

HPC workload characterization

Understanding and evaluating the utilization of the system under normal workloads.

R. Scott Studham studham@ornl.gov

Computer Science and Mathematics Division

The majority of this work was done while I was working at PNNL with Ryan Mooney, Ken Schmidt and Jarek Nieplocha.

HPC Evaluation Methods

Platform Evaluation

Determine a platform's applicability to a range of applications.

- Microbenchmarks

- IO & System Characteristics

- Application Benchmarking

Work with manufacturers and system architects to make the systems better

Focus: Evaluation of emerging platforms to understand their benefits.

Workload Characterization

Determine how existing platforms are utilized by applications and users.

- “Efficiency” of application on platform

- System Utilization

- Average job sizes

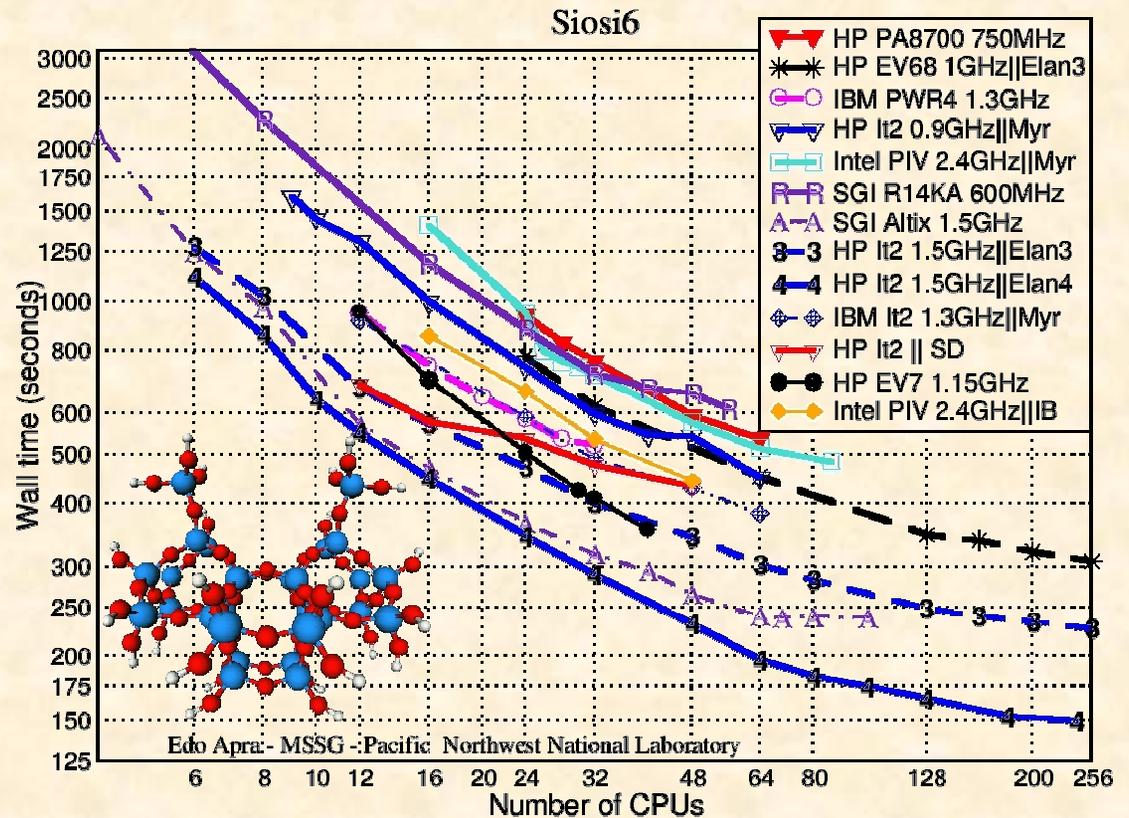
Work with application developers, users and manufacturers to decrease runtime.

Focus: Evaluation of deployed platforms to understand how they are used by general users.

Example Platform Evaluation

HPCS2 provides best capability and fastest time-to-solution for NWChem DFT benchmark

- We wanted to design a method to determine what is the average user doing vs. what the box was designed for?



Design Requirements

- Load system performance metrics (Eg FLOPs, Int Ops, Mem BW, Network BW, Disk BW.) for all jobs run on the system in a central database.
- Have a fine granularity to the data so you can see the needs of the different algorithms used.
- Keep all data and develop a mathematical “center profile”.
- Can’t impact application performance by greater than 1%

	Version1 CASC 2003	Version2 NWPerf 2003-2004	Version3 OpenWLC 2005+
Ease of use	Requires User Interaction	No User Interaction	No User Interaction
Portability	Hard coded for one computer	Designed for a Linux cluster	Run able on all major architectures
Users	O(10)	O(100)	O(1000)
Jobs	O(10 ²)	O(10 ⁴)	O(10 ⁵)
Metrics collected	O(10)	O(20)	O(40)
Deployed sites	1	1	O(10-100)

Examples of metrics collected

Each Metric is collected on all nodes once per minute

- Itanium Performance Counters, each of these metrics are collected separately for both CPU's in the compute nodes
 - *Flops – Floating point operations per second as a percent of theoretical peak
 - *Memory Bytes/Cycle – Average main memory accesses per CPU clock cycle
 - Total Stalls – Total stalls per CPU clock cycle, this may be one of 5 different stalls
- Local Scratch Usage (obtained via `fstat()`)
 - Blocks Used and Block Free
 - Inodes used and inodes free (this yields an estimate of files open)
- VMStat information (obtained via `/proc/meminfo` and `/proc/stat`)
 - *Memory swapped out (total), swap blocks in and out
 - *Memory free, used, and used as system buffers (cache + buffers)
 - *Block I/O in, and out
 - *Kernel Scheduler CPU allocation to user, kernel, and idle time
 - Processes running, and blocked
 - Interrupts, and Context Switches per second.
- Lustre I/O (Shared global Filesystem)
 - Bytes in/out (both client and OST)

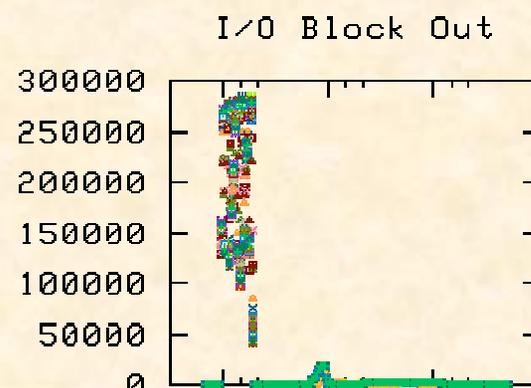
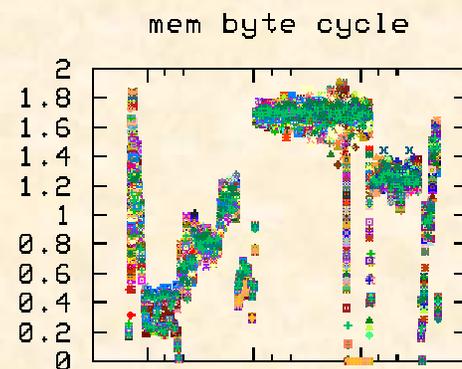
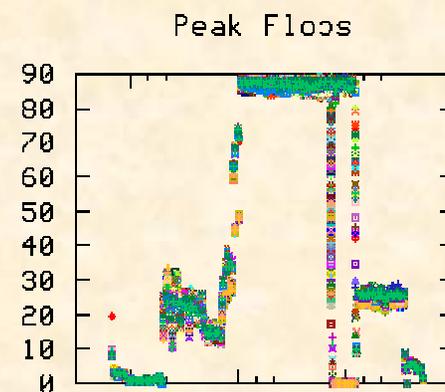
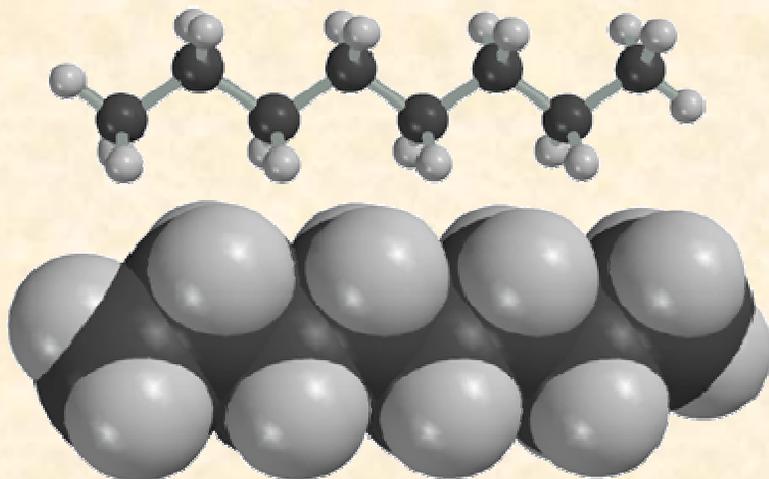
A “Good” job

The 3 graphs here are for a 3 day 600CPU job.

It was a CCSD(t) calculation of octane.

SCF at the beginning hit 61GB/s of IO.

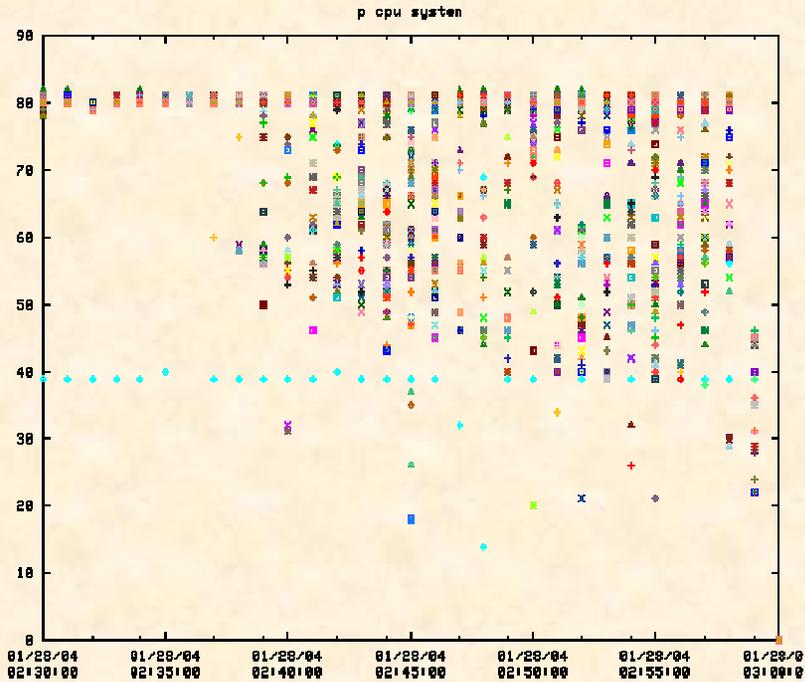
Utilized 1.8TB of memory and sustained 36% efficiency
1.3TF.



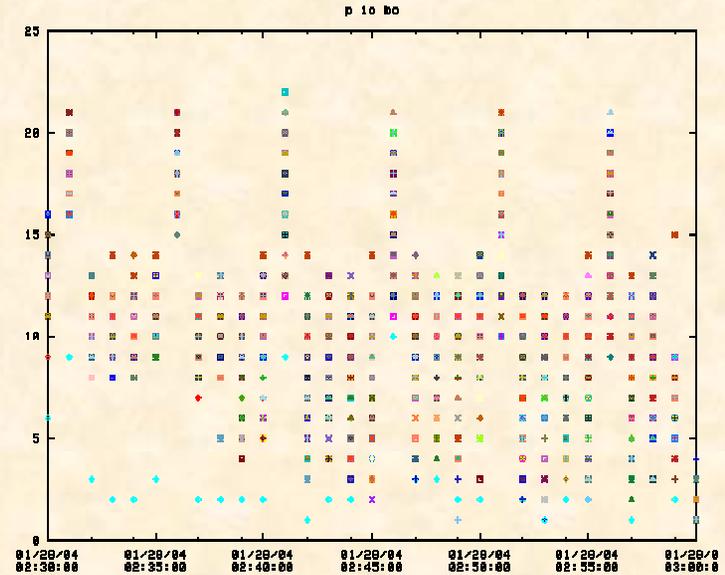
A “Bad” Job

Our “star” bad job

Suffered from floating point assists due to a compiler “feature” that caused optimistic prefetching of invalid data in some types of variable length loops



74% Mean Time in Kernel Space

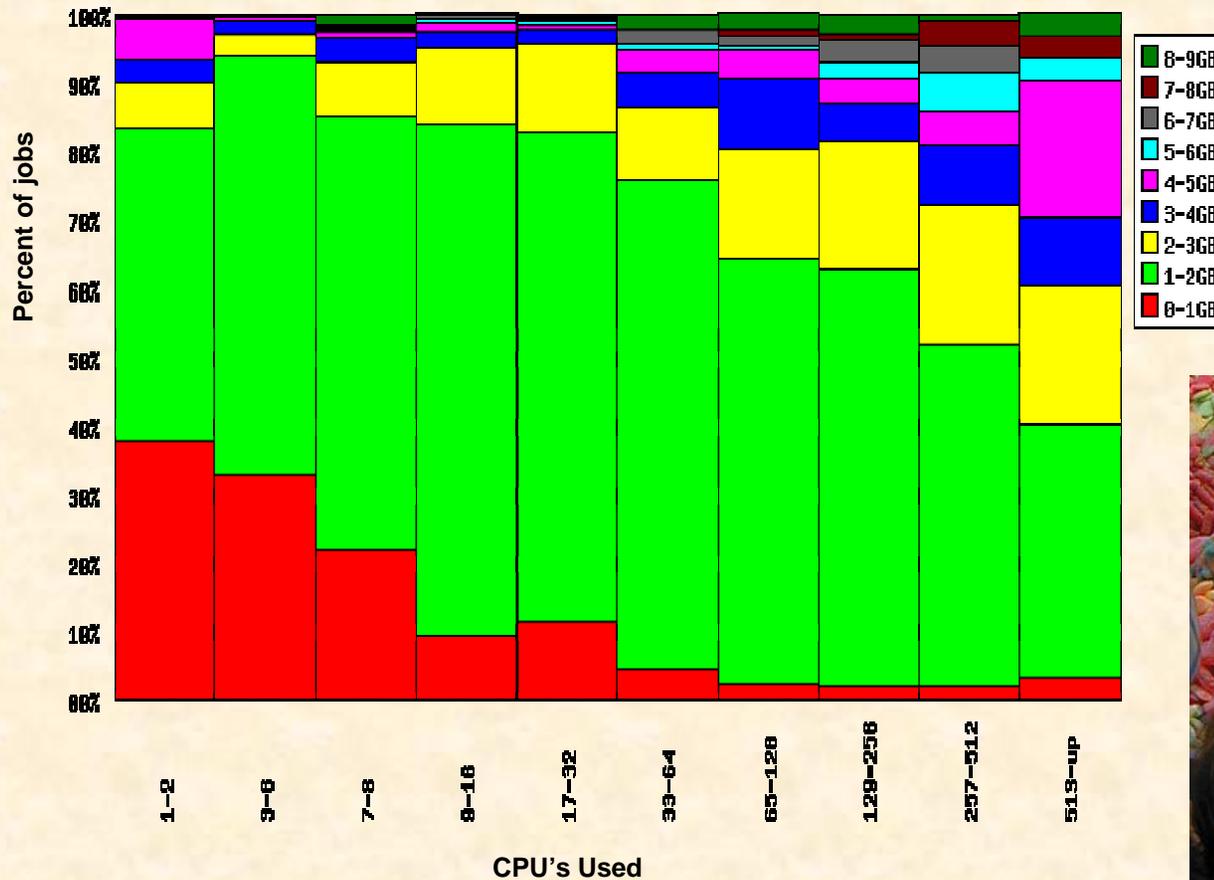


11KB/s Mean I/O to disk

```
double foo(int len) {
    ....
    for(i = 0; i<len; i++) {
        if(d_array[i] < d) {
            blah();
        }
    }
}
```

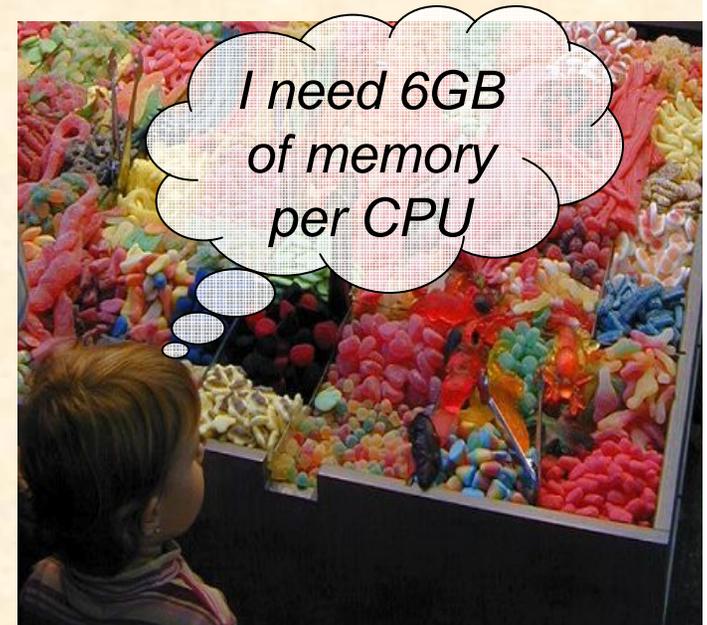
Most users do not utilize the full capability of the system.

Memory footprint per node during FY04 on 11.4TF HPCS2 system at PNNL

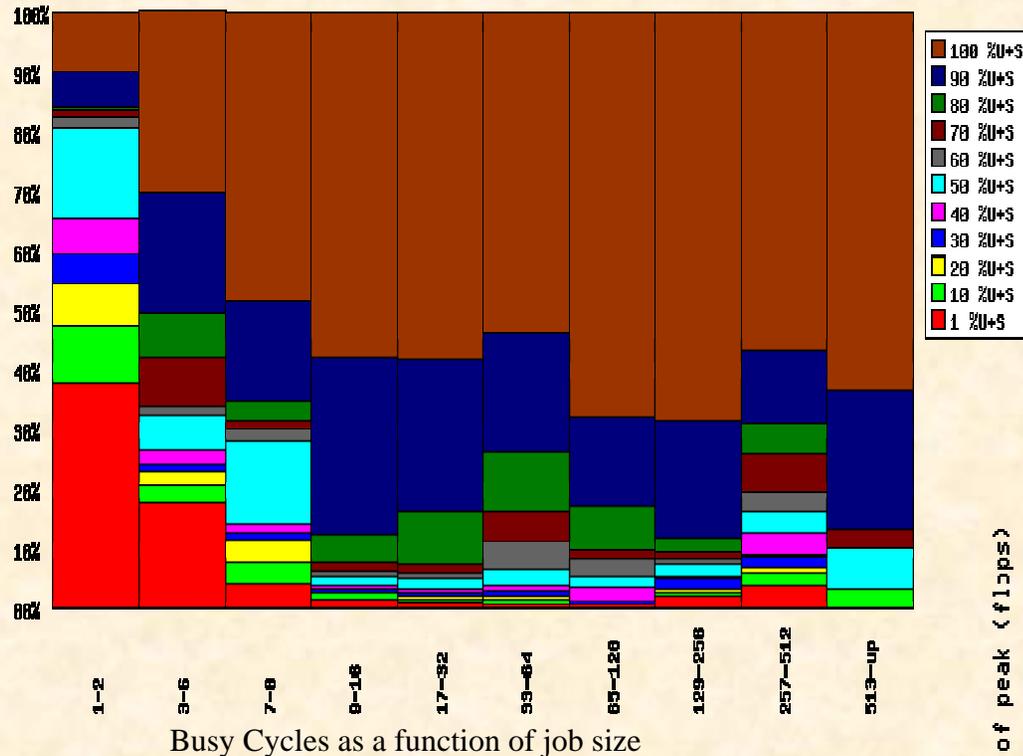


Most jobs use <25% of available memory (max avail is 6-8G)

Large jobs use more memory.

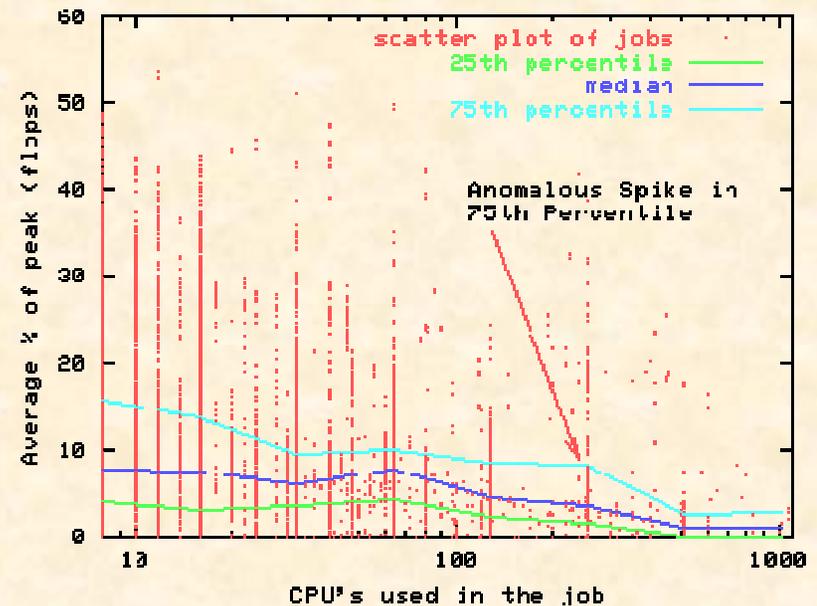


Aggregate Results



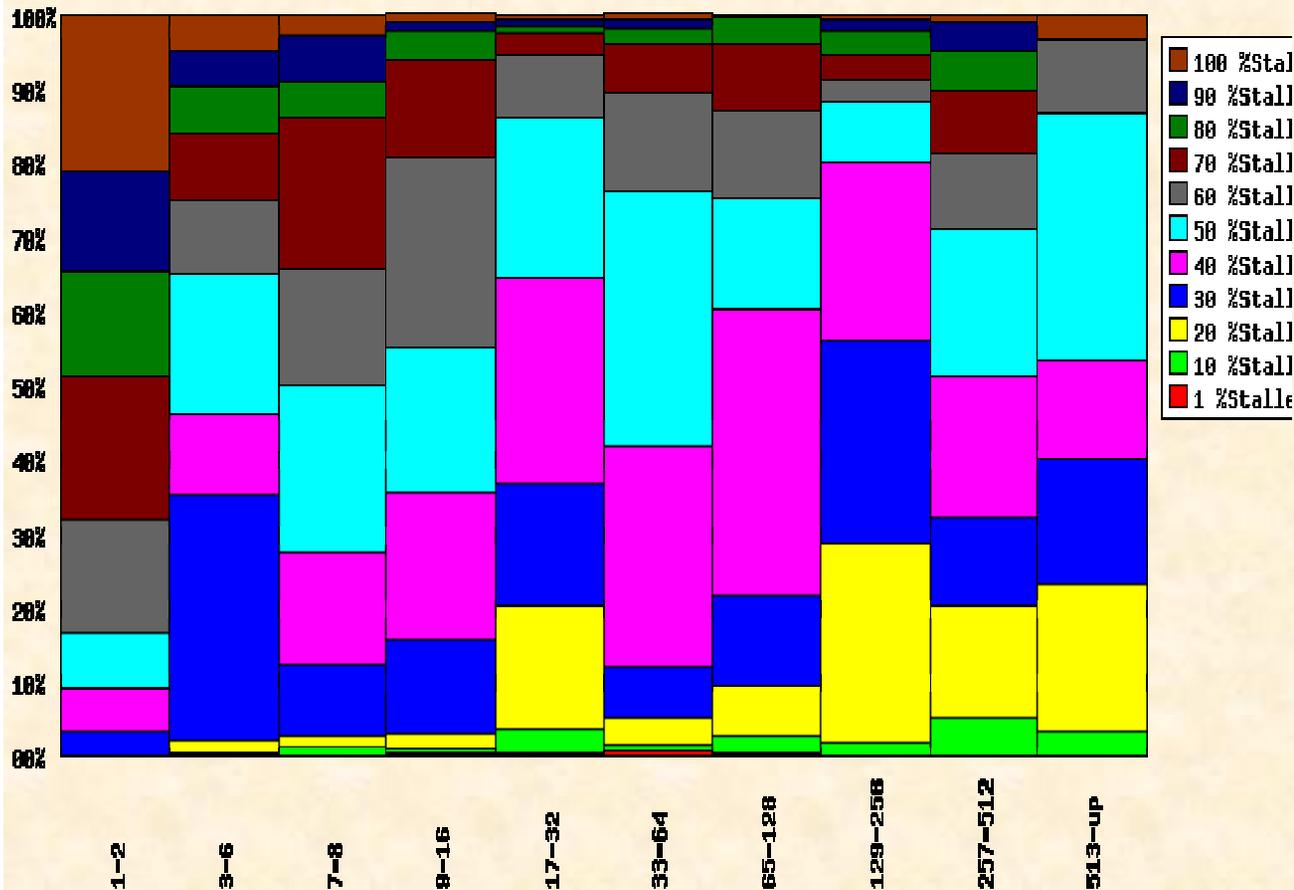
10% of the >256CPU jobs have the CPU scheduled for idle >50% of the time.

The median sustained performance for jobs over 256CPU's is 3%.



Sustained Performance as a function of CPU count

Aggregate Results



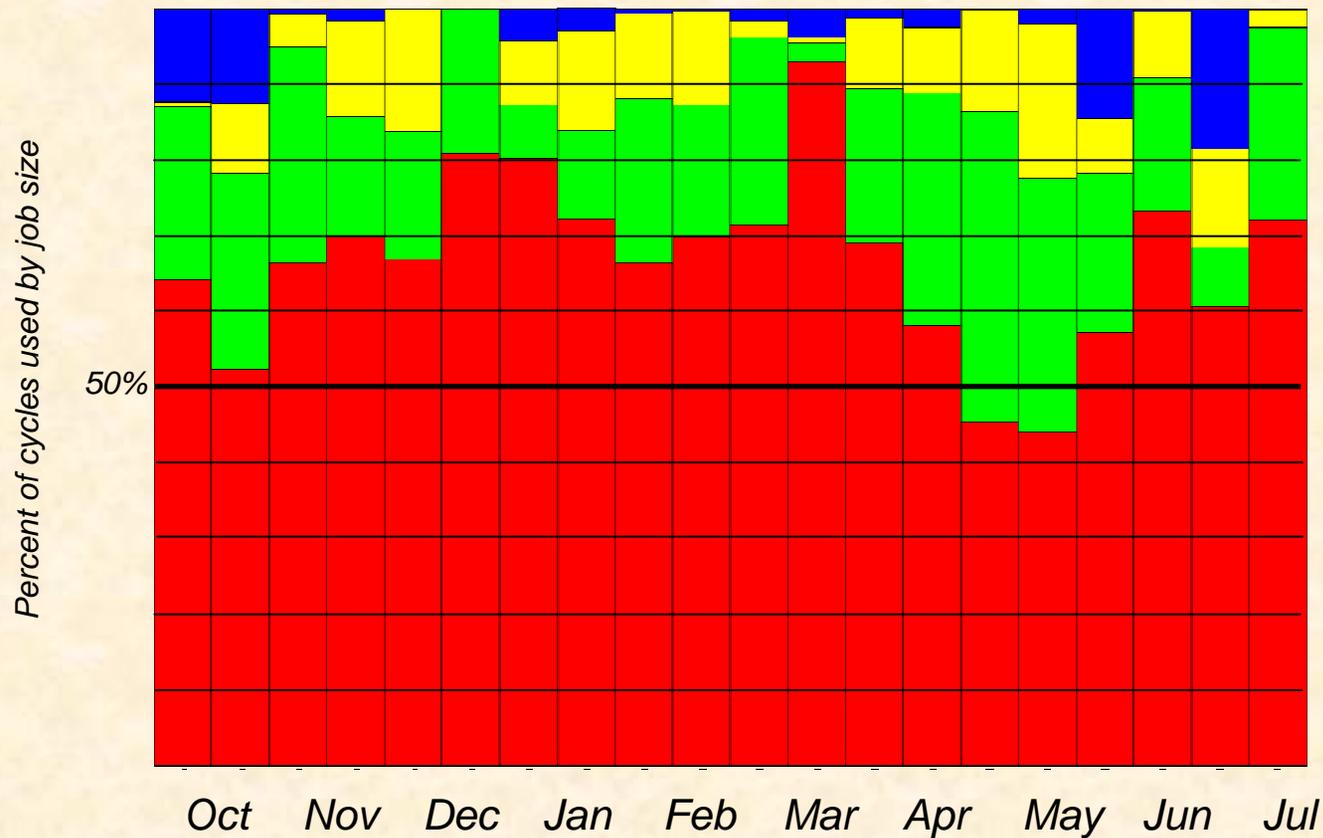
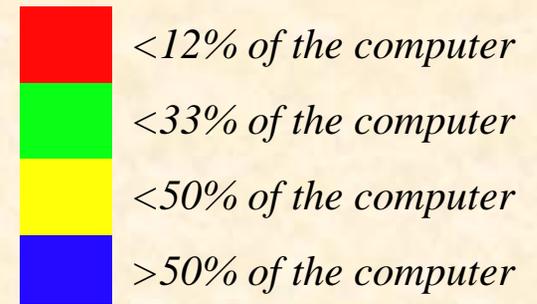
Stalled cycles as a function of job size

Sum of all stalls due to:

- BE_FLUSH_BUBBLE_ALL Branch misprediction flush or exception
- BE_EXE_BUBBLE_ALL Execution unit stalls
- BE_L1D_FPU_BUBBLE_ALL Stalls due to L1D (L1 data cache) micropipeline or FPU (floating-point unit) micropipeline
- BE_RSE_BUBBLE_ALL Register stack engine (RSE) stalls
- BACK_END_BUBBLE_FE Front-end stalls

Job size

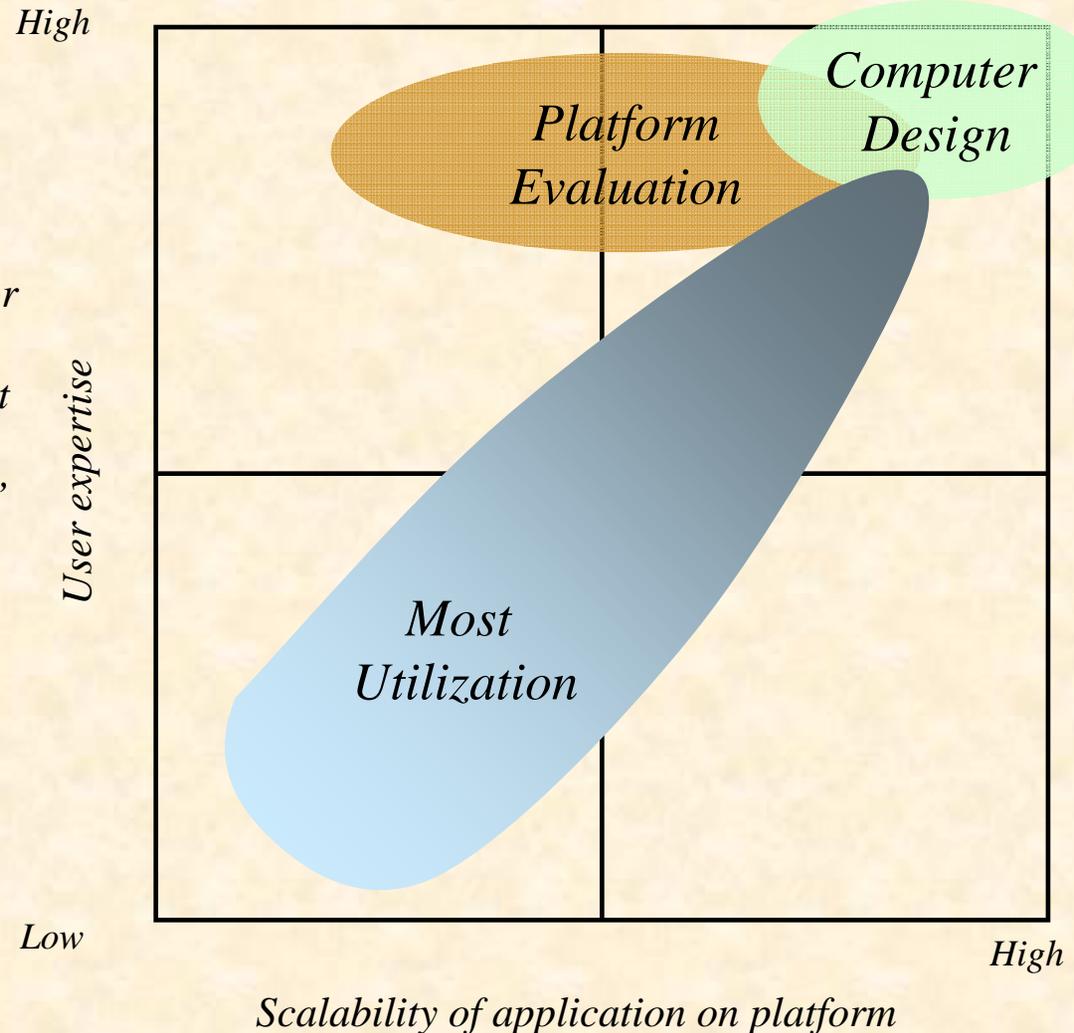
HPCS2 is primarily focused on small capacity jobs.



Discovery

After analyzing 19,883 jobs...

- Median sustained FLOP performance for jobs over 256CPU's is 3%
- Less than 20% of the memory is in use at any given point in time.
- Less than 5% of the jobs use "heavy IO" (>25MB/s per GF DGEMM)
- Over 60% of the cycles are for jobs that use less than 1/8 of the system.
- 10% of the >256CPU jobs have the CPUs scheduled for idle >50% of the time.



...we have determined that most users do not use the system as designed.

NLCF: Three purpose-built architectures optimized for applications



- Proven architecture for performance and reliability
- Most-powerful processors and interconnect
- Scalable, globally addressable memory and bandwidth
- Leverages commodity where possible
- Offers capability computing for key applications

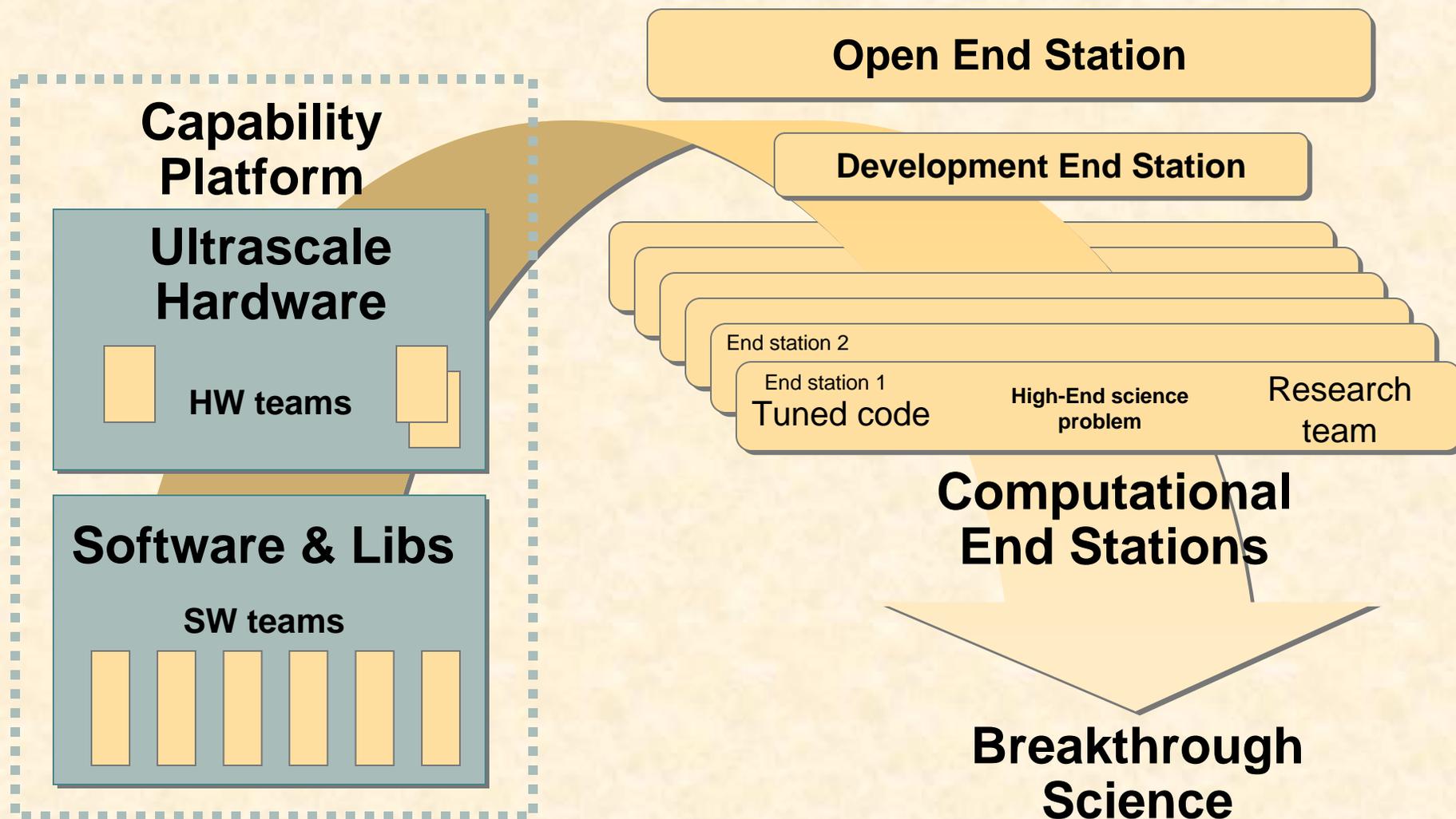


- Extremely low latency, high bandwidth, interconnect
- Efficient scalar processors, balanced interconnect
- Known system architecture – based on ASCI Red
- Shares interconnect technology with Cray X2
- Front end-hosted environment - system calls and I/O



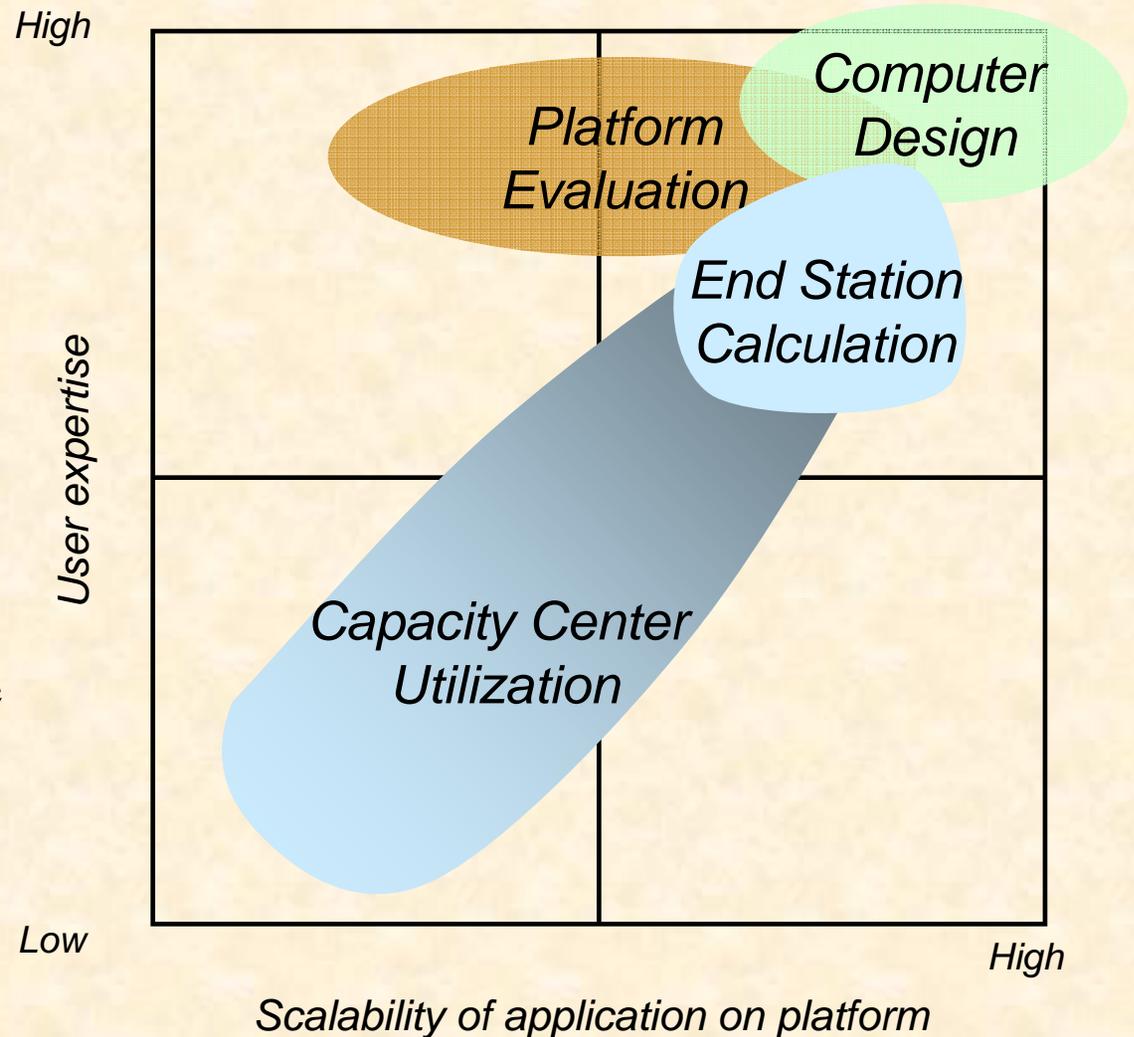
- Very low power and space requirements
- Interconnects balanced with processors
- Unique pairing mesh and tree, low latency
- Much higher parallelism (tens of thousands of CPU)
- Potential for new capabilities for selected applications
- Scalability platform for algorithms and CS research

Science teams enabled through “End Stations”



The Challenge

Work with the application community, via end stations, to educate the users as to how to efficiently use the systems and ensure the users applications scale on the given NLCF platforms.



Acknowledgements

Ryan Mooney and **Ken Schmidt** for their tireless work to develop NWPerf.
Jarek Nieplocha for his guidance on how to quantify system impacts.

This research described in this presentation was performed using the **Molecular Science Computing Facility** (MSCF) in the William R. Wiley Environmental Molecular Sciences Laboratory, a national scientific user facility sponsored by the **U.S. Department of Energy's Office of Biological and Environmental Research** and located at the Pacific Northwest National Laboratory. PNNL is operated for the Department of Energy by Battelle.

This research is sponsored by the Office of **Advanced Scientific Computing Research; U.S. Department of Energy**. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.

Experiments and data collection were performed on the Pacific Northwest National Laboratory (PNNL) 977-node Linux 11.8 TFLOPs cluster (HPCS2) with 1954 Itanium-2 processors.

The data collection server is a Dual Xeon Dell system with a 1TB ACNC IDE to SCSI Raid Array.

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

