Argonne NATIONAL LABORATORY

Extending and Benchmarking the "Big Memory" Implementation on Blue Gene/P Linux

Kazutomo Yoshii

Mathematics and Computer Science Division
Argonne National Laboratory

U.S. DEPARTMENT OF ENERGY

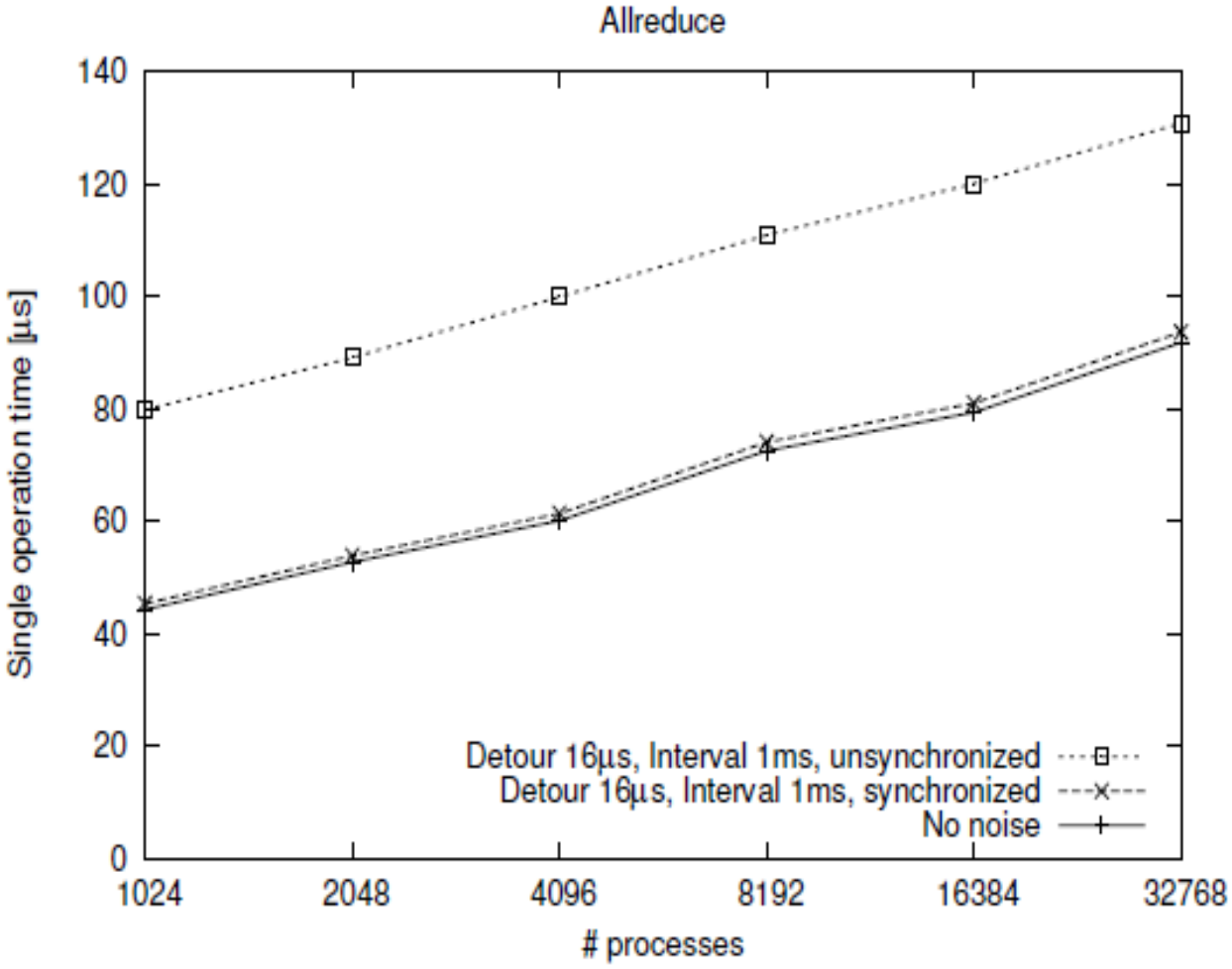
ZeptoOS Project

- Activities:
  - System Noise Study: Selfish suite
  - I/O Forwarding: ZOID
  - Memory Subsystem: Big Memory
  - Communication Stack
  - Performance Analysis: Ktau
- Open Source
- Kernel profile at the ANL BGP machine
- Collaborators:
  - U. of Oregon, U. Of Chicago, U. Of Delaware
  - ASTRON (the Netherlands Foundation for Research in Astronomy)
  - U. of Tokyo
  - IBM

IBM Blue Gene/P

- PowerPC 450
  - Compute Node(CN) and I/O Node(ION)
  - 4-way SMP , cache coherent(L1D), write-through is required
  - Software managed TLB, 64 entries, 1KB – 1GB page size
- Special Network
  - Torus, Collective, barrier, jtag
- Compute Node Kernel(CNK)
  - Tickless kernel (noise free), no pre-emption
  - Static mapped TLBs
  - 3 running modes(job submission parameter):
    - SMP, DUAL and Virtual Node(VN)
  - No flexible! e.g. no remote login
  - Getting complicated

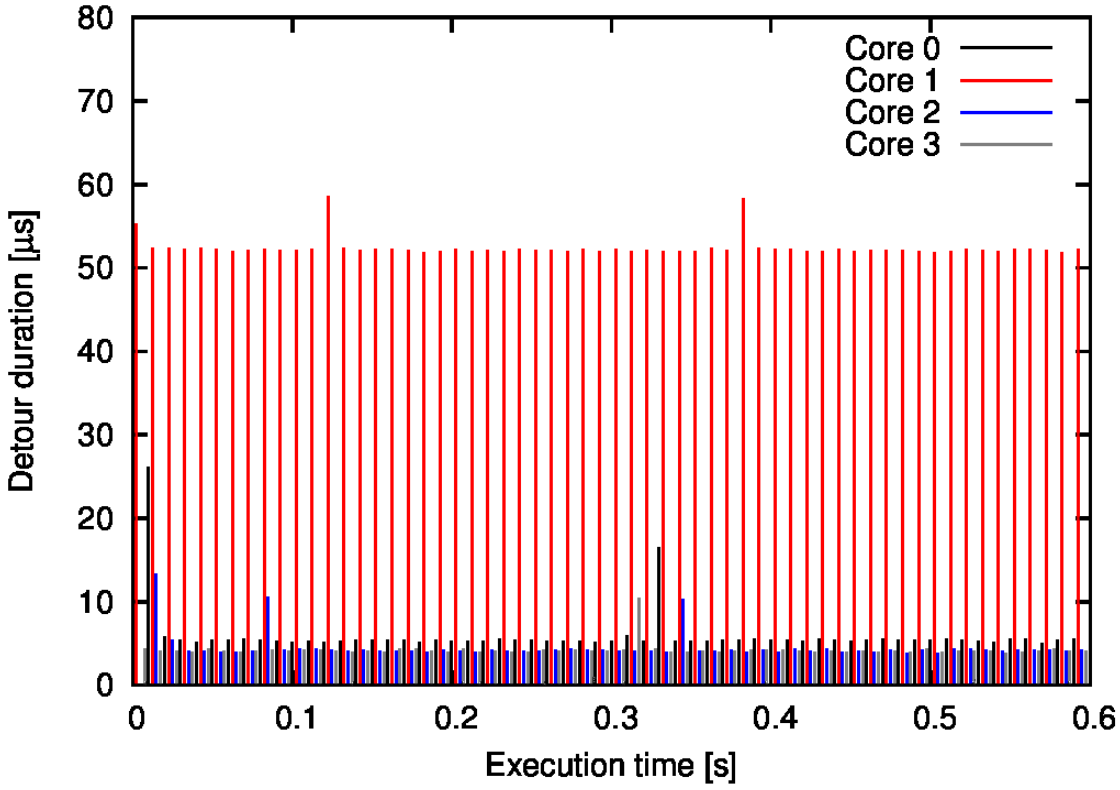
OS Noise Experiments on Blue Gene/L CNK



Injected artificial noise:

16 usec detour every 1ms
1.6% of CPU time
(0.03 - 0.1% on Linux)

OS Noise on Blue Gene/P Linux (2.6.29)



Average: 0.17%

Core1: 0.53%

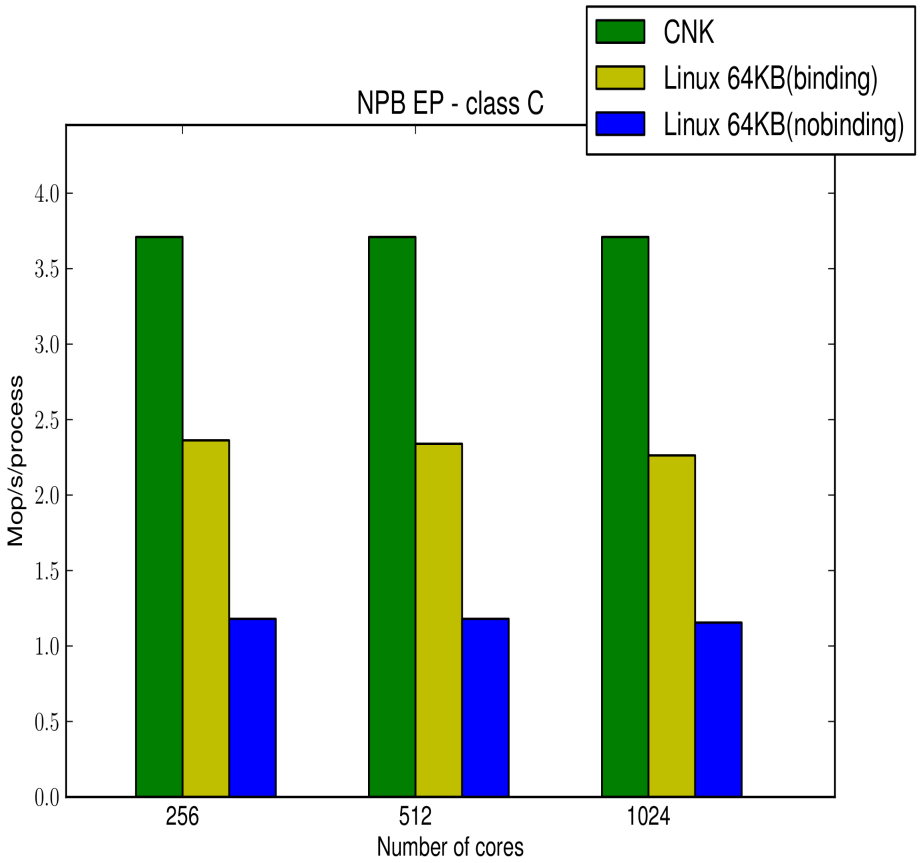
Other Cores: 0.05%

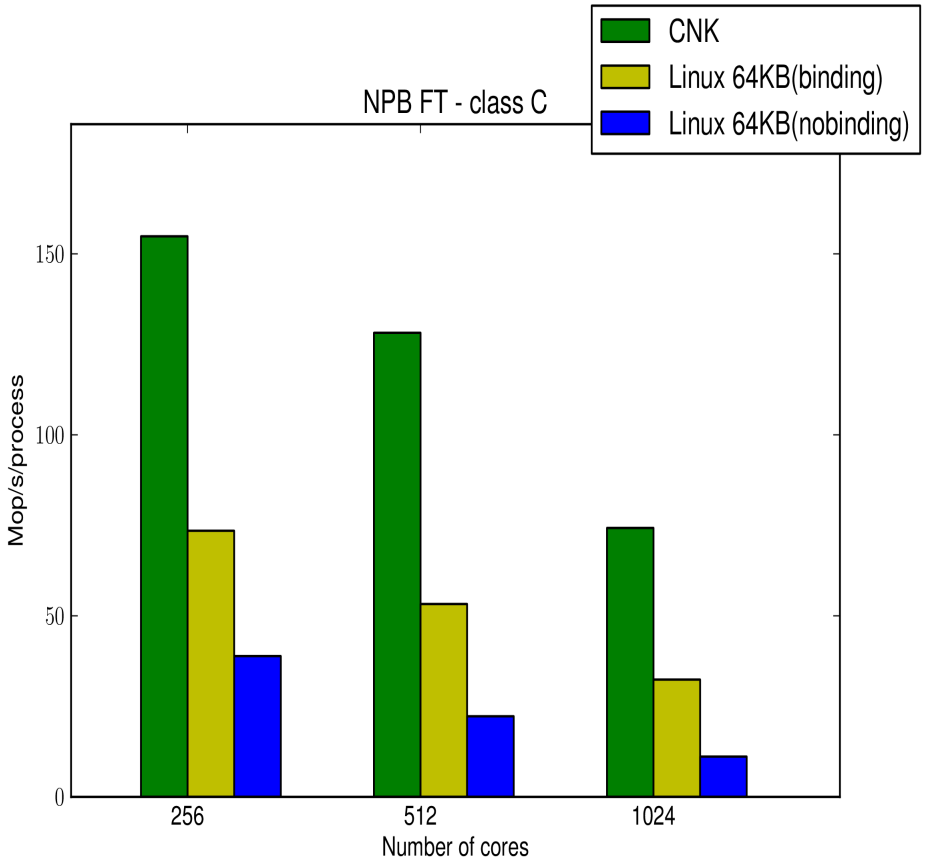
Fixable but left unmodified

Preliminary Measurements

- Start four processes on node(4-way SMP)
- Compare performance between:
  - CNK VN mode
    - strict CPU affinity
  - Linux(SMP) 64KB
    - **no-binding**
      - Up to Linux scheduler, migration might occur
    - **binding**
      - Set CPU affinity, no migration

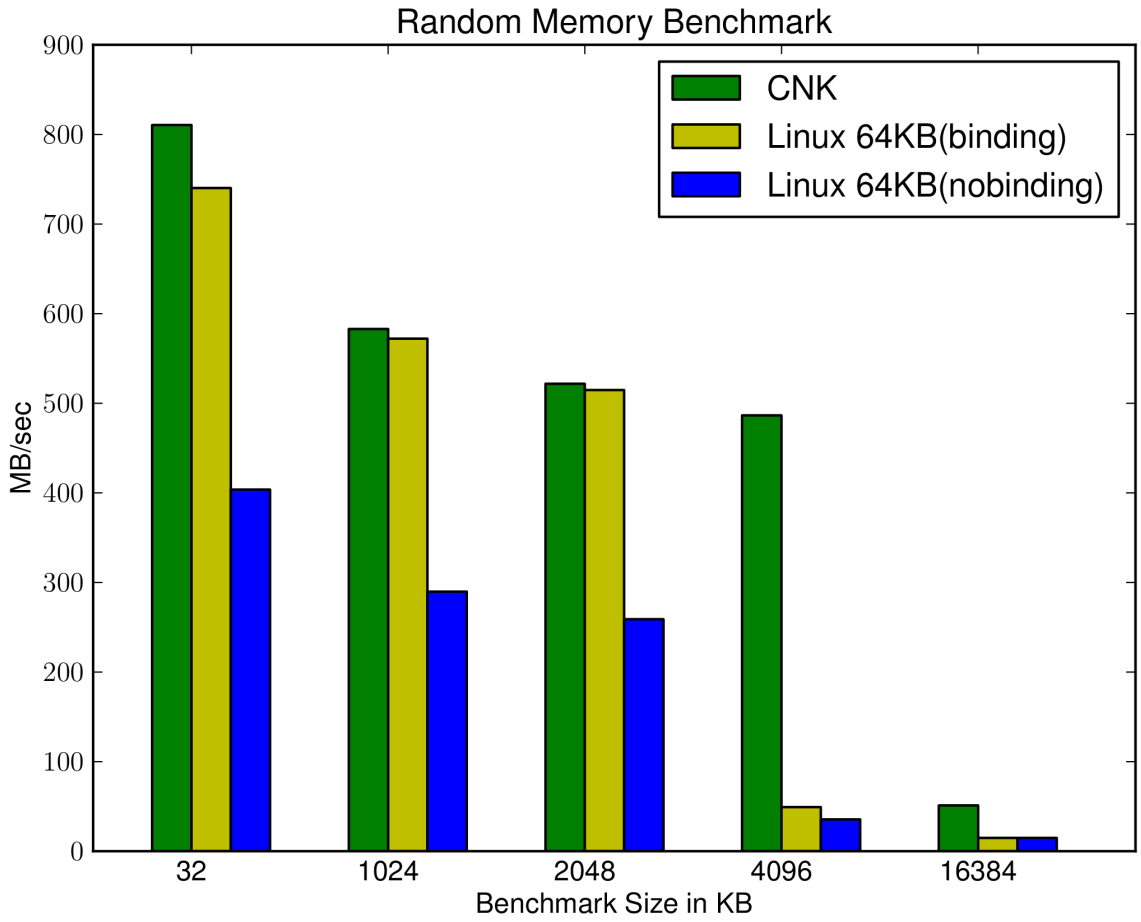
NAS Parallel Benchmark





NOTE: CNK uses MPICH/DCMF while Linux MPICH/TCP (eth-over-torus)

Single Node Performance



Benchmark array

0	3
1	7
2	4
3	1
4	5
5	6
6	0
7	2

What's happening?

- Doesn't seem OS noise is an issue
- Schedule issue: nonbinding -> binding
- **TLB miss**
  - Approx. 0.2 us per miss
    - TLB miss interrupt , a hundred instructions
  - Only 64 TLB entries per core
    - some of TLBs are used by kernel
    - With 64KB, it only covers less than 4MB
  - Doesn't impact streaming access patterns
    - A TLB miss only happen every 64KB
  - Impact a lot on stride or random access patterns
    - Every load/store incurs 0.2 us overhead in the worst case
      - e.g. load L1: 4 cycle, L3: 50, DRAM: 100 cycles

Approaches

- Hugetlbfs
  - Mitigate the issue
  - Does not eliminate TLB miss completely
  - Semi-transparent with libhugetlbfs
- Our approach: Big Memory
  - TLB miss free region to high performance application
    - Co-exist with regular page.
  - Transparent. No API is required
    - modified exec() handler
  - Support BGP DMA, which requires physical contiguous
  - Our previous Big Memory implementation
    - Successfully resolved the issue
    - Only support one process per node

Big Memory implementation

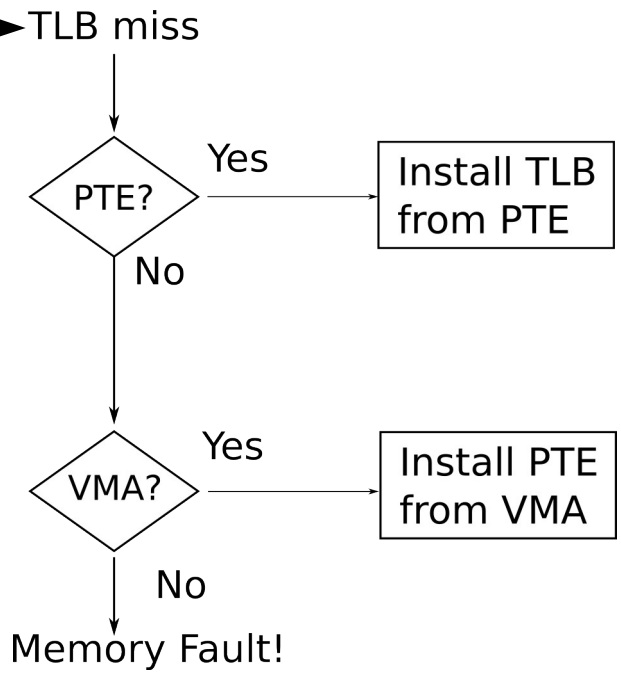
- Boot time allocation
  - Size is adjustable via kernel parameter
  - Kernel is rebooted for every job submission on Blue Gene/P
- Transparency
  - Big Memory process is identified by ELF flag(e_flags field)
  - Kernel exec() loads text, data, initial stack frame into memory
  - Create virtual memory area(VMA) for Big Memory region
  - No special API is required.
    - e.g. mmap() automatically switches to Big Memory
- Initial implementation
  - Unique resource per node (SMP mode in CNK)
  - Applications had to create threads to utilize all core(FPU) resources
- Technical Challenges
  - TLB is not flexible as well as TLB is scarce resource
  - 32-bit address space

Memory Faulting (regular process)

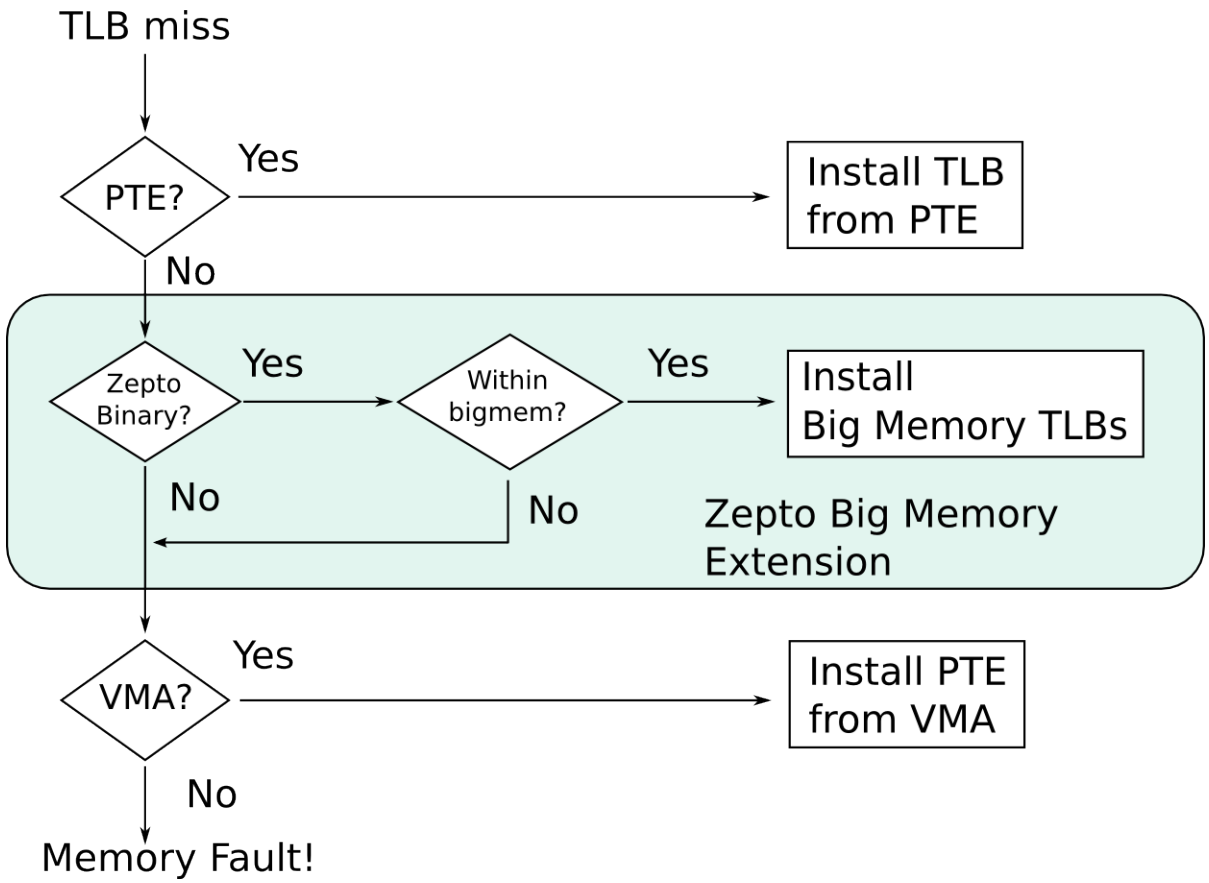
```
unsigned long  *buf;
....
buf = malloc( ... ); // mmap() expands VMA
...
buf[0] = val;  // cause TLB miss

...

buf[1] = val; // no TLB miss (if no CSW happened)
...
```



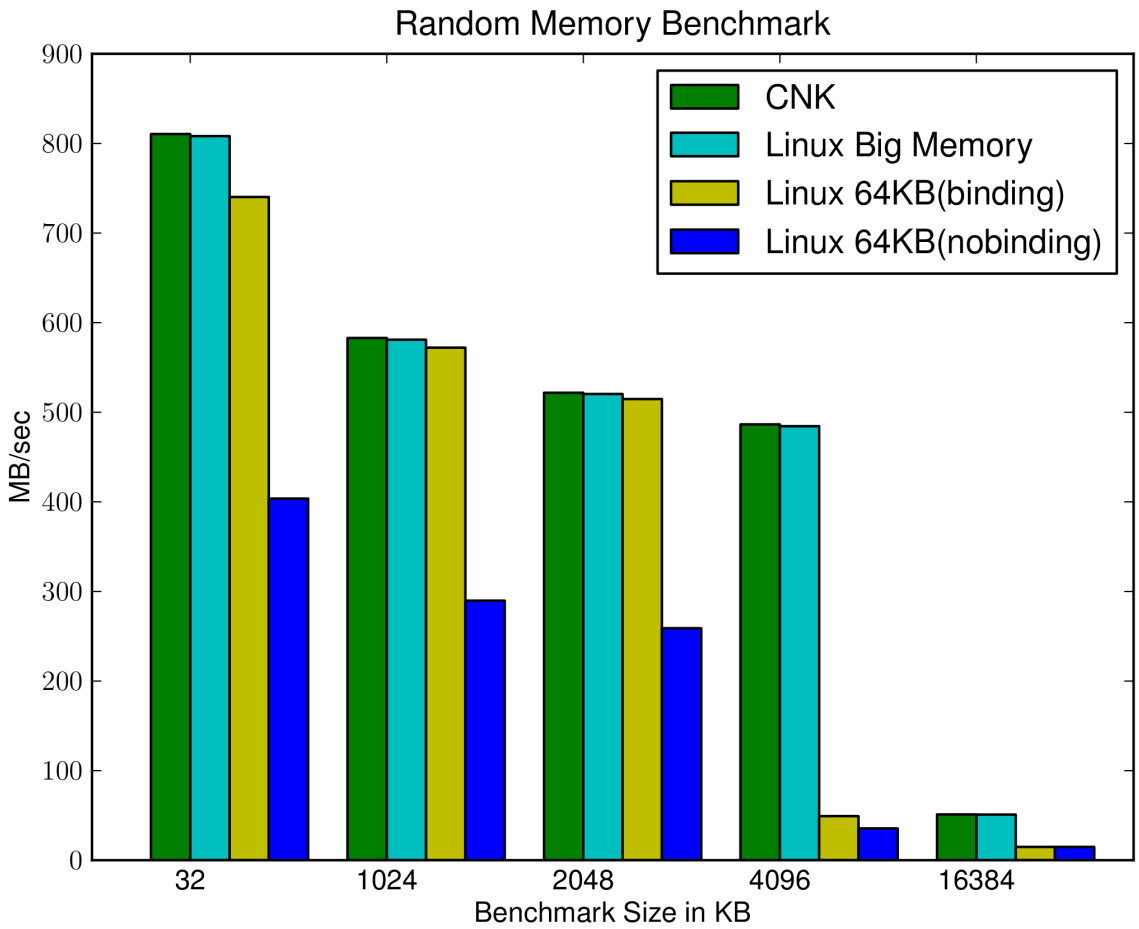
Big Memory Faulting



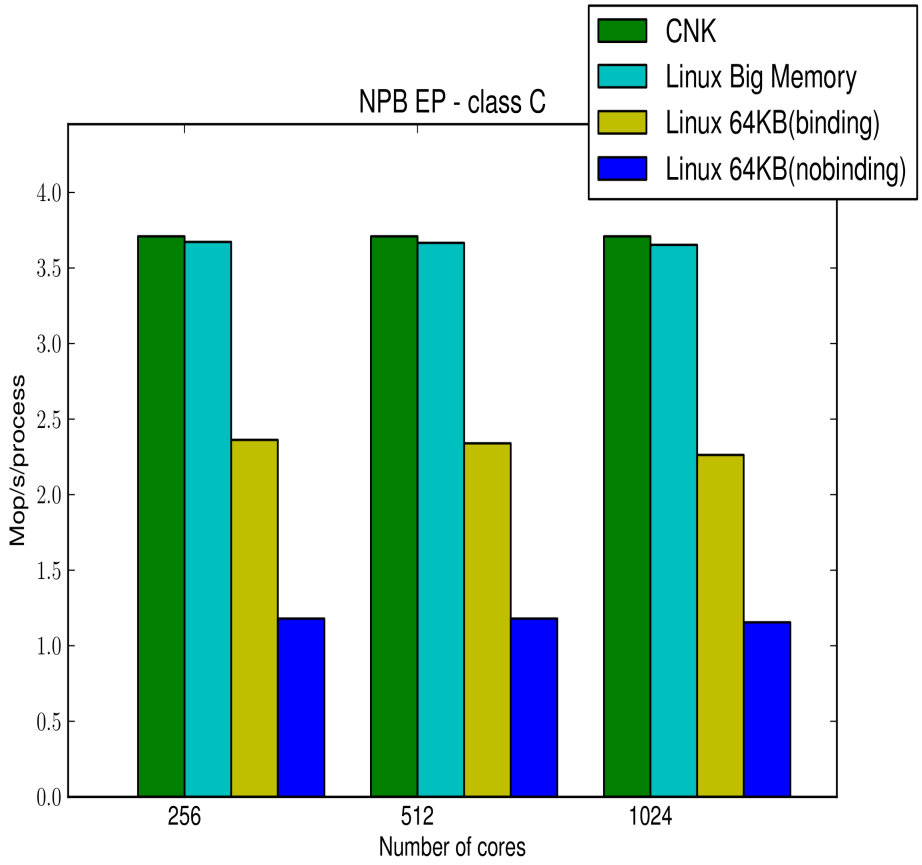
VN mode – four processes per node

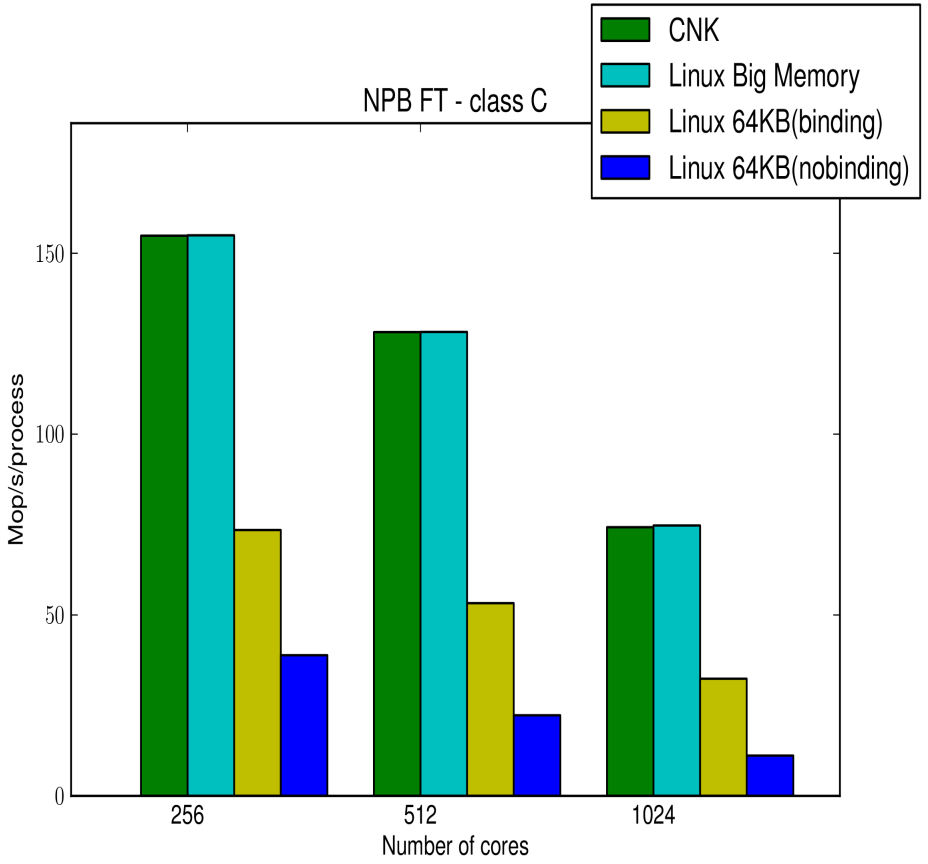
- Resource partition
  - Physical memory allocation
  - Virtual memory manager
    - Big Memory mmap() region
- Core ID to identify Big Memory process ID
- Job launcher(zoid) fork, exec() with setting CPU affinity
  - Unique resource per core
- Modifications on communication stack
  - Modified: Kernel API, SPI
  - Almost no modification: MPICH/DCMF

Memory Benchmark

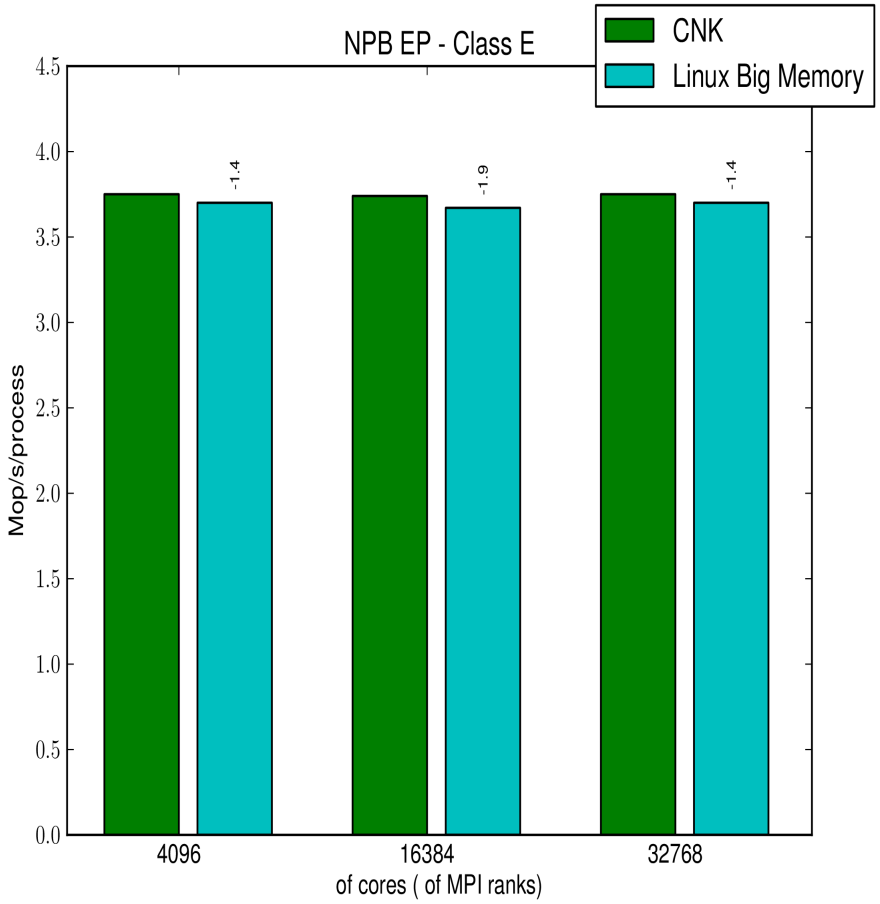


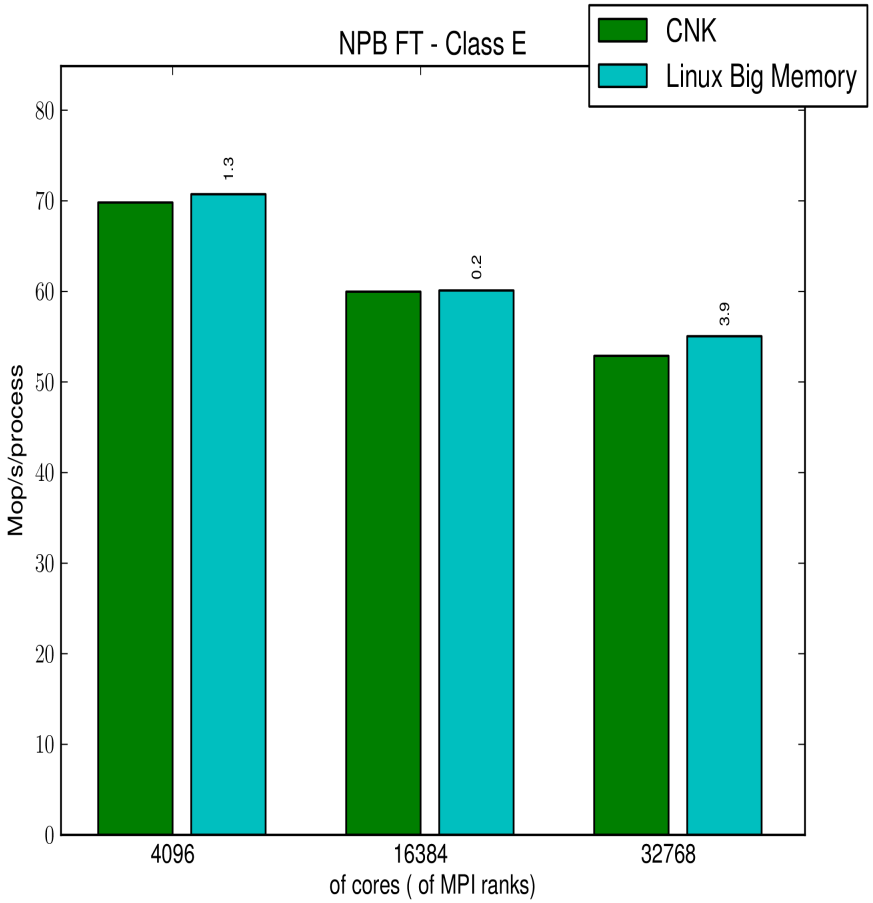
NAS Parallel Benchmark



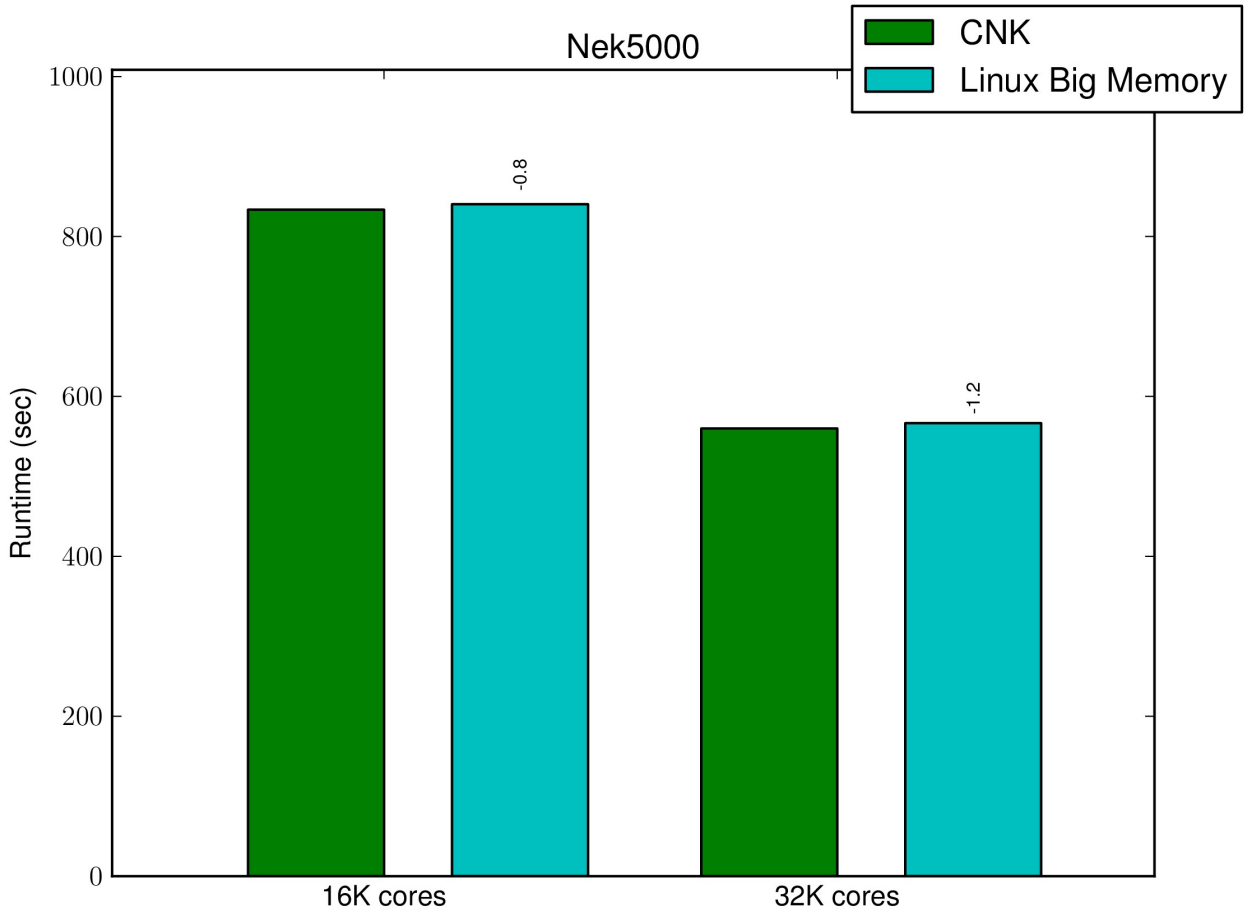


NAS Parallel Benchmark up to 32K cores





Nek5000


Nek5000
CNK
Linux Big Memory
Runtime (sec)
1000
800
600
400
200
0
-0.8
-1.2
16K cores
32K cores

Next Steps

- Merge into kernel.org?
  - very architecture specific and not generalized yet
  - Use main stream feature?
    - transparent hugepages
- Other architecture
  - x86 1GB page
- Next Generation Machine
  - Blue Gene/Q is coming
    - 16 cores (4 SMT) is challenging!
    - 64-bit address space is nice!