“The Advancement or Lack of Advancement in Programming Models Over the Past 45 Years”

Aug 14, 2012

John M Levesque
Director of Cray’s Supercomputing Center of Excellence
CTO Office

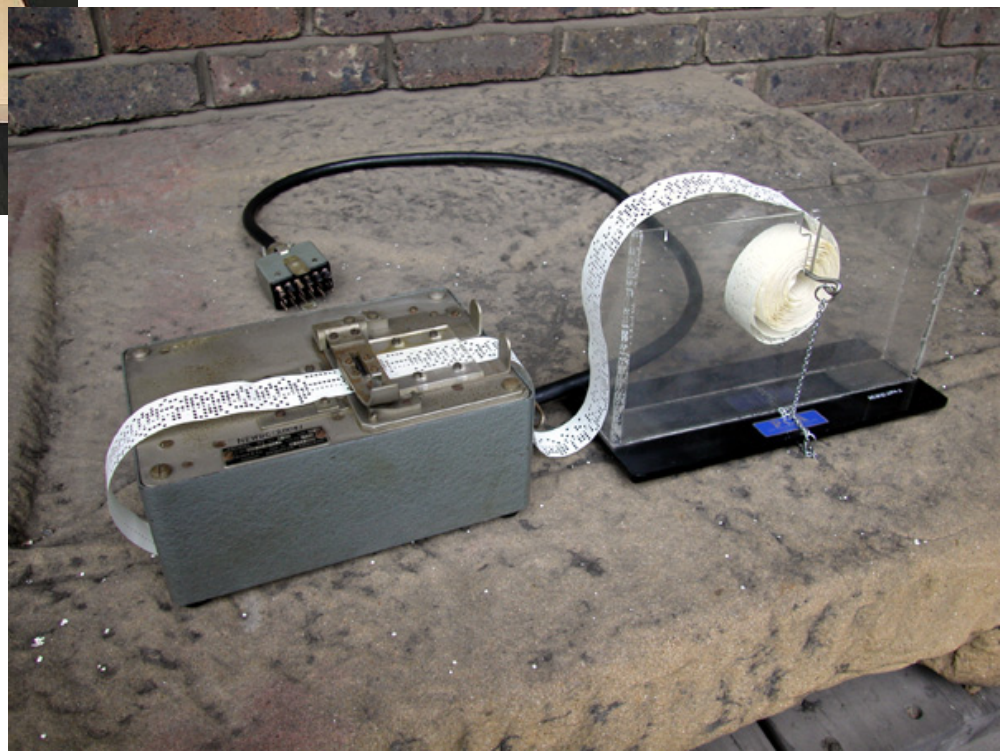
First a confession

I have never written an application from scratch, everything I have done starts with existing applications which I restructure to run faster.

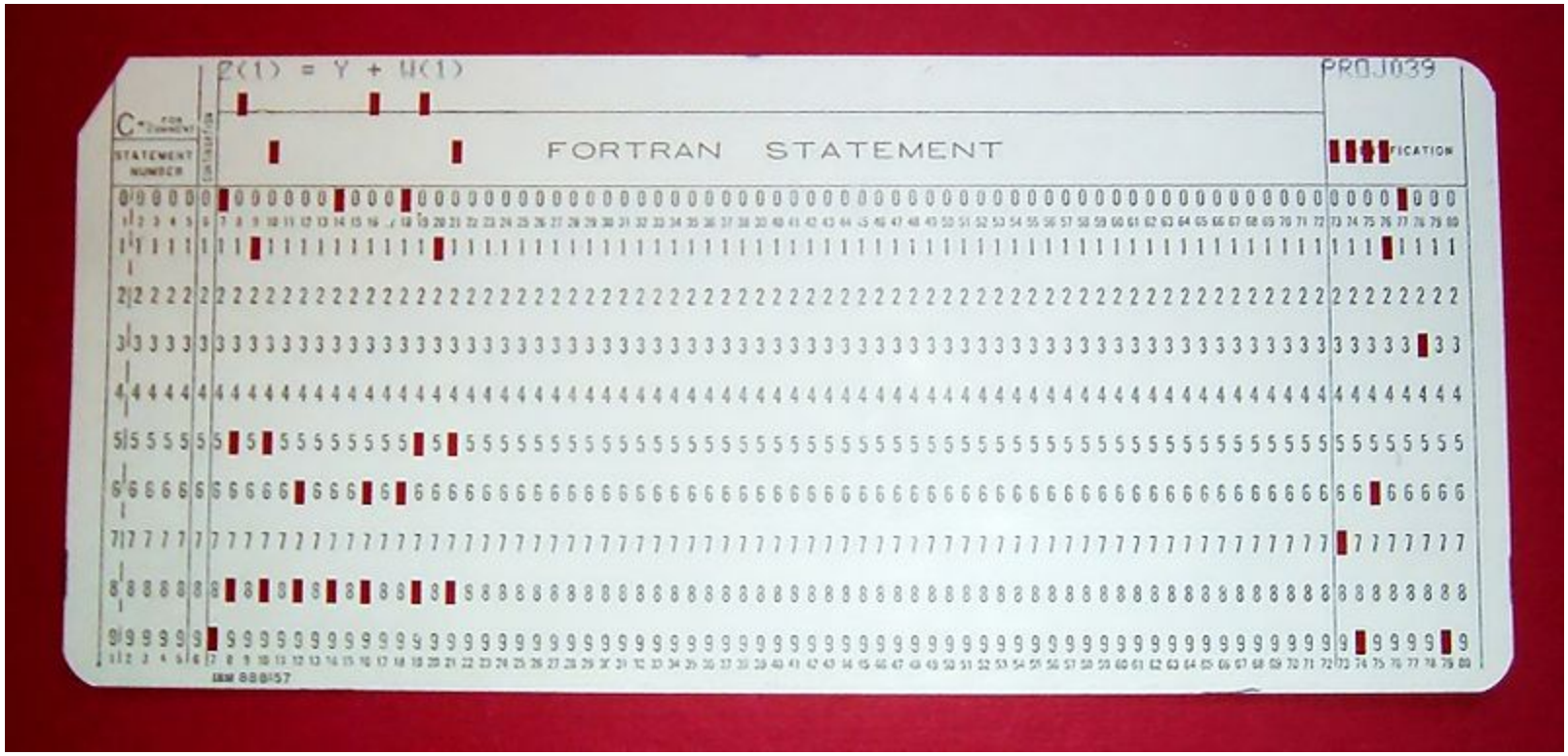
One might say that I am one of the reasons some legacy applications are still around.

What Supercomputer Was this?

What are these?



Why was the Fortran line length 72?



Computational Steering in the 70s

Set a sense switch, dump the velocity Vector field to tape, take tape to Calcomp Plotter.

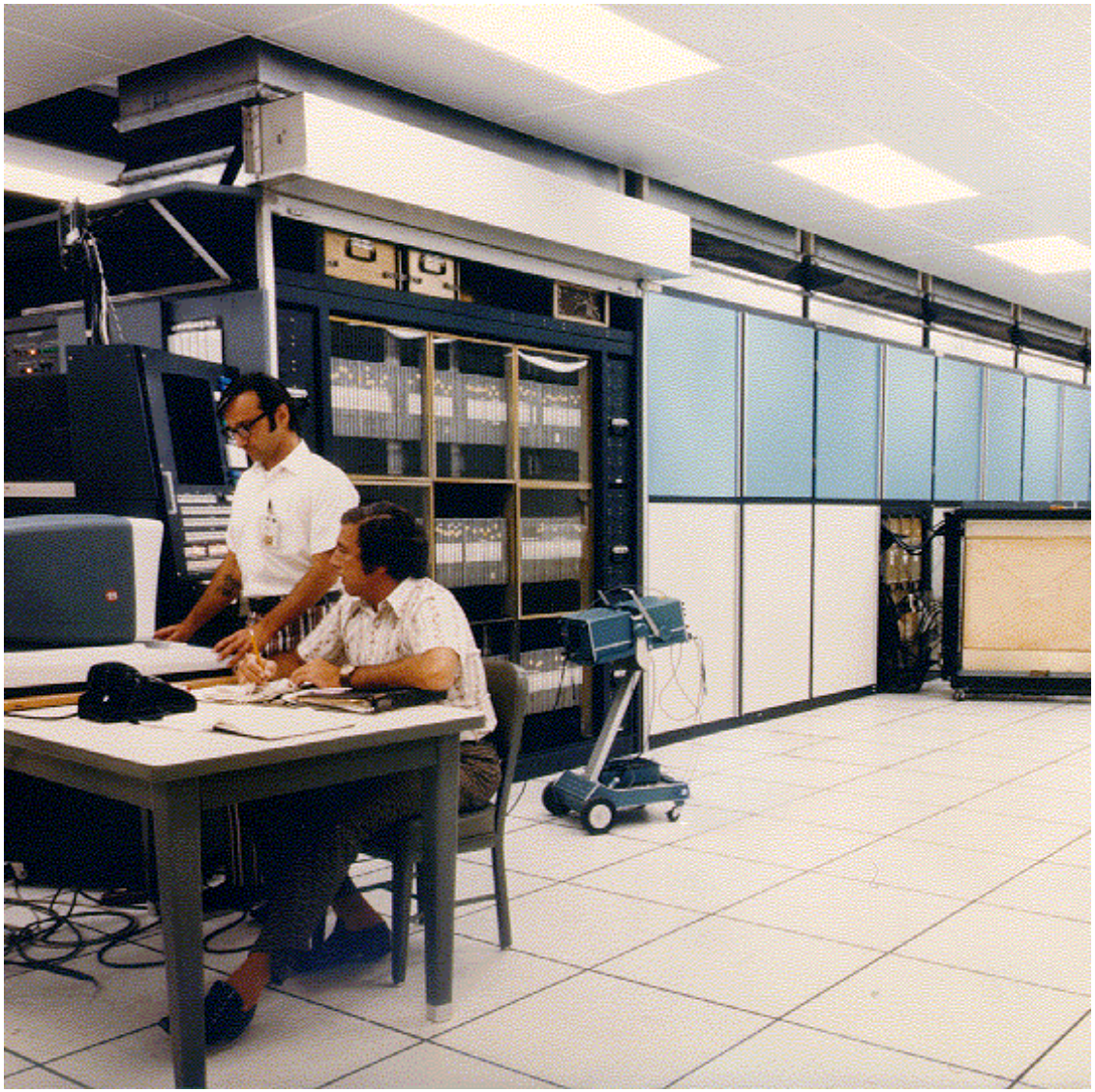
A look at the past

- **1969-1972 Air Force Weapons Laboratory**
  - CDC 6600
    - 350K 60 bit words
    - Multiple functional units
    - Computer decks and magnetic tapes
    - Magnetic tapes and calcomp printers
    - TOODY ➔ SPEED
    - My first chance at being a Millionaire
    - EFMA award
      - What is the difference between IV and VI?
      - Boxes of cards
    - Super Index

Things from the past that can help today

- **Super Index**
  - Shows usage of all variables throughout the program

What Supercomputer is This?



Anyone know why the door is open?

Who Invented the Internet?

Another Seymour Masterpiece

How much memory did this system have?

A look at the past

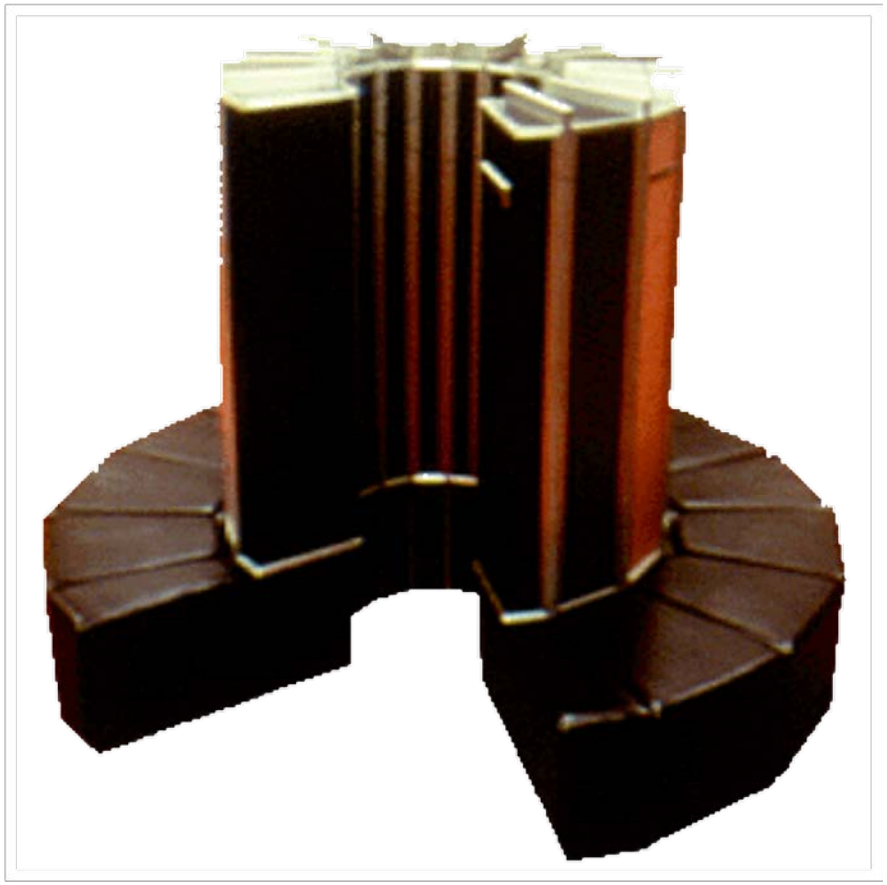
- **1972-1978 (R&D Associates)**
  - Illiac IV SIMD Parallel Processor
    - 64 SIMD processors
    - Ivtran, CFD, Paralyzer
    - Arpanet
  - CDC 7600 Segmented Functional Units
    - Stacklib and RDAlib
    - Vectorizer
  - Star 100 Memory to Memory Vector Processor

What we learned from these Systems

- **Both Illiac IV and Star 100 had to have a very high percentage SIMD, their scalar was very slow**
  - AMDAHL challenges exist today
- **Illiac IV had two Fortran like language extensions to address the SIMD instructions – IVTran and CFD**
- **Illiac IV had one of the first if not the first automatic compilers called the Paralyzer.**
- **Star 100 required extensive usage of semi-colon notation and Q8inlines to get to the power of the hardware.**
- **CDC 7600 did not have hardware vector instructions; however, it had segmented functional units**
  - Software pipelining is very effectives for segmented functional units
- **CDC 7600 had a very small 32K words of memory and a larger 512K words large core memory.**
  - Memory management was very important for porting larger applications on to the CDC 7600

Things from the past that can help today

- Super Index
  - Shows usage all variables throughout the program
- **Amdahl's law is extremely important today and will become more important as more parallelism is introduced and scalar performance is deminished**
  - Intel MIC good example today when used in standalone mode.
- **Memory management for CDC7600 SCM and LCM**



What Supercomputer is this?

A look at the past

- **1976-1979 R&D Associates**
  - Cray 1
    - Cray Vector Primitives (Vectorizer)
    - CFT -
      - automatic vectorization
      - comment line directives
    - Helping Big Oil move to the Cray
      - Using Illiac IV algorithm to vectorize reservoir model
      - Using CDC 7600 experience to handle data movement
  - Cyber 200's
    - Memory to Memory Vector Operations and Register to Register Vector Operations)
    - Q8inlines
    - Vectorizer

My first laptop

Programming for the Cray's

- **Cray vector primitives were used before the vectorizing compiler was available. July 1976**
  - User managed vector registers.
  - Met a young Jack Dongarra at LANL summer 1976
- **Birth of user directives Dec 1976 – CDIR$ IVDEP**
  - Why did Cray do this?
- **Birth of compiler diagnostics – why things were vectorized**
- **Birth of simple profiling - FLOWTRACE**
- **Scalar performance was better than CDC 7600 27.5 nsec cc versus 12.5 nsec.**
  - Vector added another factor 7-15 improvement
- **Memory Bank Conflicts – Don't dimension arrays multiple of 64**
- **Cray's 1 Mega Word memory was small compared to IBM's virtual memory**
  - Needed to use overlays, memory management

Door Prize for anyone guessing what this is.

What we learned from the Cray 1 and Cyber 200 series

- **Vectors are nice; however, a faster scalar processor is wonderful**
- **Vector Registers are very powerful, beneficial in reducing the pressure on the memory bandwidth**
- **Chaining, pipelining of vector instructions was able to keep vector units busy**
- **A good optimizing compiler is the best way to increase programmer productivity**
  - Comment line directives are an effective way for the user to communicate with the compiler

Things from the past that can help today

- **Super Index**
  - Shows usage all variables throughout the program
- **Amdahl's law is extremely important today and will become more important as more parallelism is introduced and scalar performance is deminished**
  - Intel MIC good example today when used in standalone mode.
- **Memory management for CDC7600 SCM and LCM**
- **How to vectorize code**
  - Most of you have forgot how to write vector code
    - "if the compiler doesn't vectorize it cannot be vectorized"
- **Comment line directives really work**

What Supercomputer is this?

During this decade,

More money was spent on Disposal Diapers than on Supercomputer

What Supercomputer is this?

A look at the past

- **1980-1992 Pacific Sierra Research**
  - Nec SX, Fujutsu VPP, Hitachi VP
    - Register to Register Vector Operations
    - Adjustable register size
    - VAST and FORGE
  - Cray 1s, Cray XMP, Cray YMP, Cray 2
    - Start of shared memory parallelization
      - Cray Micro tasking and Macro tasking
    - VAST and FORGE
  - Univac and Shell Oil
    - MIMDIZER
  - Intel Paragon, Ncube,…
    - HPF

What we learned from these systems

- **Distributed Memory MPP systems (DMP)**
  - This was undoubtedly the most difficult transition in supercomputer architectures.
  - The user had to decompose their problem and handle all communication during the execution of the applications
  - Many different message passing libraries, PVM, Intel's NX, Ncube
  - Porting non DMP applications to DMP required extensive memory analysis which was extremely difficult without tools.
- **Shared Memory Parallelization on the Big Vector Iron**
  - Cray's Micro-tasking and Macro-tasking
  - User's directives always better than Automatic Parallelization

Why HPF failed

- **User expectations were too high**
  - In order for a new language to succeed, there has to be a large group of users who are willing to work with the language and compiler developers to allow time for issues to be resolved.
- **HPF was not incremental as one would like**
  - Except in very special cases incremental implementation of HPF was sure to run slow
- **Many of the HPF features were a challenge to implement efficiently**
- **MPI became the preferred mean of developing distributed memory programs**

Things from the past that can help today

- Super Index
  - Shows usage all variables throughout the program
- Amdahl's law is extremely important today and will become more important as more parallelism is introduced and scalar performance is deminished
  - Intel MIC good example today when used in standalone mode.
- Memory management for CDC7600 SCM and LCM
- Memory analysis for MPP
- How to vectorize code
  - Most of you have forgot how to write vector code
    - "if the compiler doesn't vectorize it cannot be vectorized"
- Comment line directives really work
- Compilers are limited in what they can do automatically

Famous Jurassic Park Prop

A look at the past

- **1992-1998 Applied Parallel Research**
  - Intel Paragon, Ncude, Cray T3D,T3E
    - Massively Parallel Progessors
    - PVM, NX
    - Parallel super scalar systems
    - MIMDIZER, HPF
    - F– and Co-Array Fortran
  - Connection Machine
    - CMAX – Fortran 77 – CM Fortran

What we learned from MPP systems MIMD and SIMD

- Very difficult to utilize
- Most applications were totally rewritten at this time
- Compilers and tools were not helpful, programmers had to supply all the information about the application.
- Attempts at making DMP easier
  - HPF
  - MIMDizer
  - FORGE
- Automatic Compilation for DMP was doomed to failure

A look at the past

- **1998-2001 IBM Research – Start of CoDesign**
  - IBM SP
    - MPI
    - Conversion of T3E to IBM SP
    - Conversion of CM Fortran to IBM SP

Advanced Computing Technology Center (ACTC)

- **Established within IBM Research**
  - Work with potential customers of IBM SP systems to port and optimized they applications
  - Funded by Peter Ungaro
  - Developed proto-type software, primarily tools and libraries to assist transition from Cray T3E to IBM SP
    - Not appreciated by Peg Williams
- **Working with customers and potential customers to optimize applications was all fun and beneficial to the company**

Things from the past that can help today

- Super Index
  - Shows usage all variables throughout the program
- Amdahl's law is extremely important today and will become more important as more parallelism is introduced and scalar performance is deminished
  - Intel MIC good example today when used in standalone mode.
- Memory management for CDC7600 SCM and LCM
- Memory analysis for MPP
- How to vectorize code
  - Most of you have forgot how to write vector code
    - "if the compiler doesn't vectorize it cannot be vectorized"
- Comment line directives really work
- Compilers are limited in what they can do
- Computer Vendors working closely with users results in better application performance

The best Co-Array machine

A look at the past

- **2001-2010 Cray Inc**
  - Joined Cray and started working on the SV1 and SV2 (X1 and X1E)
  - Volunteer to head Cray Supercomputing Center of Excellence at Oak Ridge National Laboratory (2003-present
    - Work with DoE researchers to port and optimize their application for the
      - X1 and X1E
      - XT3, XT4, XT5, XE6
    - Were able to have major applications available to do day one science immediately after acceptance

What we learned from the X1 and X1E

- **If you squint your eyes , the X1 and X1E architectures looks a lot like the Nvidia Fermi accelerator**
  - X1-X1E 16-32 parallel units with a vector unit
  - Fermi 16 parallel units with SIMT units
- **Hardware gather/scatter significantly helps with vector performance**
- **Shared Memory parallelism as well as high speed interconnect required good memory alignment**

Things from the past that can help today

- Super Index
  - Shows usage all variables throughout the program
- Amdahl's law is extremely important today and will become more important as more parallelism is introduced and scalar performance is deminished
  - Intel MIC good example today when used in standalone mode.
- Memory management for CDC7600 SCM and LCM
- Memory analysis for MPP
- How to vectorize code
  - Most of you have forgot how to write vector code
    - "if the compiler doesn't vectorize it cannot be vectorized"
- Comment line directives really work
- Compilers are limited in what they can do
- Computer Vendors working closely with users results in better application performance
- Specialized Hardware development for a small company is not good business

The System that shot down a satellite

A look at the past

- **2003-2010 Cray Inc**
  - Red Storm and XT
    - Highly scalable MPP system
    - Multi-core nodes
      - From 1-2-4-8-16-32
  - Continuation of Cray CoE

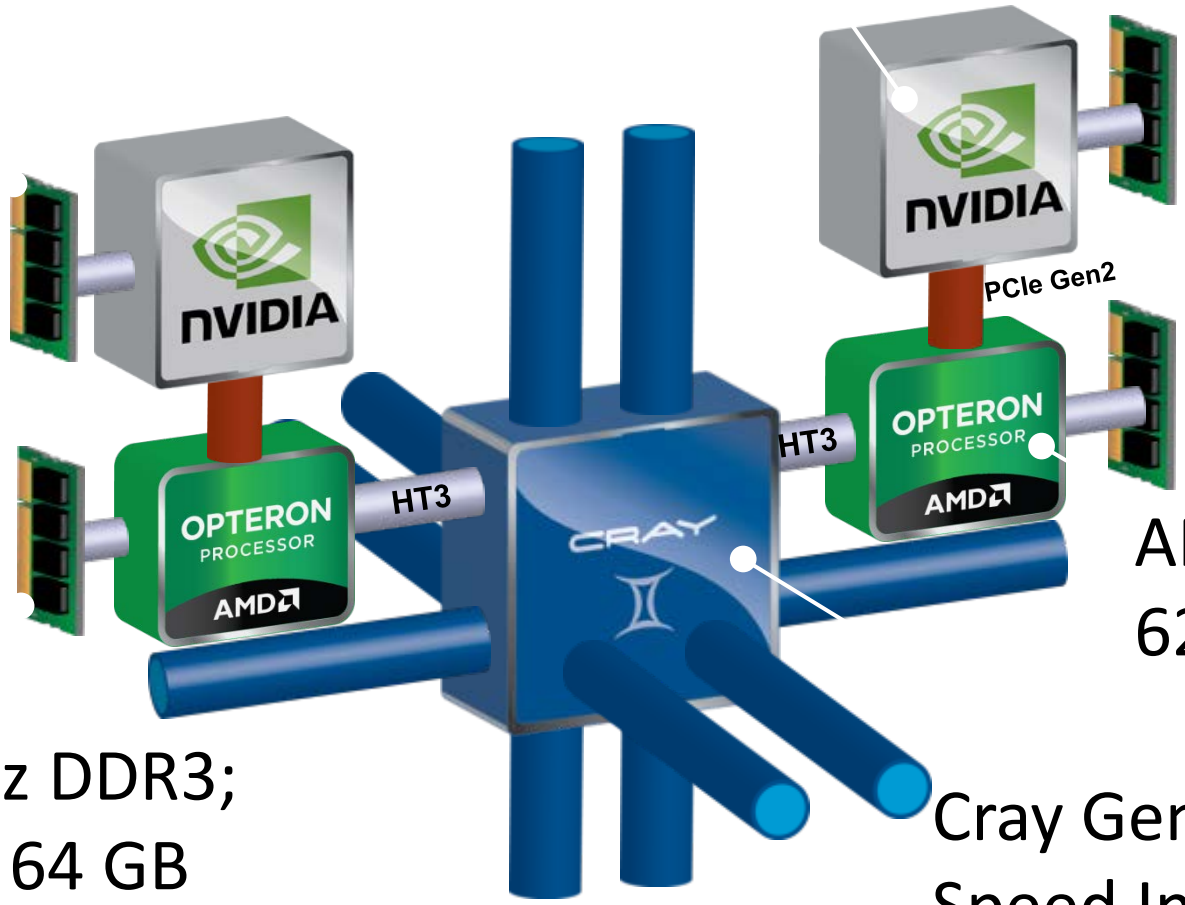
What we learned from Red Storm

- **Light weight Operating System was required to scale to hundreds of thousand of nodes**
- **Upgradability is wonderful**
- **Moving to higher core counts does not mean that core to core you will get better performance, need to start looking at node to node.**
- **On Multi-core Systems**
  - Scaling to higher core counts
    - Weak Scaling
    - Operating System Jitter
    - Network Topology and link saturation
  - Dealing with higher cores/node
    - Messages/second handled by the NIC
    - Sharing Injection bandwidth

Our Current Challenge

NVIDIA Tesla GPU
with 665GF DPFP

6GB GDDR5;
138 GB/s



AMD Series
6200 CPU

1600 MHz DDR3;
16, 32 or 64 GB

Cray Gemini High
Speed Interconnect

A look at the past

- **2010-Present Cray Inc**
  - XK6
  - Hybrid multi-core
    - AMD Interlagos with Nvidia Fermi/Kepler
  - Nvidia's Cuda
  - PGIs Cuda-Fortran
  - OpenMP Extensions for Accelerators
  - OpenACC from PGI, CAPS and Cray Inc

Why OpenACC will succeed

- **Comment Line directives**
  - Fortran, C and C++
- **It is incremental**
- **Can generate performance portable application that can run on existing multi-core MPPs with or without an accelerator**
  - Can run in offload mode or standalone mode for the MIC
- **Compilers are good at giving feedback on bottlenecks**
- **Big applications have been successfully ported and will be used to demonstrate how one can use OpenACC to utilize Hybrid Supercomputers.**

Things from the past that are helping today

- **Super Index – option on Cray Compiler**
  - Shows usage all variables throughout the program
- **Amdahl's law is extremely important today and will become more important as more parallelism is introduced and scalar performance is deminished**
  - Intel MIC good example today when used in standalone mode.
- **Memory management for CDC7600 SCM and LCM**
- **Memory analysis for MPP**
- **How to vectorize code**
  - Most of you have forgot how to write vector code
    - "if the compiler doesn't vectorize it cannot be vectorized"
- **Comment line directives really work**
- **Compilers are limited in what they can do**
- **Computer Vendors working closely with users results in better application performance**
- **Specialized Hardware development for a small company is not good business**

So what are the lessons learned that are applicable to the future

- **Vectorization is very important and difficult to do**
  - For the past ten years, vectorization has not been important – it is now on every system out there
- **Memory management is very important and difficult to do**
  - To some degree, understanding array usage throughout the program is a requirement for utilizing accelerators as it was for MPI programming
  - Looking forward we will probably have two separate memories on all systems, a large slow memory and a small fast memory – sounds a lot like the CDC 7600 SCM and LCM
- **Tools are important and have not kept up with the hardware**
  - We desperately need a memory tools that shows all the places in a program where a memory range is utilized.
  - We have so many degrees of parallelism, it is difficult to measure all at once
- **Compilers are not omnipotent, they need programmer help**
  - Comment line directives have been used successful for the past 35 years
  - More sophisticated feedback from the compiler would be helpful

Some appropriate Quotes

It's hardware that makes a machine fast. It's software that makes a fast machine slow.
Craig Bruce

The trouble with programmers is that you can never tell what a programmer is doing until it's too late."
- *Seymour Cray*

"Program testing can be used to show the presence of bugs, but never to show their absence!"
- *Edsger Dijkstra*

"Cringley's Second Law: Ease of use with equivalent performance varies with the square root of the cost of development. That means that to design a computer that's ten times easier to use would cost 100 times as much. "

Some Conclusions

- **We will succeed in utilizing Exascale systems, we have never failed**
  - We are already at 10s of Petaflops
- **It will take "Smart Programming" regardless of the programming model employed**
- **Some percentage of applications will get to an Exaflop with incremental refactoring of the code**
- **Some percentage of applications will only get to an Exaflop with a significant rewrite**
- **For a programming model to succeed, as large percentage of the user base must adopt and work with the developers to improve and maintain that approach**
  - MPI, OpenMP, OpenACC,pthreads
  - Charm ++, Global Arrays