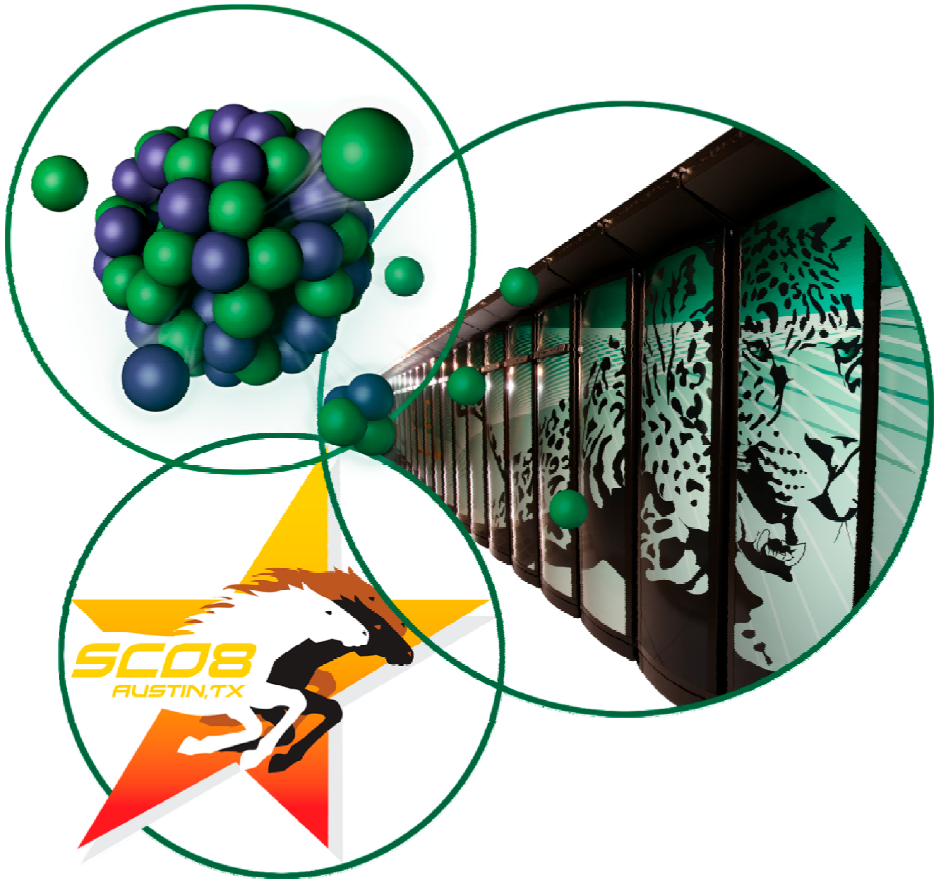
FAST-OS: Petascale Single System Image



Presented by

Jeffrey S. Vetter (PI)
Collin McCurdy
Weikuan Yu

Future Technologies Group
Computer Science and Mathematics Division

Managed by UT-Battelle
for the Department of Energy

Oak Ridge National Laboratory

Petascale single system image

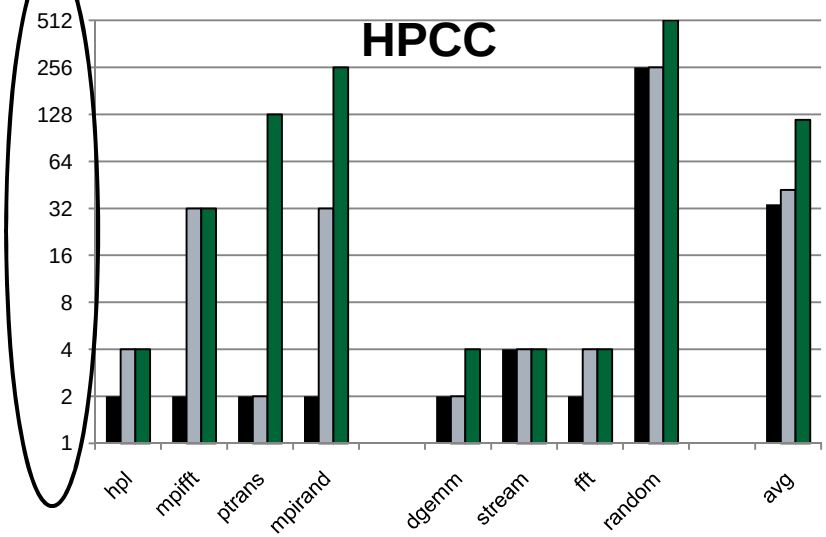
- **What?**
  - Make a collection of standard hardware look like one big machine, in as many ways as feasible
- **Why?**
  - Provide a single solution to address all forms of clustering
  - Simultaneously address availability, scalability, manageability, and usability
- **Components**
  - Paging behaviors of applications, i.e., reducing TLB misses
  - Performance and scalability of parallel shared root file system
  - Parallel I/O virtualization in virtual clusters
  - Reducing “noise” from operating systems at scale

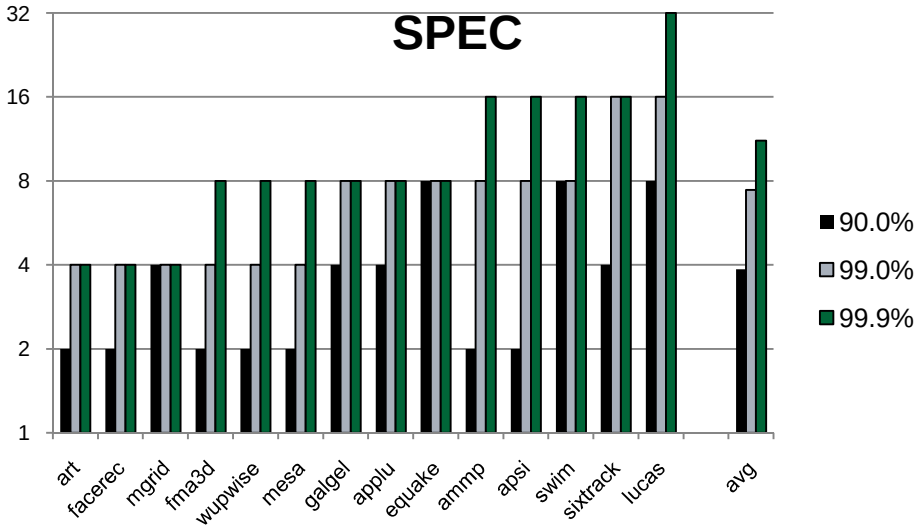
Investigating the TLB behavior of high-end scientific applications on commodity microprocessors
(McCurdy, Cox, Vetter; ISPASS'08)

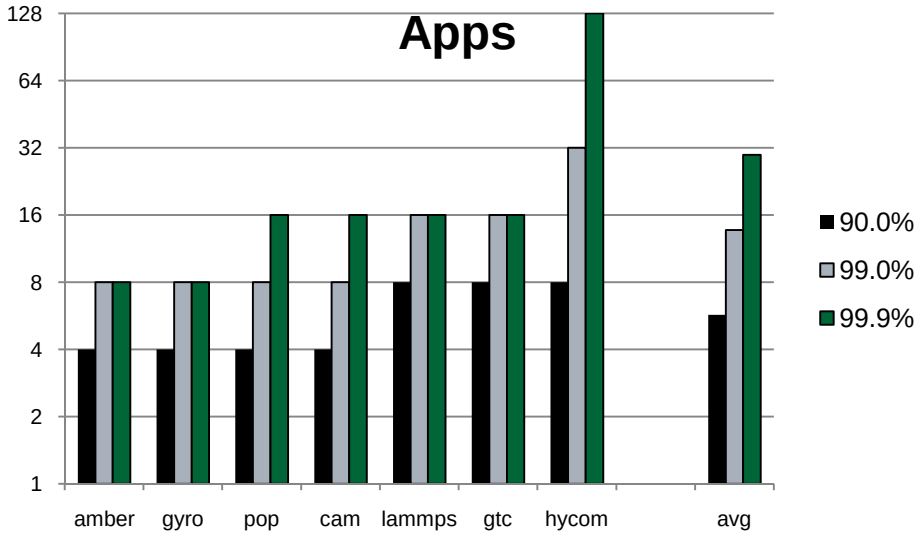
- Studies difference in TLB behavior between benchmarks (SPECFP and HPCC) and applications
- Two parts:
  1. Implementation-independent results
     - Uses "reuse distance" metric to estimate TLB size requirements for each benchmark and application
     - Striking differences between benchmarks and application
       - Benchmarks underestimate requirements for large (2 MB) pages
       - Benchmarks overestimate requirements for small (4 KB) pages
  2. Validation and performance impact
     - Hardware counter measurements on Opteron demonstrate the difference on real hardware
     - The difference can matter

TLB requirements, large pages

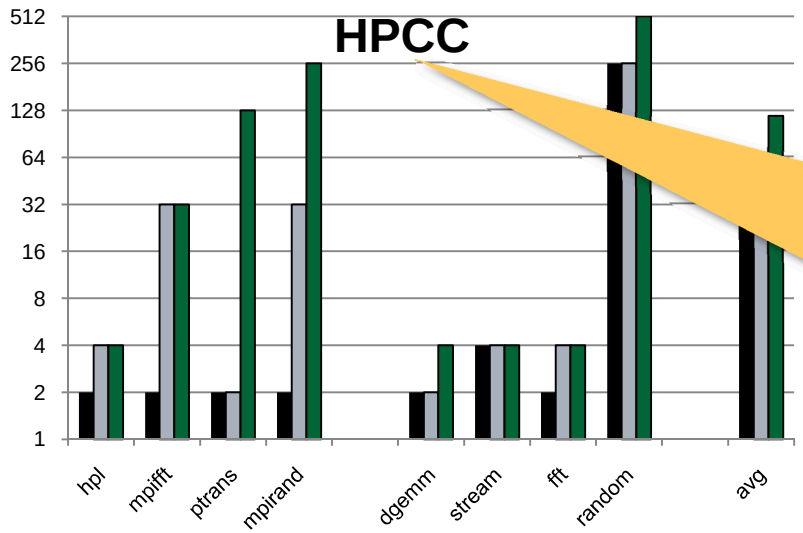
Number of TLB entries required to ensure (90,99,99.9)% references hit



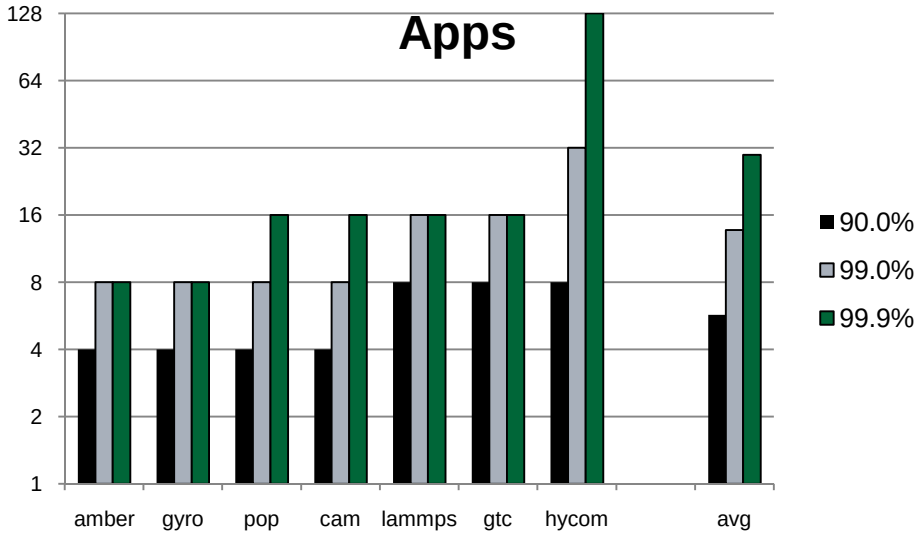




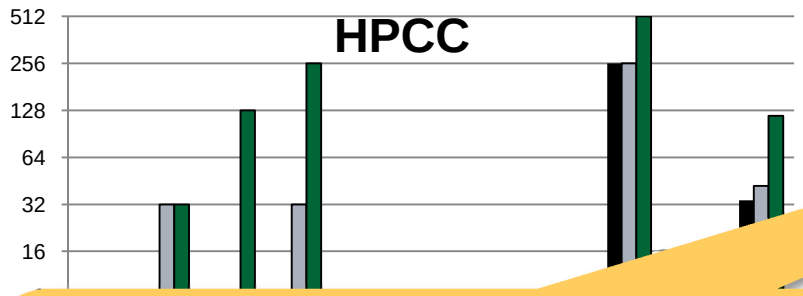
TLB requirements, large pages

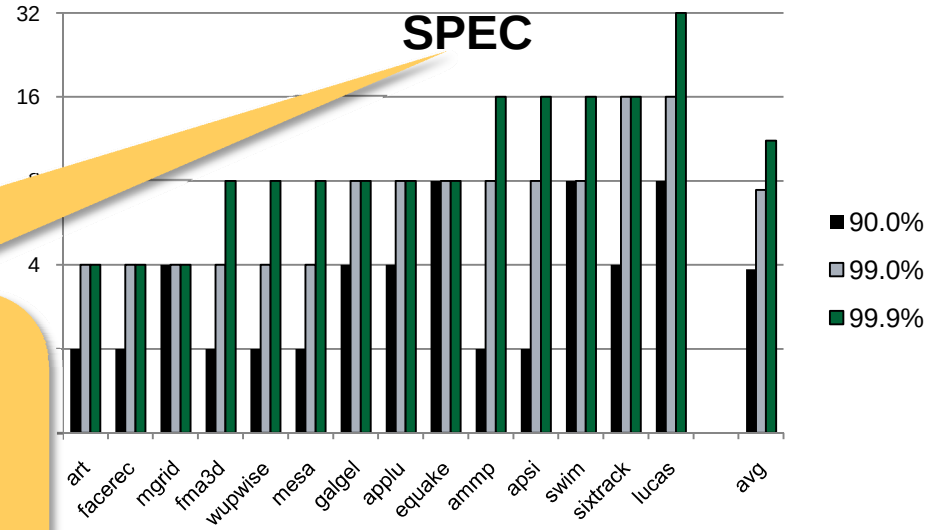


- 7 require only 2 entries for 90% coverage
- 4 require only 4 entries for 99.9% coverage

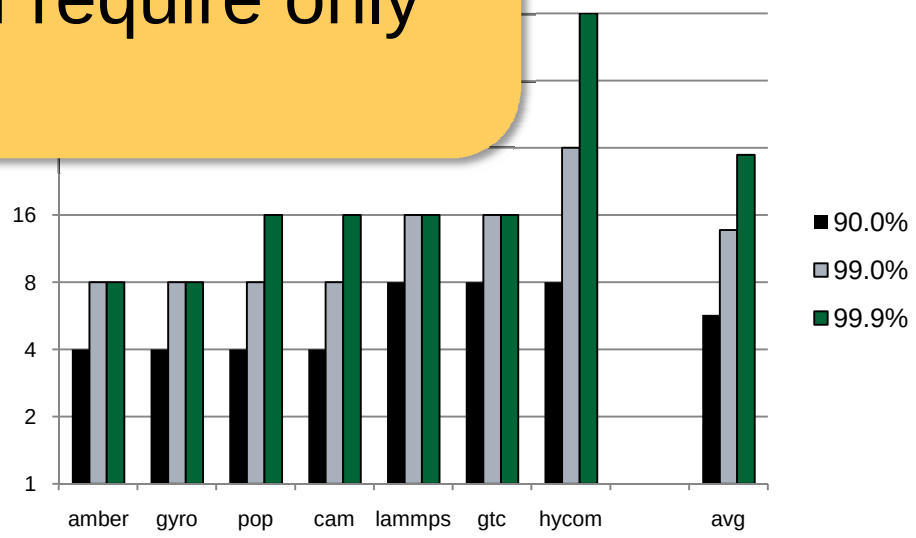


TLB requirements, large pages

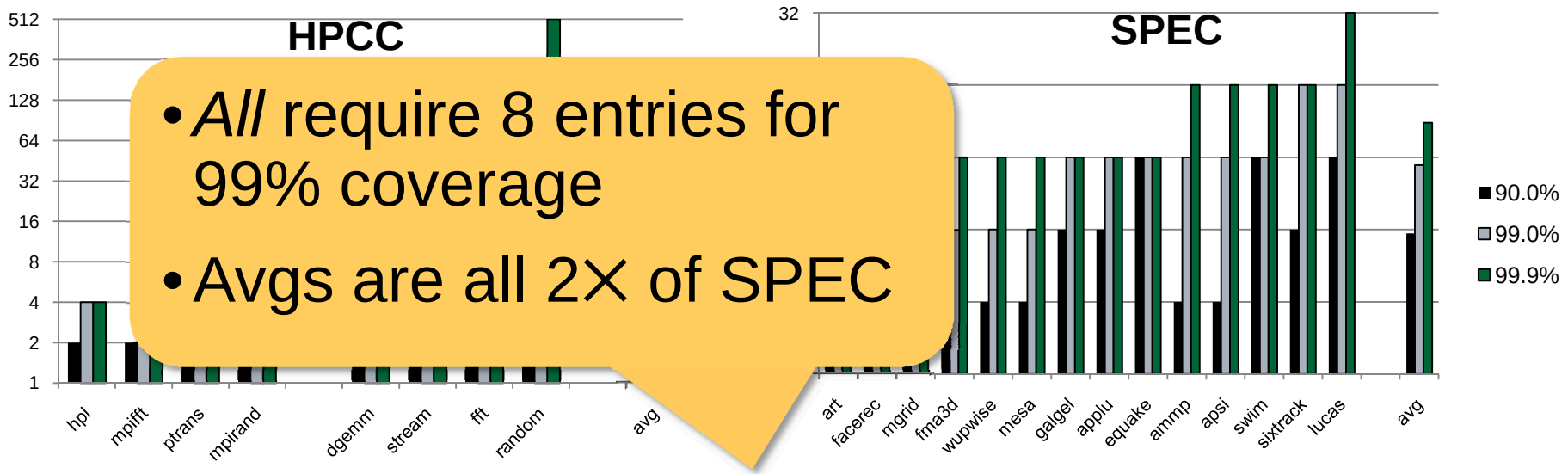




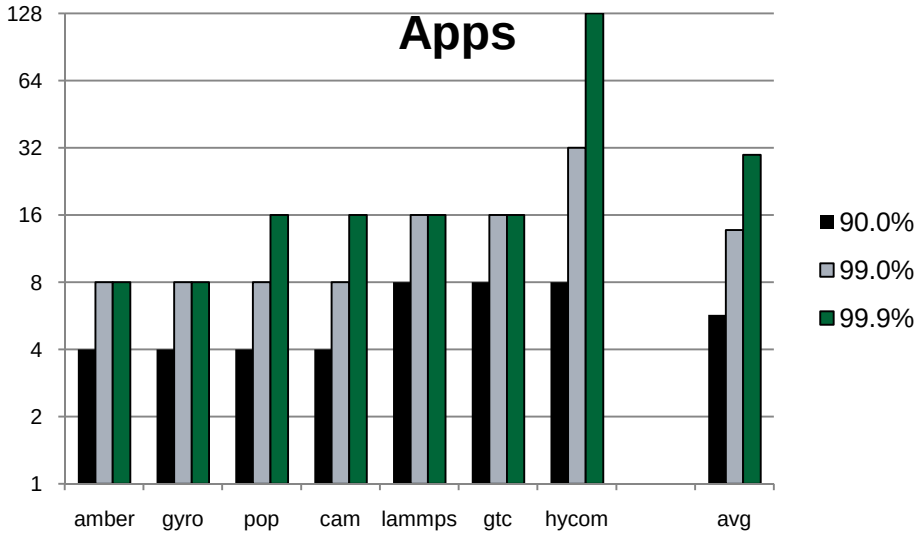
- Several require only 4 entries for 99.9%
- Nearly half require only 4 for 99%



TLB requirements, large pages



- *All* require 8 entries for 99% coverage
- Avgs are all 2X of SPEC



Why do benchmarks underestimate?

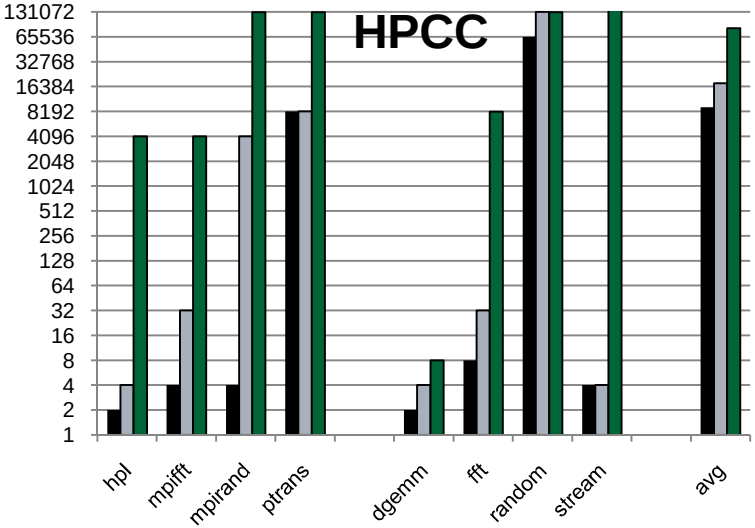
Applications feature more arrays "in-flight"

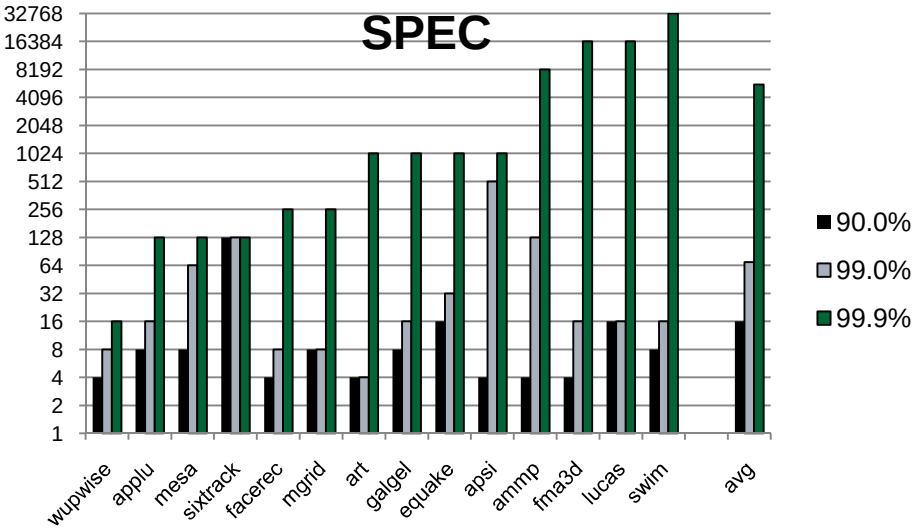
- **Consider a simple model:**

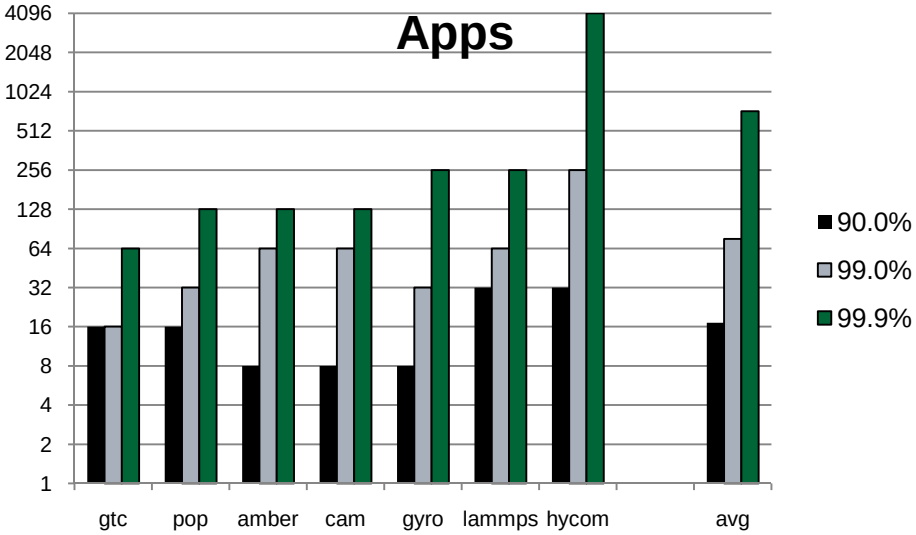
```
do i=1,LARGE_N
    x = x + A(i) + B(i) + C(i)…
enddo
```

- **In general, two ways to "avoid" reuse**
  1. **Cross page boundary**
  2. **Access a different array**
- **But large pages result in few page crossings…**
- **So reuse distance largely determined by number of arrays accessed**
- **Code inspection supports hypothesis**
  - **HPCC benchmarks feature many operations on a few arrays**
  - **Contrast: One module of HYCOM operates on more than 40 arrays**

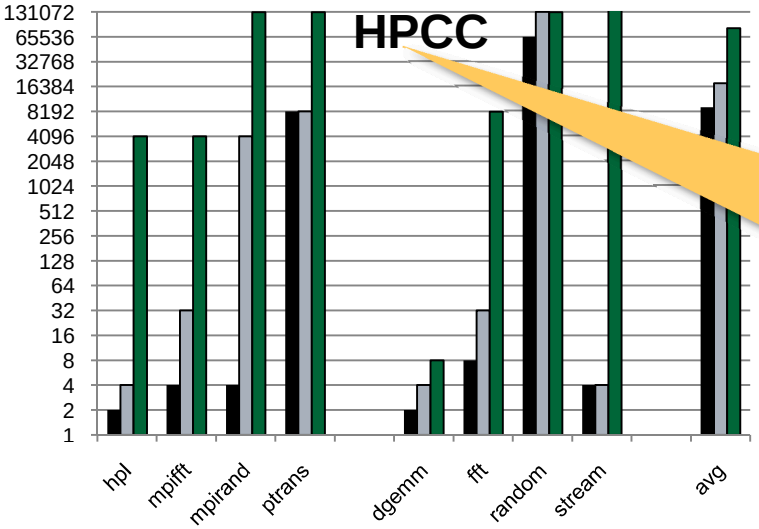
TLB requirements, 4 KB

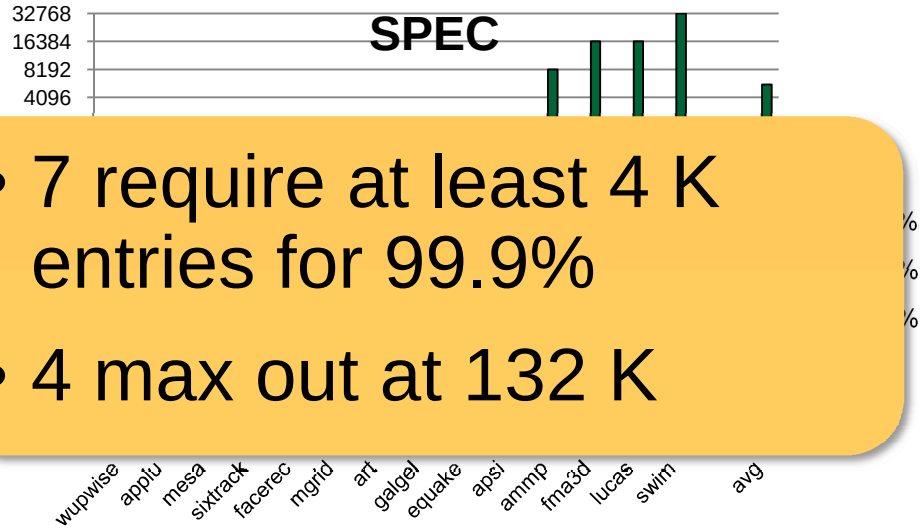




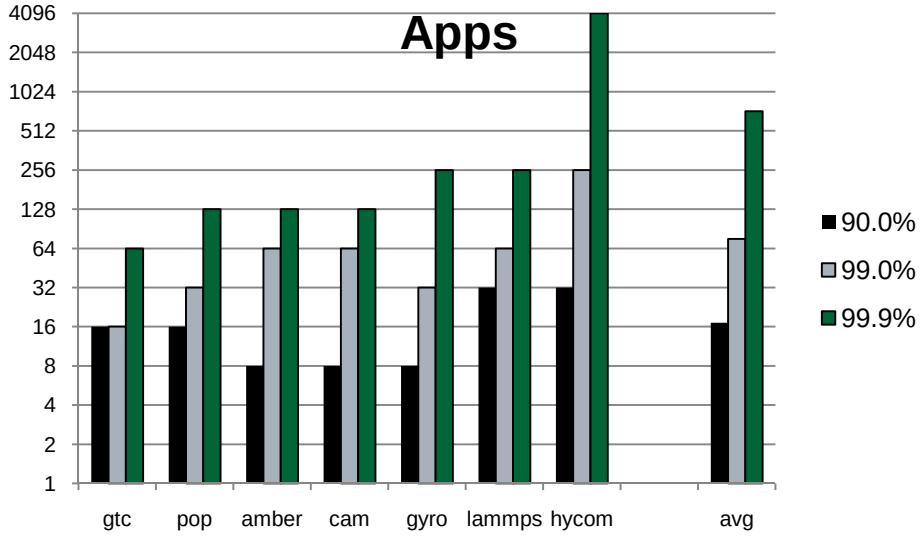


TLB requirements, 4 KB

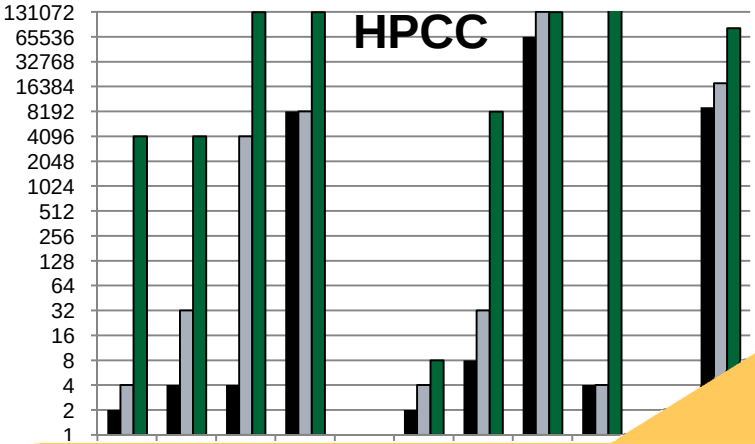


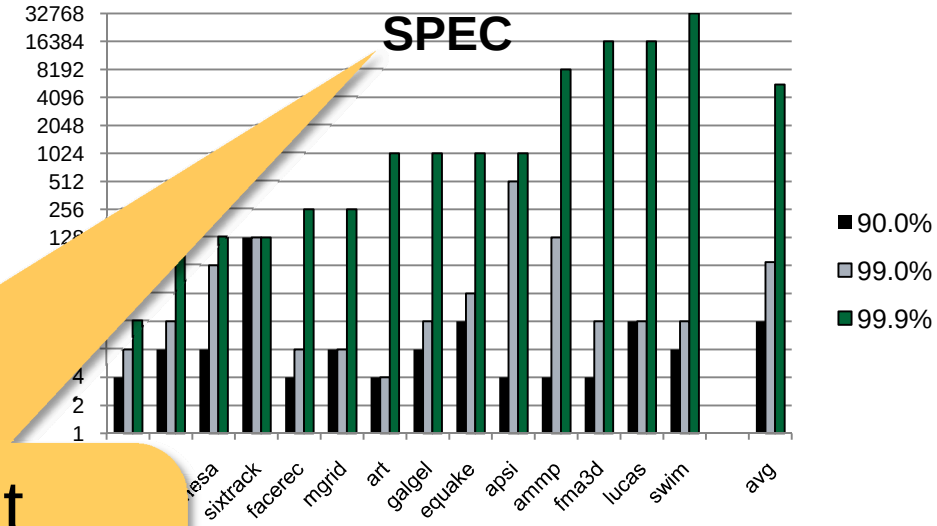


- 7 require at least 4 K entries for 99.9%
- 4 max out at 132 K

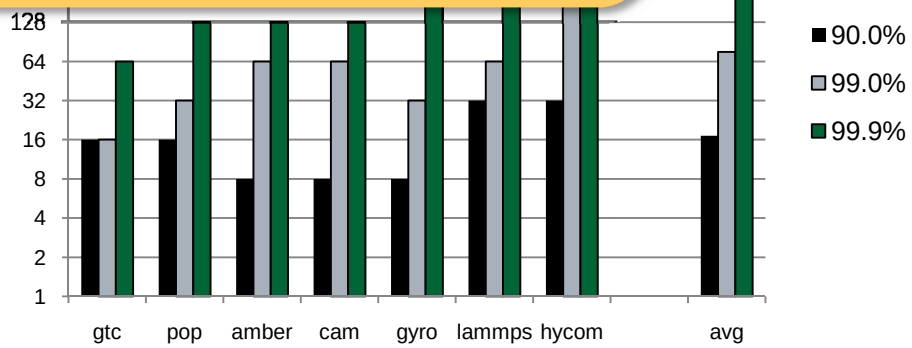


TLB requirements, 4 KB

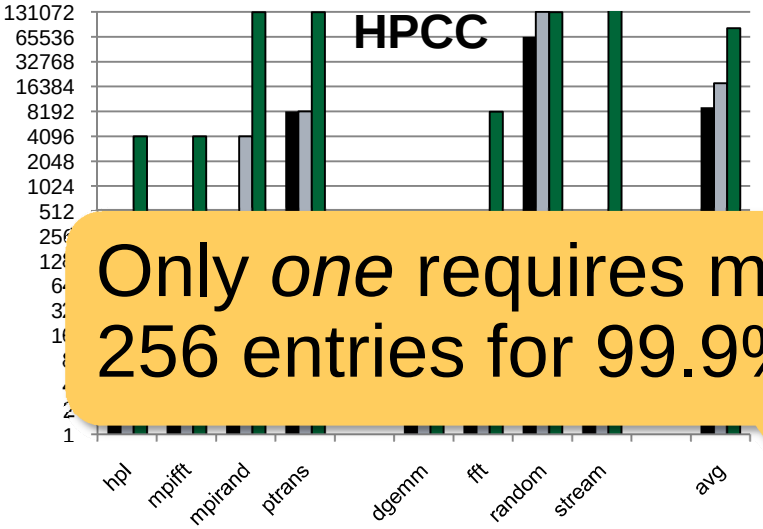


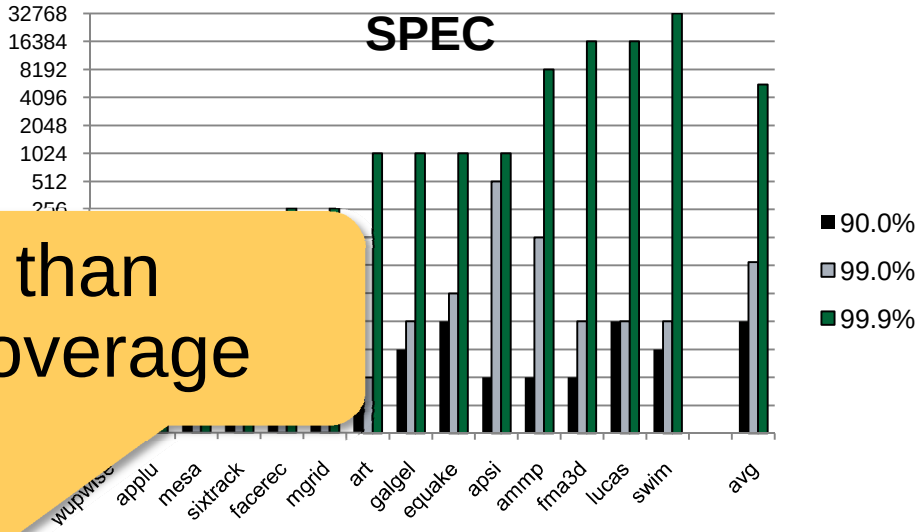


- More than half require at least 1 K entries for 99.9%
- Avg for 99.9% is over 4 K

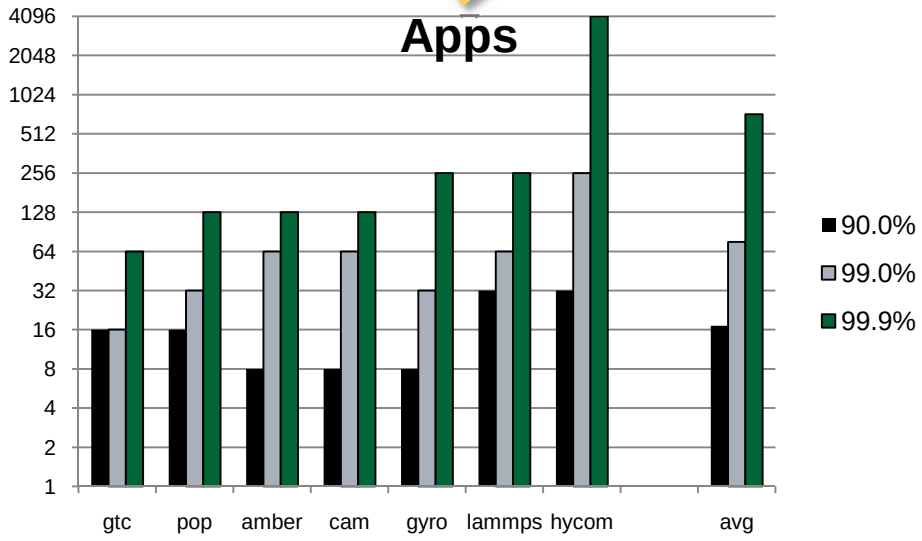


TLB requirements, 4 KB





Only *one* requires more than 256 entries for 99.9% coverage



Why do benchmarks overestimate?

Applications feature more reuse per iteration

- **Consider the same example**

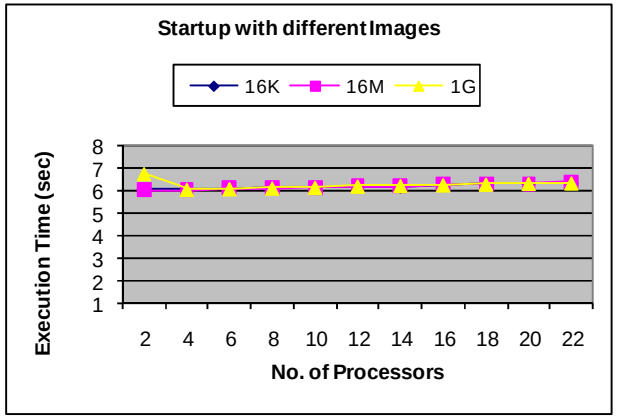
```
do i=1,LARGE_N
    x = x + A(i) + B(i) + C(i)…
enddo
```

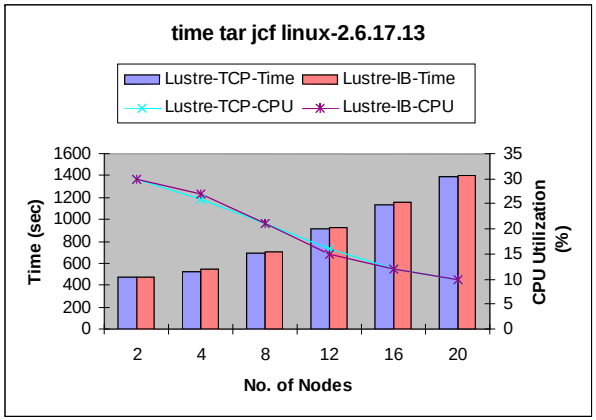
- **Now each array requires many entries**
  - **Page crossings dominate**
    - **Guaranteed to miss every 512 (double) elts**
    - **Impossible to reach 99.9% coverage**
- **Applications must not be streaming…**
- **Again, code inspection supports the claim**

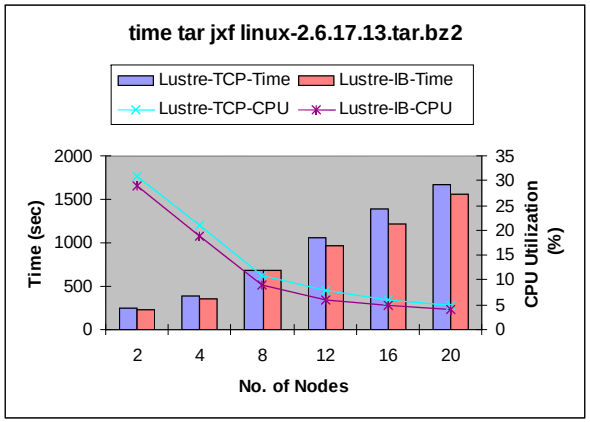
Parallel root file system

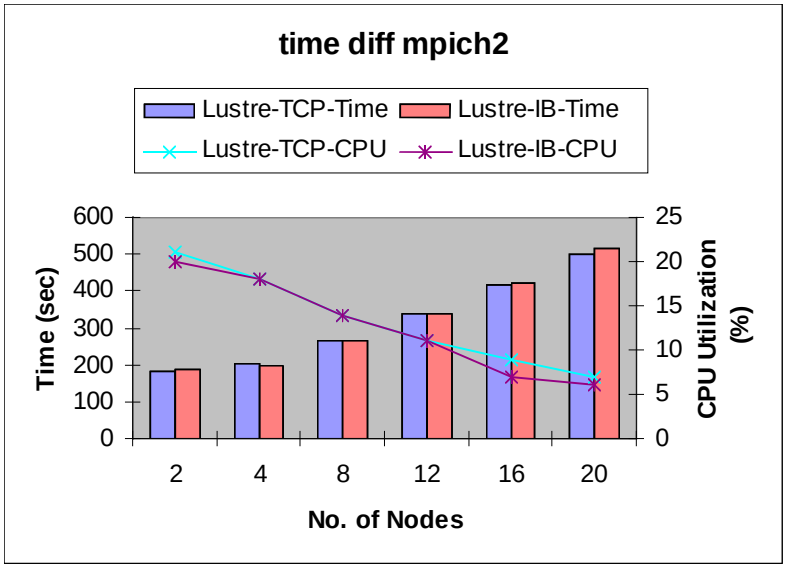
- **Goals of study**
  - Use a parallel file system for implementation of shared root environment
  - Evaluate performance of parallel root file system
  - Evaluate the benefits of high-speed interconnects
  - Understand root I/O access pattern and potential scaling limits
- **Current status**
  - RootFS implemented using NFS, PVFS-2, Lustre, and GFS
  - RootFS distributed using ramdisk via etherboot
    - Modified mkinitrd program locally
    - Modified to init scripts to mount root at boot time
  - Evaluation with parallel benchmarks (IOR, b_eff_io, NPB I/O)
  - Evaluation with emulated loads of I/O accesses for RootFS
  - Evaluation of high-speed interconnects for Lustre-based RootFS

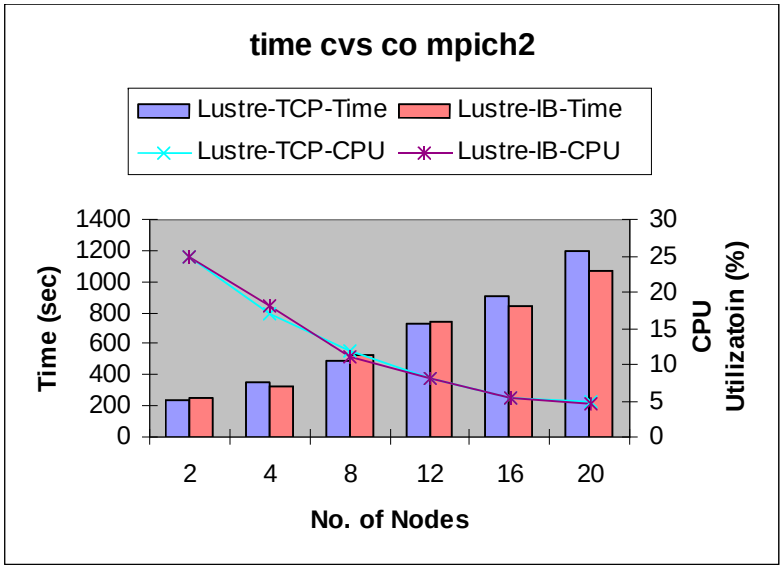
Performance with synthetic I/O accesses











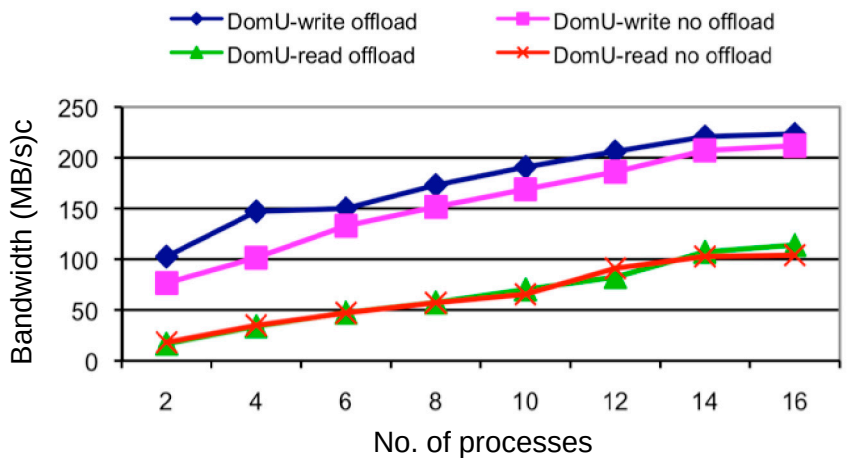
Xen-based HPC: A parallel I/O perspective
W. Yu, J.S. Vetter, CCGrid 2008

Studies the performance of parallel I/O in virtualized environments

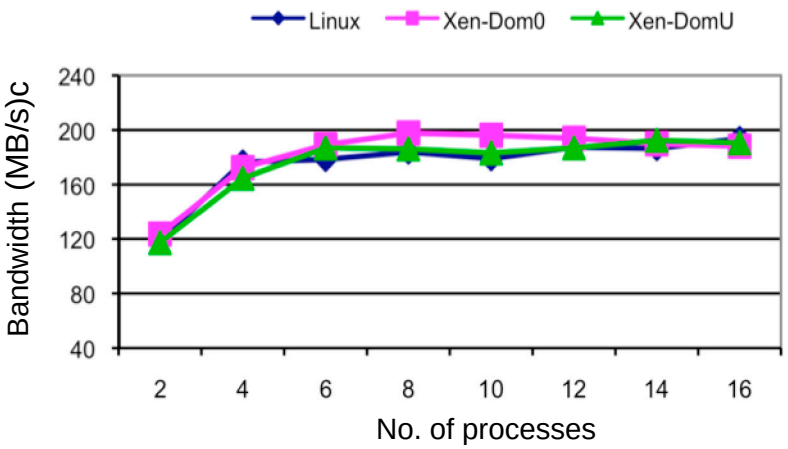
- **Virtualization overhead for parallel I/O**
  - **Examine the virtualization overhead**
  - **Tune the parallel IO performance for virtualization**
- **Migration cost for parallel I/O**
- **Analyzed gigabit Ethernet and InfiniBand**

Virtualization overhead

Parallel I/O on gigabit ethernet

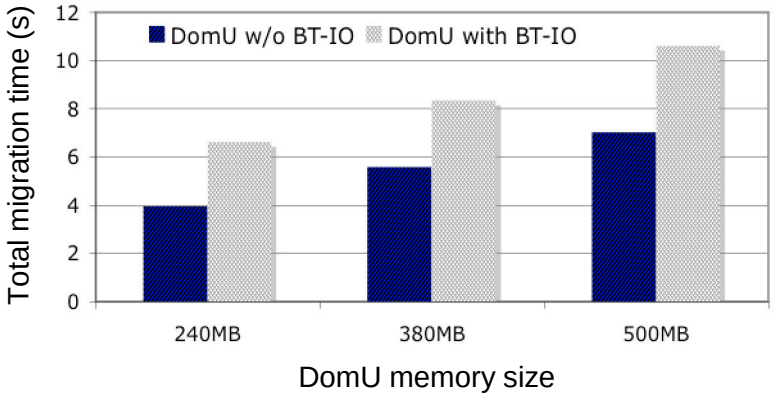


IOR write to A shared file—IB

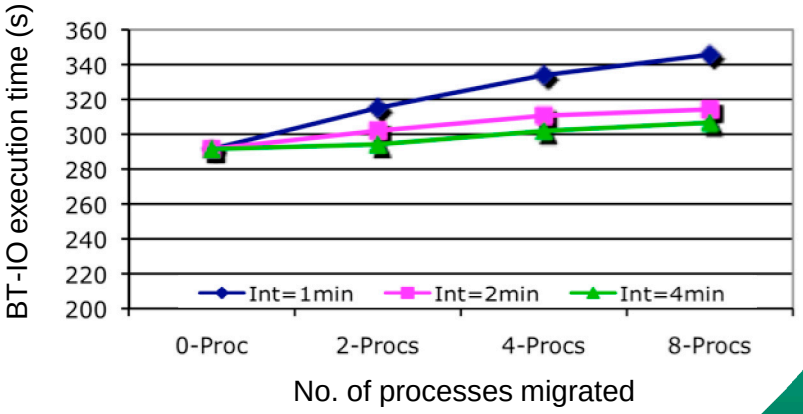


Migration cost

Parallel migration time



Migration time with different intervals



Contacts

Jeffrey S. Vetter

Principle Investigator
Future Technologies Group
Computer Science and Mathematics Division
(865) 356-1649
vetter@ornl.gov

Collin McCurdy

(865) 241-6433
cmccurdy@ornl.gov

Weikuan Yu

(865) 574-7990
wyu@ornl.gov