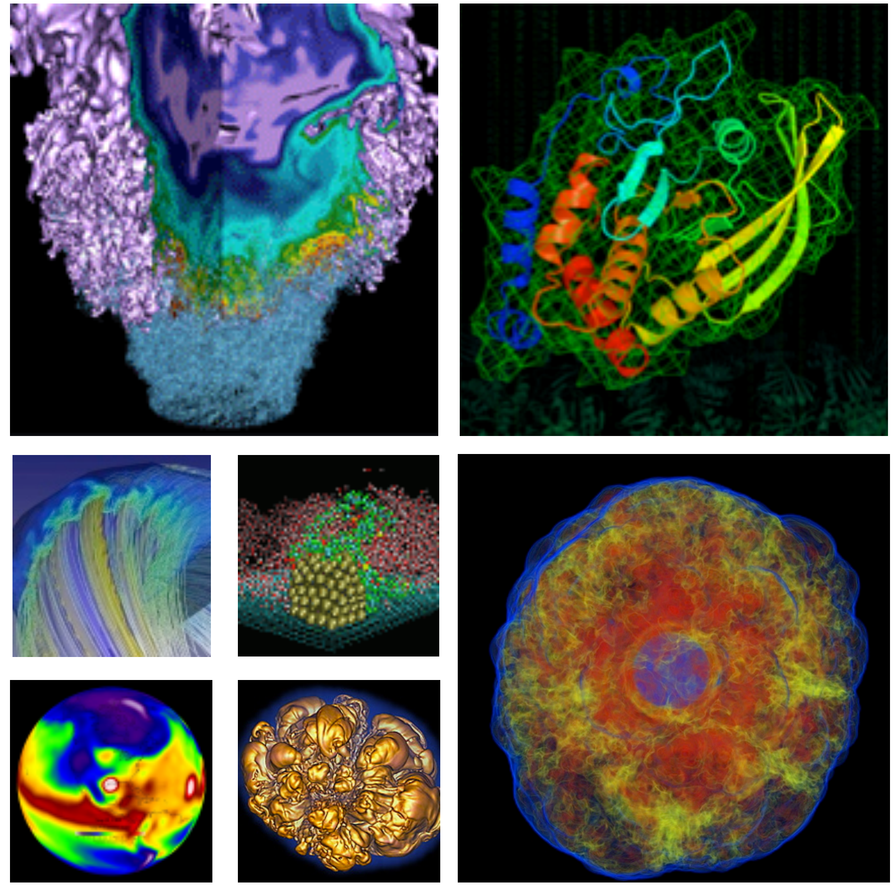
Estimating the Performance Impact of the HBM on KNL Using Dual-Socket Nodes

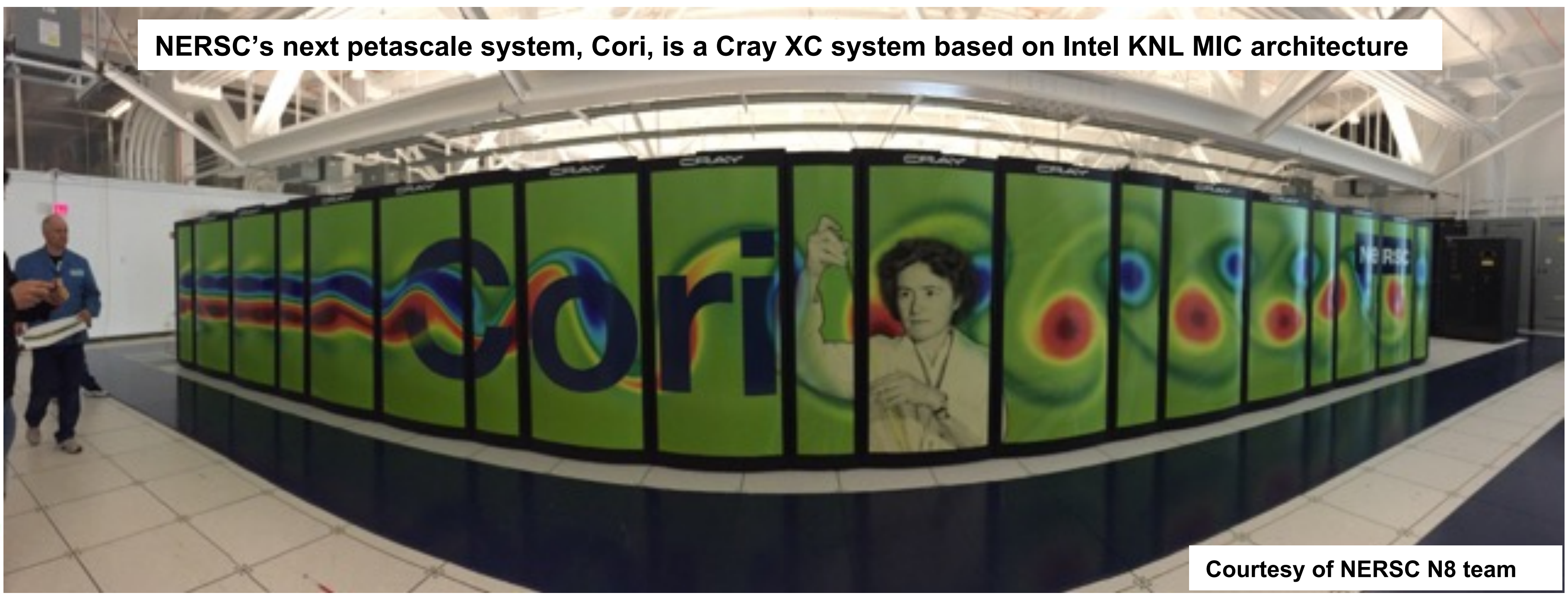


Zhengji Zhao
NERSC User Services Group

SC15 IXPUG BoF, Austin TX,
Nov 18, 2015

Acknowledgement

- Martijn Marsman at Univ. of Vienna
- Jeongnim Kim, Martyn Corden, Christopher Cantalupo, Ruchira Sasanka, Karthik Raman, and Chris Newburn at Intel
- Jack Deslippe at NERSC
- Thank you!



NERSC's next petascale system, Cori, is a Cray XC system based on Intel KNL MIC architecture

Courtesy of NERSC N8 team

MCDRAM and DDR memories available on KNL

- **MCDRAM is significantly higher in bandwidth (HBW) than DDR, efficient use of MCDRAM is important to get most performance out of KNL.**
  - MCDRAM has 5x of DDR memory bandwidth
  - 16 GB MCDRAM and >400 GB DDR memory
- **Using tools provided by Intel, users can test/simulate the benefit of the MCDRAM memory on today's dual socket Xeon nodes.**
  - Use the QPI bus to simulate low bandwidth memory (DDR)
  - This is not an accurate model of the bandwidth and latency characteristics of the KNL on package memory, but is a reasonable way to determine which data structures rely critically on bandwidth.

New libraries and tools available for allocating memory on MCDRAM

- **Memkind, Auto HBW, numactl, hstreams, libhugetlbfs, …**
  - Memkind is a user extensible heap manager.
  - AutoHBW automatically allocate the arrays of certain size to the MCDRAM at run time. No code change is required
- **Application memory footprint < MCDRAM size (numactl is the best option to allocate everything (stack, heap) out of MCDRAM)**
- **Application memory footprint > MCDRAM size**
  - Can do source Modifications (heap allocations: use memkind)
  - Cannot do source modifications (heap allocations : use AutoHBW – allocates based on memory size)
  - Stack allocations ("currently" can use only numactl , can use "—preferred" option for partial MCDRAM allocations)
- **Intel VTune (memory-access analysis) could be used to identify the candidates for MCDRAM.**

New libraries and tools available for allocating memory on MCDRAM

- **Memkind, Auto HBW, numactl, hstreams, libhugetlbfs, …**
  - Memkind is a user extensible heap manager.
  - AutoHBW automatically allocate the arrays of certain size to the MCDRAM at run time. No code change is required
- **Application memory footprint < MCDRAM size (numactl is the best option to allocate everything (stack, heap) out of MCDRAM)**
- **Application memory footprint > MCDRAM size**
  - Can do source Modifications (heap allocations: use memkind)
  - Cannot do source modifications (heap allocations : use AutoHBW – allocates based on memory size)
  - Stack allocations ("currently" can use only numactl , can use "—preferred" option for partial MCDRAM allocations)
- **Intel VTune (memory-access analysis) could be used to identify the candidates for MCDRAM.**

Please sign up the new memory types IXPUG working group at ixpug.org

Using Memkind library on NERSC's Edison, a Cray XC30 based on the dual-socket Ivy Bridge nodes

- **Add compiler directive !DIR ATTRIBUTES FASTMEM in Fortran codes**
  - real, allocatable :: a(:,:), b(:,:), c(:)
  - !DIR$ ATTRIBUTES FASTMEM :: a, b, c
- **Use hbw_malloc, hbw_calloc to replace the malloc, calloc in the C/C++ codes**
  - #include <hbwmalloc.h>
  - malloc(size) -> hbw_malloc(size)
- **Link the codes to the memkind and jemalloc libraries**
  - module load memkind
  - ftn -dynamic -g -O3 -openmp mycode.f90

    # compiler wrappers link the code to the –lmemkind –ljemalloc libraries.
- **Run the codes with the numactl and env MEMKIND_HBW_NODES**
  - module load memkind # only needed for dynamically linked apps
  - export MEMKIND_HBW_NODES=0
  - aprun -n 1 -cc numa_node numactl --membind=1 --cpunodebind=0 ./a.out

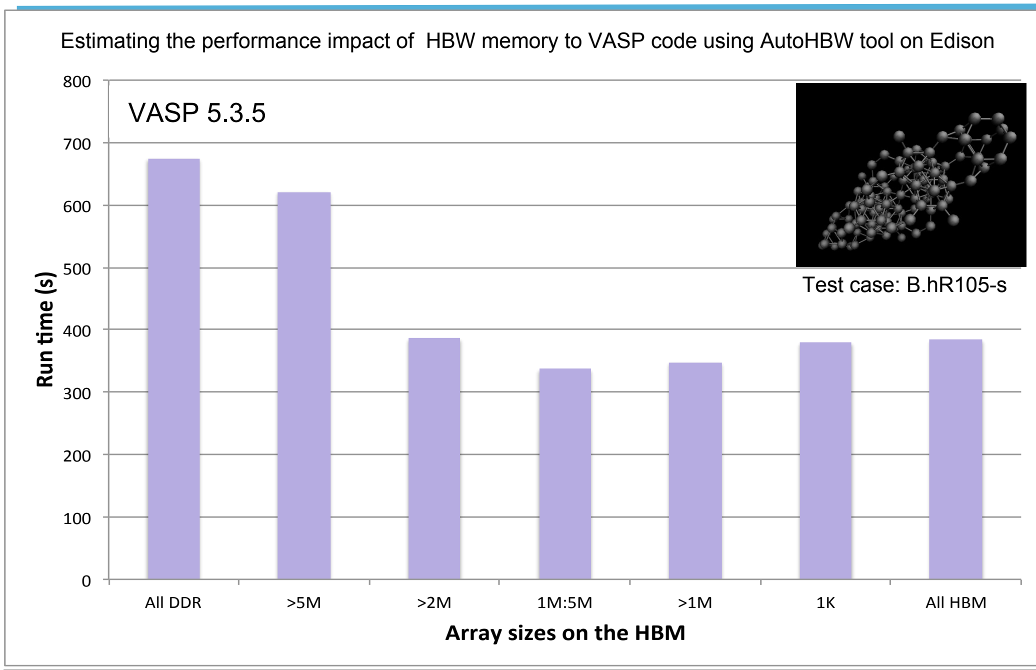
Using AutoHBW tool on the dual-socket, Ivy Bridge nodes on Edison

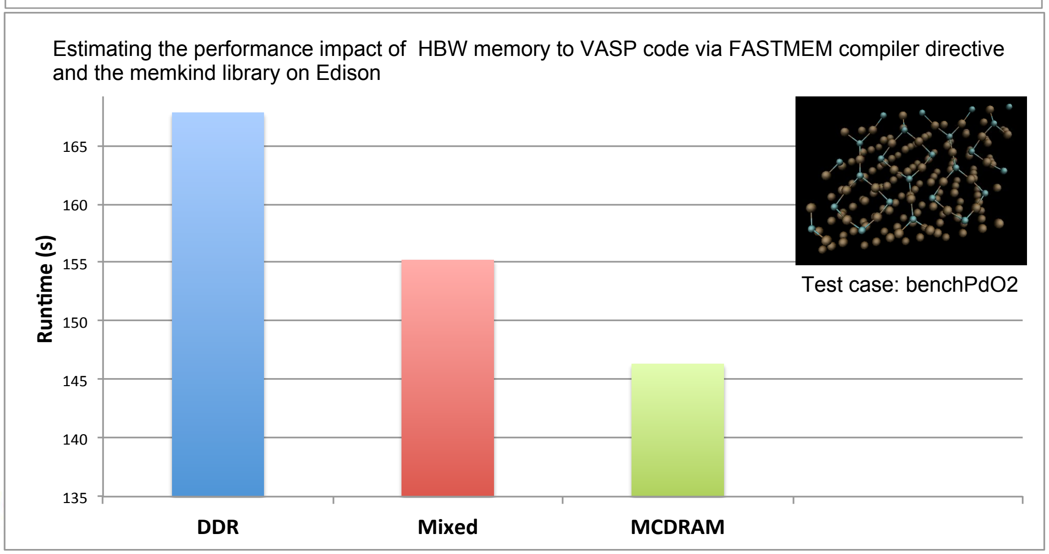
- **Link the codes to the autohbw, memkind and jemalloc libraries**
  - module load autohbw
  - ftn -g -O3 -openmp mycode.f90
    # this will link to the autohbw, memkind, and jemalloc libraries automatically
- **Run the codes with the numactl and proper environment variables**
  - export MEMKIND_HBW_NODES=0
  - export AUTO_HBW_LOG=0
  - export AUTO_HBW_MEM_TYPE=MEMKIND_HBW
  - export AUTO_HBW_SIZE=5K # all allocation larger than 5K allocated in HBM
  - export AUTO_HBW_SIZE=1K:5K
    # all allocations between sizes 1K and 5K allocated in HBW memory
  - aprun –n 1 –cc numa_node numactl --membind=1 --cpunodebind=0 ./a.out

Examples:

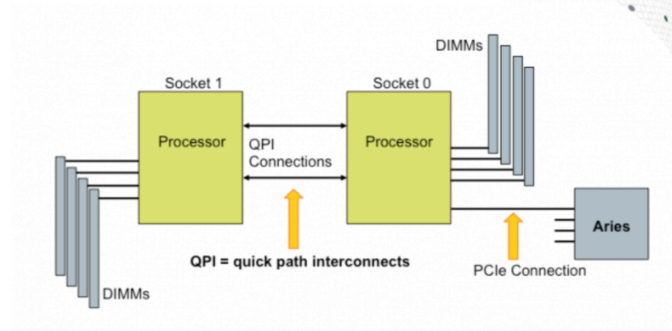
AUTO_HBW_MEM_TYPE=MEMKIND_HBW (Default)
AUTO_HBW_MEM_TYPE=MEMKIND_HBW_HUGETLB
AUTO_HBW_MEM_TYPE=MEMKIND_HUGETLB

Estimating HBW memory performance impact to application codes using dual-socket Ivy Bridge nodes on Edison as proxy to KNL





Edison compute node diagram



Edison, a Cray XC30, with dual-socket Ivy Bridge nodes interconnected with Cray's Aries network, the bandwidths of the near socket memory (simulating MCDRAM) and the far socket memory via QPI (simulating DDR) differ by 33%

VASP is a material science code that consumes the most computing cycles at NERSC.

This test used a development version of the VASP code.

Adding the FASTMEM directives to the code was done by Martijn Marsman at Vienna University

References

- **Memkind and Auto HBW tool**
  - http://memkind.github.io/memkind
  - http://memkind.github.io/memkind/memkind_arch_20150318.pdf
  - http://ihpcc2014.com/pdf/100_KNL_HPC_Developer_Forum_SC_14.pdf
- **Edison**
  - http://www.nersc.gov/users/computational-systems/edison/
- **VASP**
  - VASP: http://www.vasp.at/
  - G. Kresse and J. Furthm_ller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. Comput. Mat. Sci., 6:15, 1996