead of simply accessing shared memory

*If the machine doesn't match your code's attributes,
it makes the programming job much more difficult.*

And code's vary significantly in their requirements...

Cray's two prong attack on Productivity

■ First Approach

- ✿ Investigate new languages
 - ▶ Chapel
 - ▶ We love CAF and UPC
- ✿ Investigate productivity infra-structure

■ Second Approach

- ✿ User does not want to program in a new language
 - ▶ Automatically show users the issues with their program and suggest optimization
- ✿ User does not want to learn new GUI
 - ▶ Automatically supply user with the information they need to restructure application

Cray's Profiling Tools

■ Profile Feedback

- User sets up test problem and submits to the Cray profiling tool
 - ▶ Results come back after several separate executions and displays
 - Top Bottleneck and remedy, if remedy exists
 - » Environment variable suggestions
 - » Mostly MPI settings
 - » TLB problems
 - » Where and why
 - » Striding problems
 - » Where and why
 - » Load Balancing problems
 - » Recommend layout to combine lightly used MPI tasks with heavily utilized MPI tasks on the same node
 - » Message passing statistics
 - » Latency or Bandwidth dominated

Cray's Profiling Tools

■ Automated Assistance for MPI

- ✿ Given message sizes, optimal setting for MPI tuning environment variables can be suggested and even generated
- ✿ Given load imbalance, optimal layout across cores on a node can be given
 - ▶ Actual node number file can be given to use as input to *aprun*
 - Combine highest compute tasks with lowest compute tasks to average out memory bandwidth usage
 - Combine highest messaging tasks with lowest to average out network bandwidth usage
- ✿ Given MPI data, information can to given for improving MPI performance
 - ▶ Are messages pre-posted?
 - ▶ MPI_ALLTOALL and MPI_ALLREDUCE
 - Identify areas in code where bottleneck occur

Cray's Profiling Tools

- Automated Assistance for code optimization
 - ✿ Hardware counters will supply
 - ▶ TLB information
 - ▶ Cache information
 - ▶ Vectorization information
 - ▶ Etc
 - ✿ Cray compiler can identify
 - ▶ Vector lengths
 - ▶ Parallel efficiency
 - ▶ Granularity of parallel region

Cray's Profiling Tools

■ Automated Assistance for Hybrid Computing

⚙️ Cray compiler can identify

- ▶ Vector lengths
- ▶ Parallel efficiency
- ▶ Granularity of parallel region

⚙️ Provide projections of benefit in using Granite

- ▶ Given details in performance of both nodes and an understanding of the computational characteristics of the application, the profiling tool can estimate the benefit of moving portions of the application to Granite

⚙️ What we cant do right now

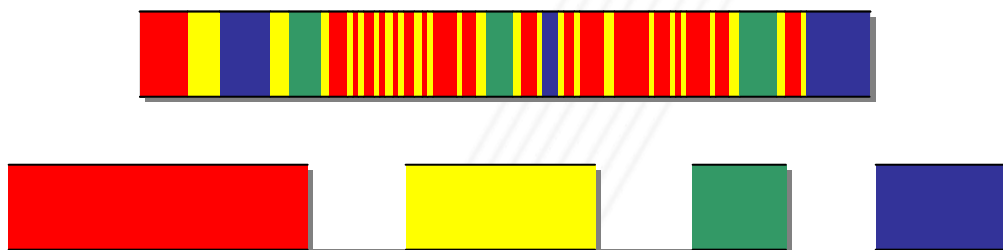
- ▶ Identify Data Structure usage between Granite and the MPP

Performance Estimates

Overview of Cray's Current Process

- For estimations for future Cray XT Opteron based systems, we currently gather single core, dual core, Craypat data for each benchmark
- This data allows the separation of an MPI code into its constituent performance components

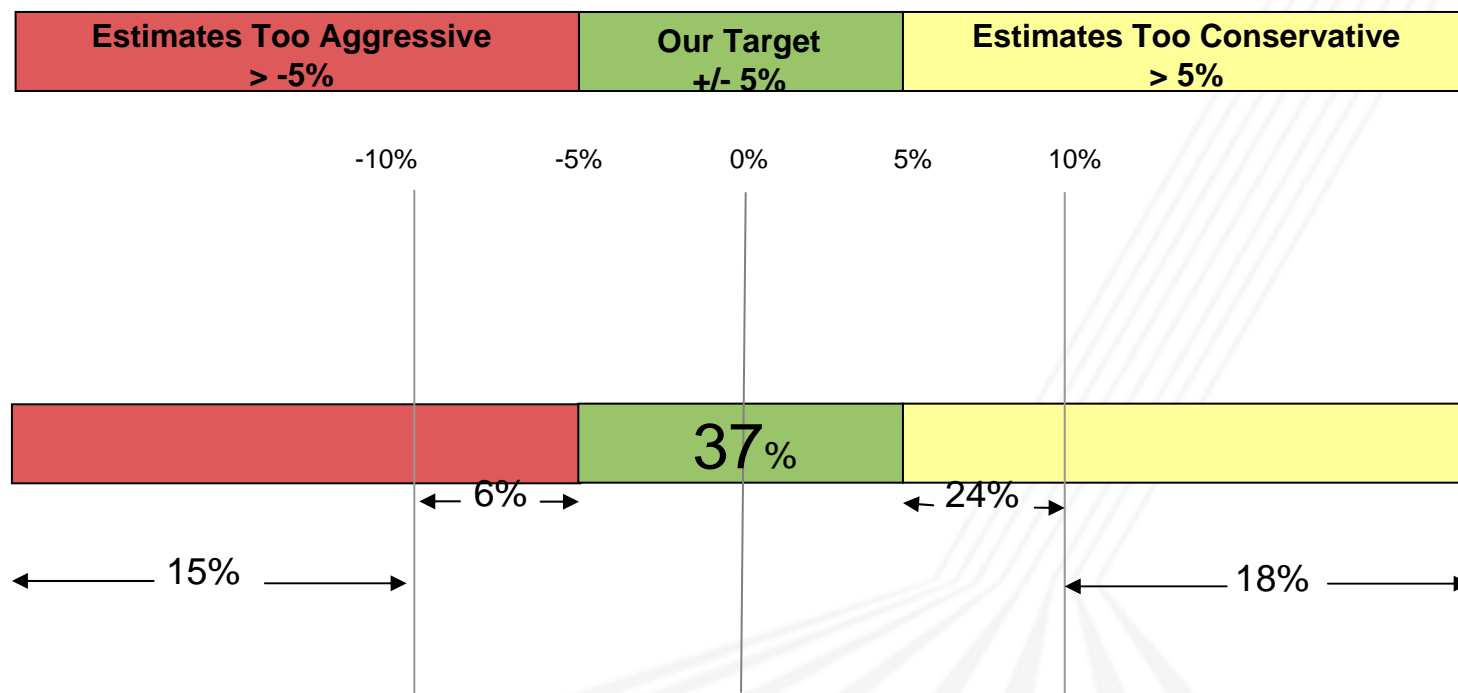
- Clock-limited computation
- Memory-limited computation
- Communication
- IO



- That data is then fed into existing empirical models to estimate future system performance
- To ensure empirical model accuracy, all results are reviewed by relevant application/benchmark engineers
- Goal is to be within $\pm 5\%$ of actual performance on the system at time of installation – and we are continually working to improve on this target.

$$t_{\text{tot}} = t_{\text{comm}} + t_{\text{compute}} + t_{\text{io}} + t_{\text{oh}}$$

2007 Estimation Accuracy Summary - *Initial* Runs on Targeted Platform



Initial Estimation Accuracy: Estimates compared to Initial Results
on Actual Hardware

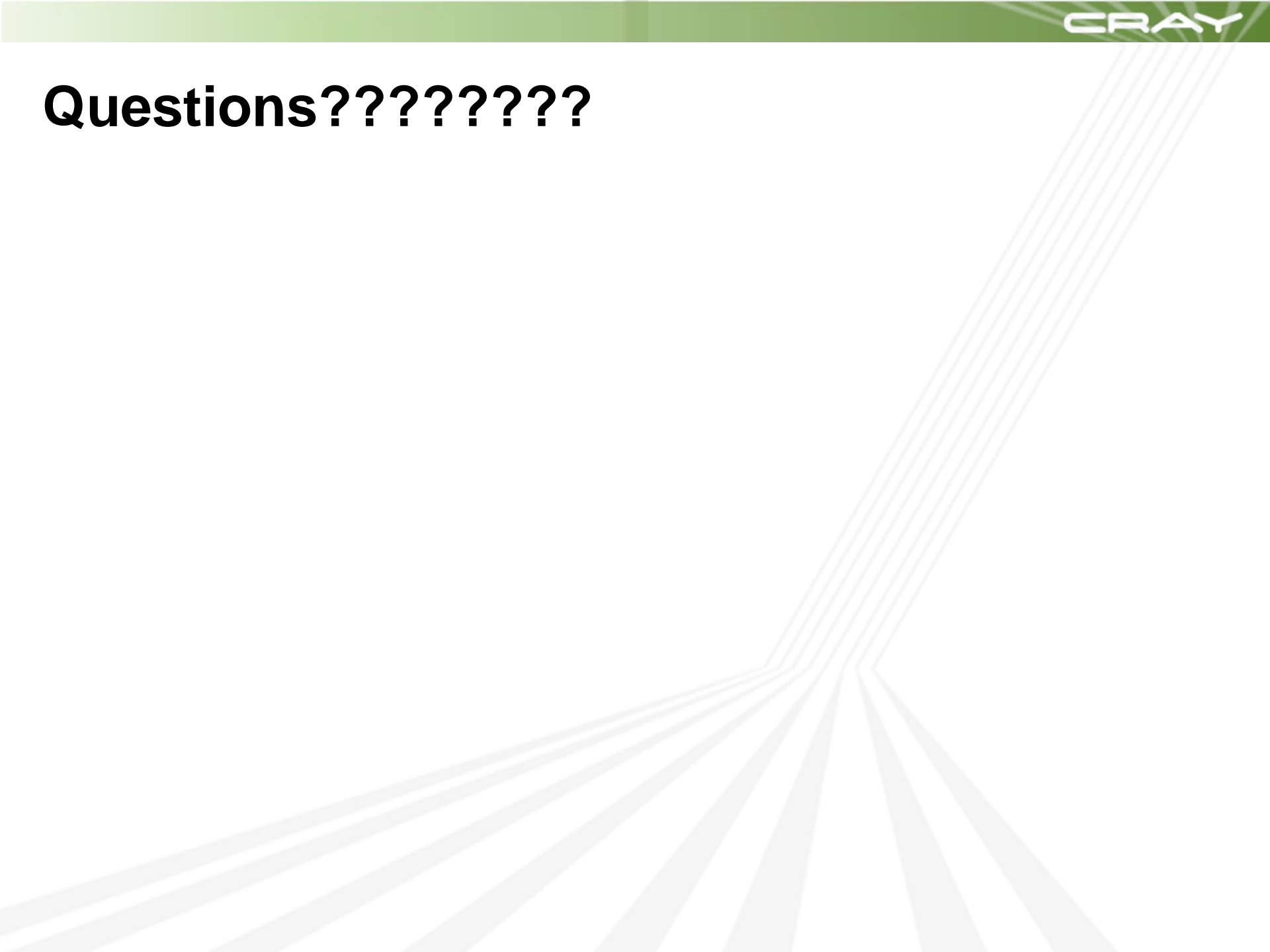
Estimations done in 2006 and then run on actual HW when introduced in 2007

And now do this “Automagically”

- Craypat profiling tools has all the information
 - ✿ With database information on the performance details of the MPP node and Granite node
 - ▶ Can approximate the performance on Granite versus the multi-core node
 - ▶ Can show user sections of the application that will benefit from moving to Granite
- What Craypat can't do
 - ✿ Identify data communication necessary for using Granite
 - ✿ Do the necessary restructuring to separate the two sections of code

After all none of us believe in
AUTOMAGIC

Questions????????

An abstract graphic consisting of several thin, light gray lines that originate from the bottom right corner and converge towards the top right corner, creating a sense of depth and perspective. The lines are set against a white background with a green header bar at the top.