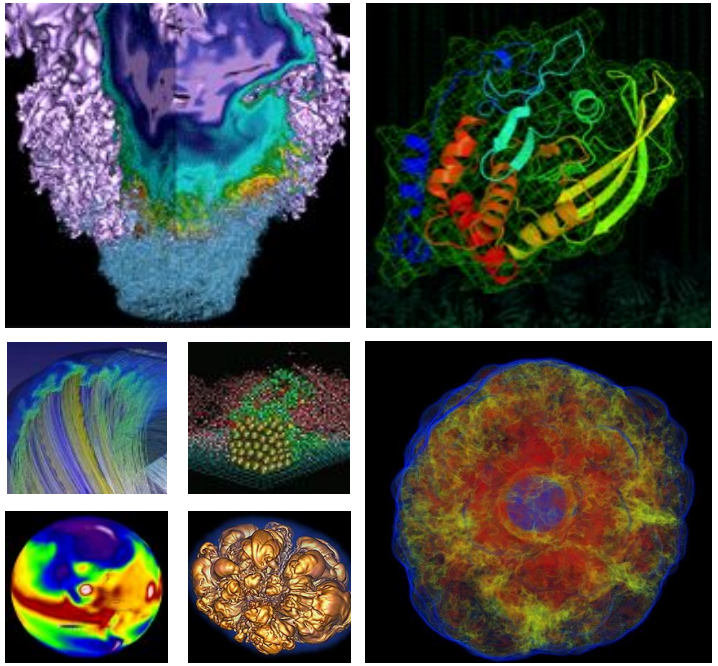
Cori Application Readiness Strategy and Early Experiences



NERSC

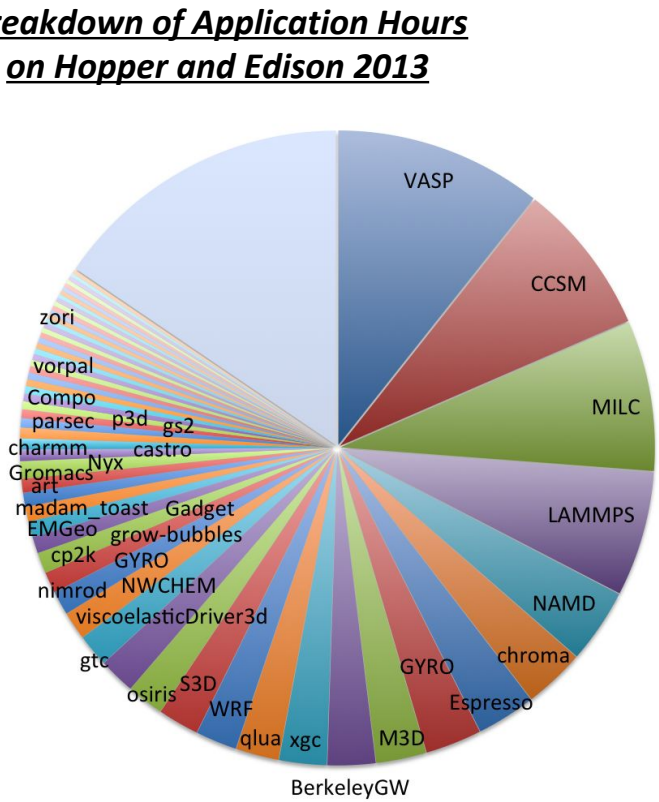
March, 2016

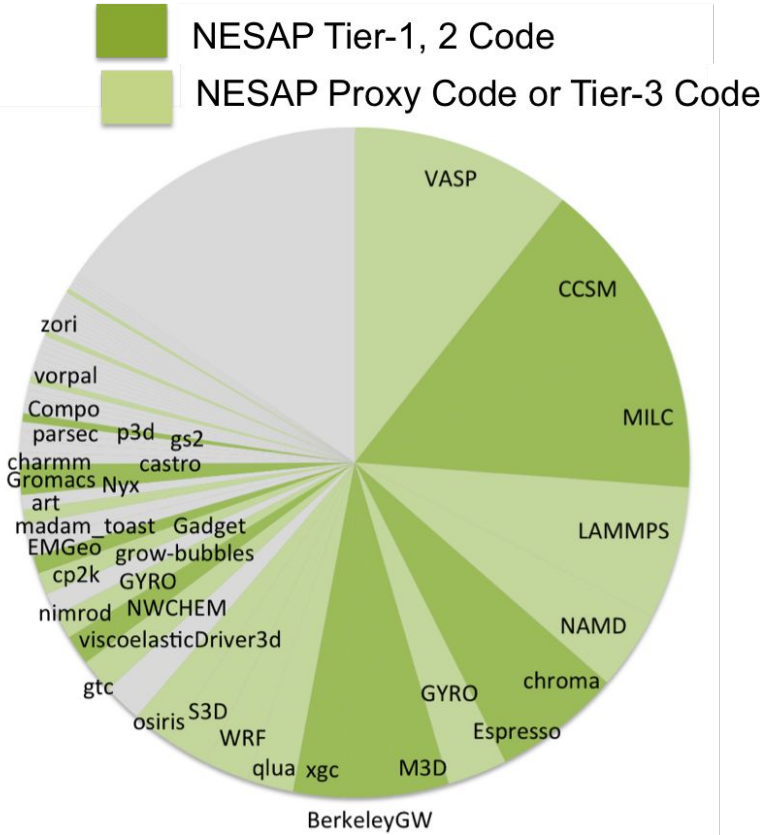
U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Code Coverage

Breakdown of Application Hours on Hopper and Edison 2013





Resources for Code Teams

- **Early access to hardware**
  - Access to Babbage (KNC cluster) and early “white box” test systems expected in 2015
  - Early access and significant time on the full Cori system
- **Technical deep dives**
  - Access to Cray and Intel staff on-site staff for application optimization and performance analysis
  - Multi-day deep dive (‘dungeon’ session) with Intel staff at Oregon Campus to examine specific optimization issues
- **User Training Sessions**
  - From NERSC, Cray and Intel staff on OpenMP, vectorization, application profiling
  - Knights Landing architectural briefings from Intel
- **NERSC Staff as Code Team Laisons (Hands on assistance)**
- **8 Postdocs**

NESAP Postdocs

Taylor Barnes
Quantum ESPRESSO

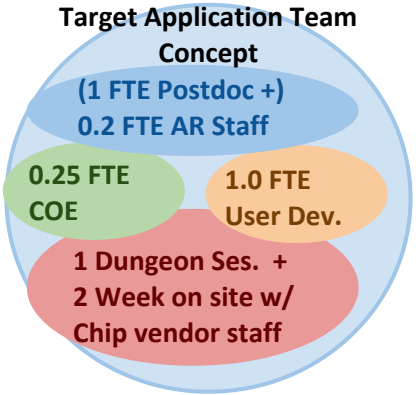
Brian Friesen
Boxlib

Andrey Ovsyannikov
Chombo-Crunch

Mathieu Lobet
WARP

Tuomas Koskela
XGC1

Tareq Malas
EMGeo



NERSC Staff associated with NESAP

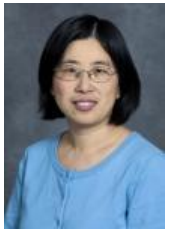

Katie Antypas


Nick Wright


Richard Gerber

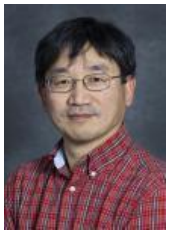

Brian Austin


Zhengji Zhao


Helen He


Ankit Bhagatwala


Stephen Leak


Woo-Sun Yang

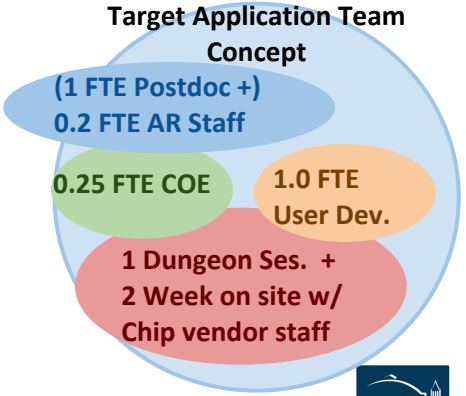

Rebecca Hartman-Baker


Doug Doerfler

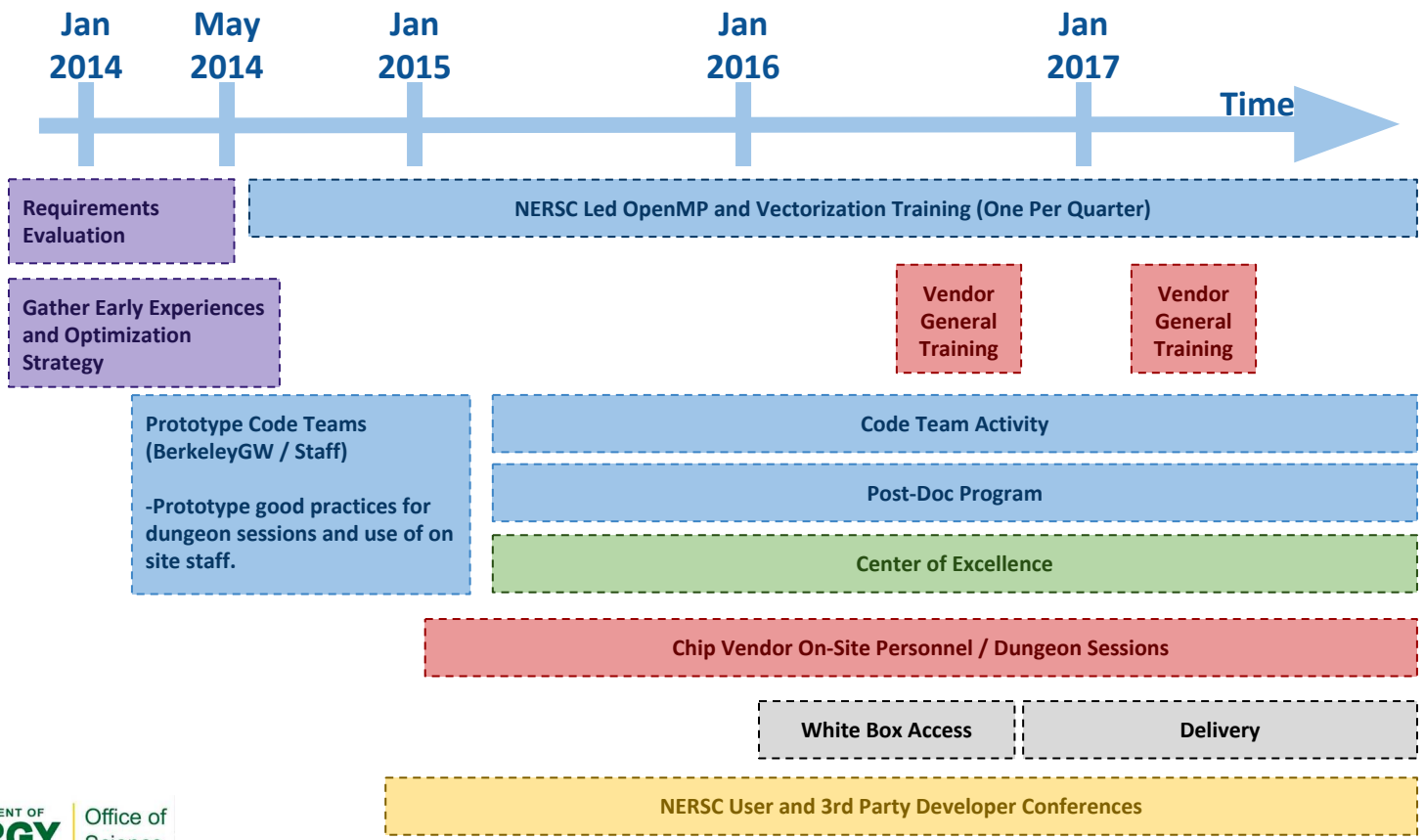

Jack Deslippe


Brandon Cook

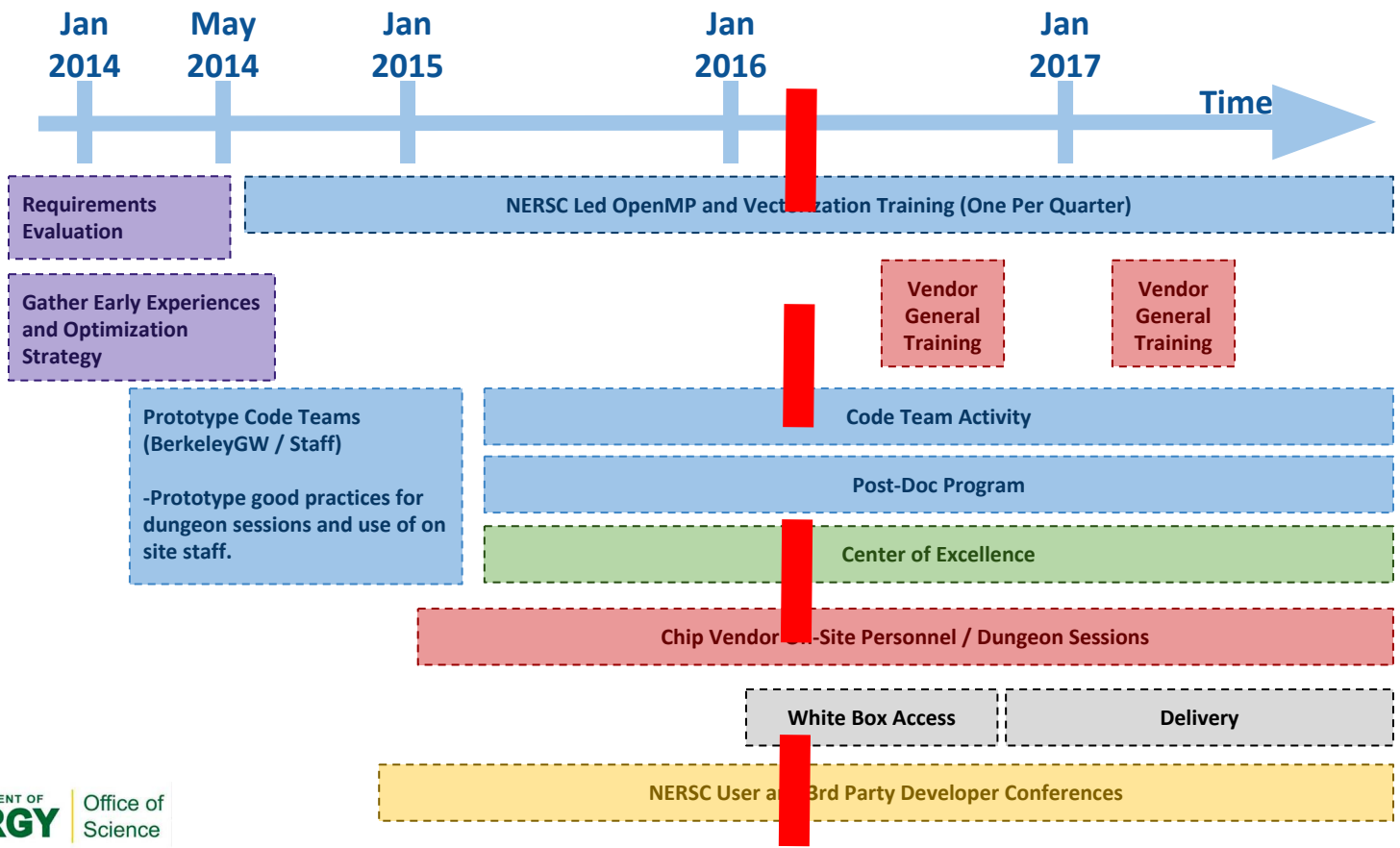

Thorsten Kurth



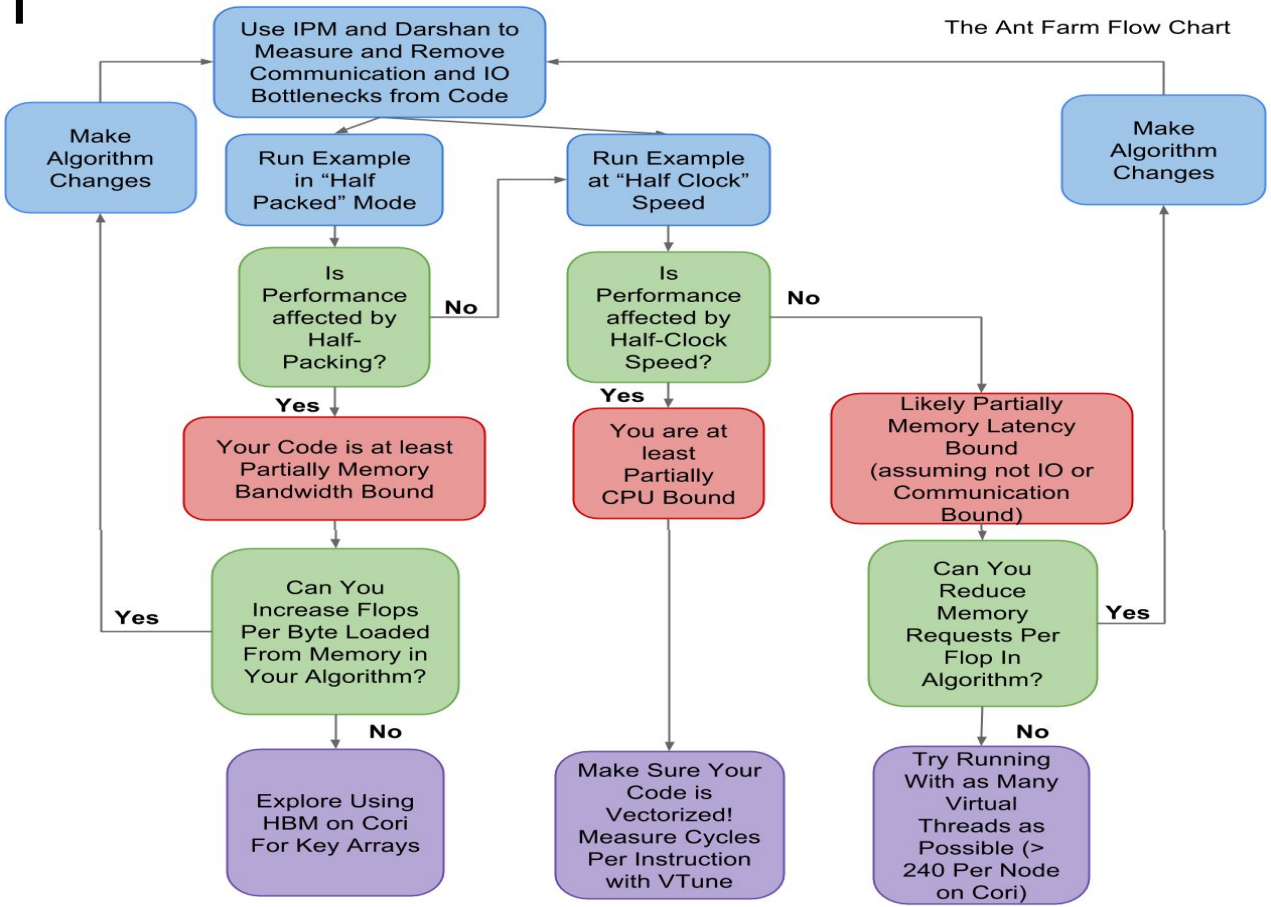
Timeline


Jan 2014
May 2014
Jan 2015
Jan 2016
Jan 2017
Time
Requirements Evaluation
NERSC Led OpenMP and Vectorization Training (One Per Quarter)
Gather Early Experiences and Optimization Strategy
Vendor General Training
Vendor General Training
Prototype Code Teams (BerkeleyGW / Staff)
-Prototype good practices for dungeon sessions and use of on site staff.
Code Team Activity
Post-Doc Program
Center of Excellence
Chip Vendor On-Site Personnel / Dungeon Sessions
White Box Access
Delivery
NERSC User and 3rd Party Developer Conferences

Timeline


Jan 2014
May 2014
Jan 2015
Jan 2016
Jan 2017
Time
Requirements Evaluation
NERSC Led OpenMP and Vectorization Training (One Per Quarter)
Gather Early Experiences and Optimization Strategy
Vendor General Training
Vendor General Training
Prototype Code Teams (BerkeleyGW / Staff)
-Prototype good practices for dungeon sessions and use of on site staff.
Code Team Activity
Post-Doc Program
Center of Excellence
Chip Vendor On-Site Personnel / Dungeon Sessions
White Box Access
Delivery
NERSC User and 3rd Party Developer Conferences

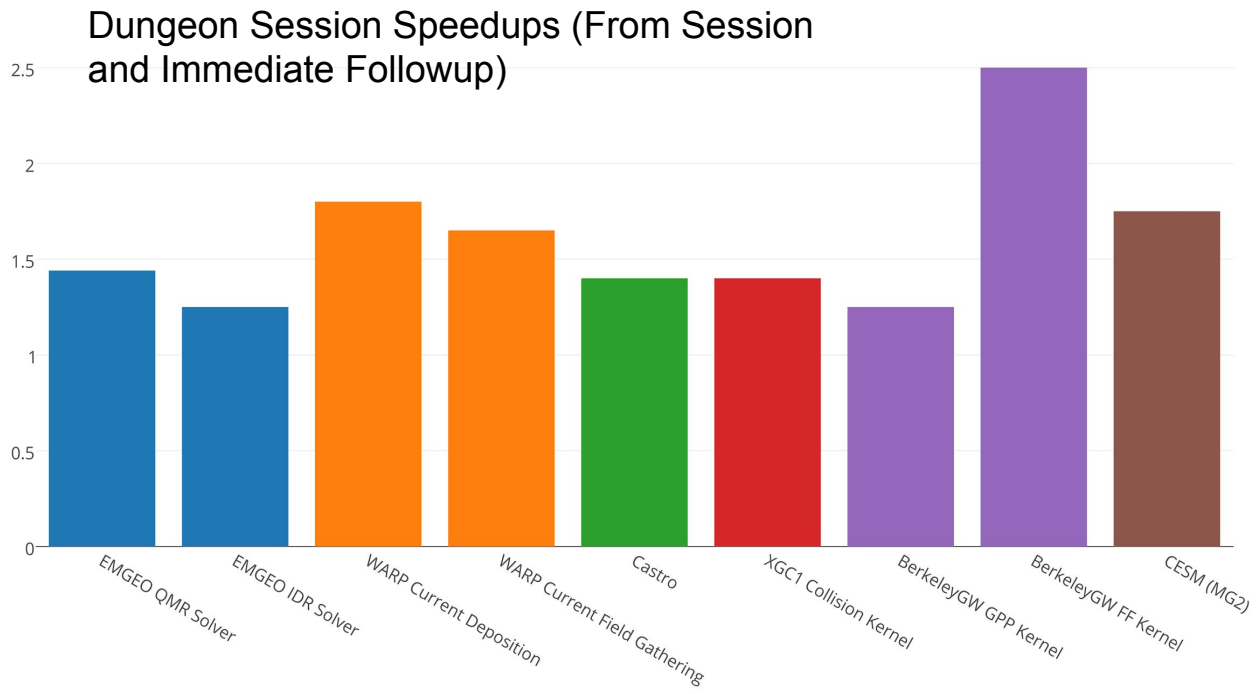
Dungeon Prep



Working With Vendors

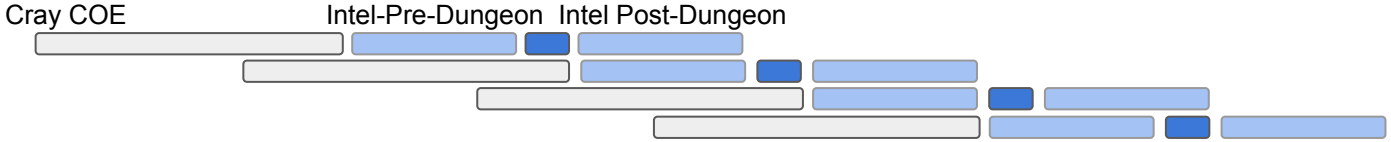
NERSC Is uniquely positioned between HPC Vendors and HPC Users and Applications developers.

NESAP provides a power venue for these two groups to interact.

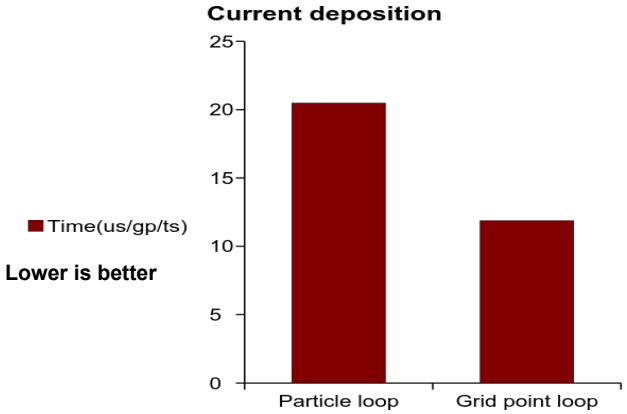


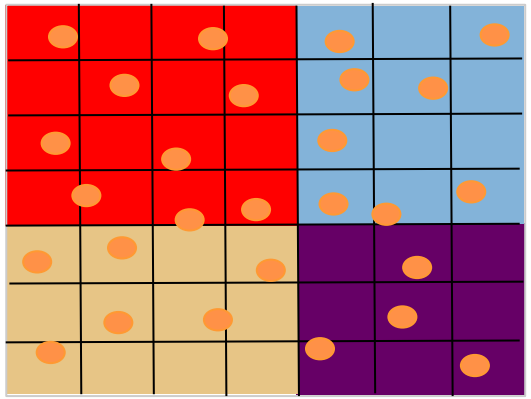
What Has Gone Well

1. Setting requirements for Dungeon Session motivates teams to get started early and improves quality of dungeon session.
2. Engagement with IXPUG and user communities (Exascale Workshops at CRT)
3. Large number of NERSC and Vendor Training (Vectorization, OpenMP, Tools/Compilers) Well Received
4. Learned a Massive Amount about Tools and Architecture (VTune, SDE, HBM etc.)
5. Vendor staff helpful to work with. Very pro-active.
6. Pipelining Code Work Via Cray and Intel resources



Warp Vectorization Improvements at The Dungeon - Directly enabled by tiling work with Cray COE in Pre-dungeon



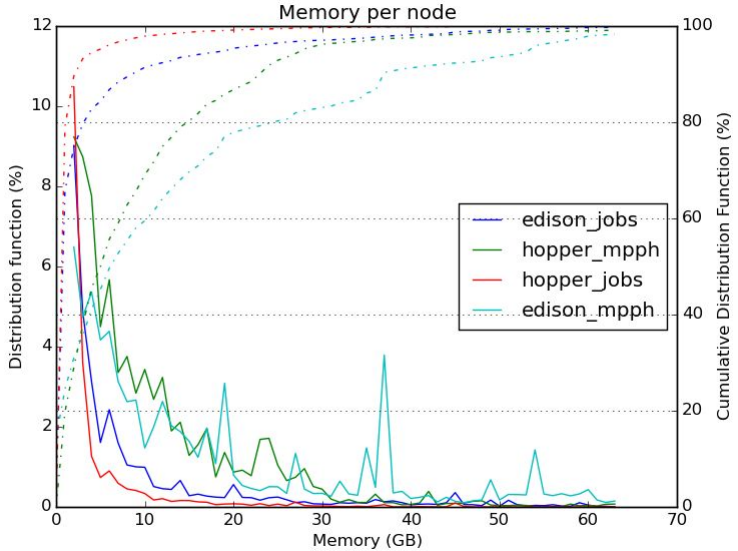


Techniques and Tools to Target Arrays for Fastmem:

Application	All memory on far memory	All memory on near memory	Key arrays on near memory
BerkeleyGW	baseline	52% faster	52.4% faster
EmGeo	baseline	40% faster	32% faster
XGC1	baseline		24% faster

What Has Gone Well (Cont)

7. Bandwidth sensitive applications that live in HBM expected to perform very well.



The N9 workload analysis shows a large fraction of jobs use < 16GB of memory per node

8. A lot of Lessons Learned: techniques to place key-arrays in fast-memory, improve prefetching effectiveness, coping without L3 cache etc...

9. CPU Intensive tasks (BGW GPP Kernel) expected to perform well (> Haswell) on KNL.

10. Postdocs deeply engaged.

CESM (NERSC LEAD Helen He)

Version 1

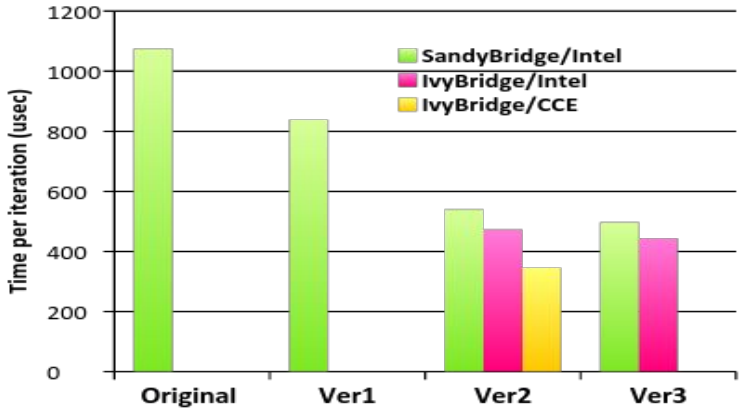
- Simplify expressions to minimize #operations
- Use internal GAMMA function

Version 2

- Remove “elemental” attribute, move loop inside.
- Inline subroutines. Divide, fuse, exchange loops.
- Replace assumed shape arrays with loops
- Replace division with inversion of multiplication
- Remove initialization of loops to be overwritten later
- Use more aggressive compiler flags
- Use profile-guided optimization (PGO)

Version 3 (Intel compiler only)

- Use !$OMP SIMD ALIGNED to force vectorization



XGC1 (NERSC Lead Helen He Ankit Bhagatwala)

Original

```
real(8),dimension
   (5,(col_f_nvr-1)*(col_f_nvz-1),
   (col_f_nvr-1)*(col_f_nvz-1)) :: Ms

do index_ip = 1, mesh_Nzm1
  do index_jp = 1, mesh_Nrm1
    index_2dp = index_jp+mesh_Nrm1*(index_ip-1)

    tmp_vol = cs2%local_center_volume(index_jp)
    tmp_f_half_v = f_half(index_jp, index_ip) *
   tmp_vol
    tmp_dfdr_v = dfdr(index_jp, index_ip) *
   tmp_vol
    tmp_dfdz_v = dfdz(index_jp, index_ip) *
   tmp_vol

    tmpr(1:3)= tmpr(1:3)+
   Ms(1:3,index_2dp,index_2D)* tmp_f_half_v
    tmpr(5)  = tmpr(5)  +
   Ms(4,index_2dp,index_2D)*tmp_dfdr_v +
```

Optimized

```
real (8),dimension
    ((col_f_nvr-1),5,(col_f_nvz-1),
    (col_f_nvr-1)*(col_f_nvz-1)) :: Ms

do index_ip = 1, mesh_Nzm1
  do index_jp = 1, mesh_Nrm1
    index_2dp = index_jp+mesh_Nrm1*(index_ip-1)
    tmp_vol = cs2%local_center_volume(index_jp)
    tmp_f_half_v = f_half(index_jp, index_ip) *
    tmp_vol
    tmp_dfdr_v = dfdr(index_jp, index_ip) * tmp_vol
    tmp_dfdz_v = dfdz(index_jp, index_ip) *  tmp_vol

    tmpr(index_jp,1) = tmpr(index_jp,1) +
    Ms(index_jp,1,index_ip,index_2D)*
    tmp_f_half_v
    tmpr(index_jp,2) = tmpr(index_jp,2) +
    Ms(index_jp,2,index_ip,index_2D)*
    tmp_f_half_v
    tmpr(index_jp,3) = tmpr(index_jp,3) +
    Ms(index_jp,3,index_ip,index_2D)*
    tmp_f_half_v
    tmpr(index_jp,5) = tmpr(index_jp,5) +
    Ms(index_jp,4,index_ip,index_2D)*          tmp_dfdr_v
+   Ms(index_jp,2,index_ip,index_2D)*          tmp_dfdz_v
```

Example From Cray COE Work on XGC1

~40% speed up for kernel

XGC1 (NERSC Lead Helen He Ankit Bhagatwala)

Example From Cray COE Work on XGC1

~40% speed up for kernel

Original

```
real(8),dimension
  (5,(col_f_nvr-1)*(col_f_nvz-1),
   (col_f_nvr-1)*(col_f_nvz-1)) :: Ms

do index_ip = 1, mesh_Nzm1
  do index_jp = 1, mesh_Nrm1
    index_2dp = index_jp+mesh_Nrm1*(index_ip-1)

    tmp_vol = cs2%local_center_volume(index_jp)
    tmp_f_half_v = f_half(index_jp, index_ip) *
   tmp_vol
    tmp_dfdr_v = dfdr(index_jp, index_ip) *
   tmp_vol
    tmp_dfdz_v = dfdz(index_jp, index_ip) *
   tmp_vol

    tmpr(1:3)= tmpr(1:3)+
   Ms(1:3,index_2dp,index_2D)* tmp_f_half_v
    tmpr(5)  = tmpr(5)  +
   Ms(4,index_2dp,index_2D)*tmp_dfdr_v +
```

Optimized

```
real (8),dimension
   ((col_f_nvr-1),5,(col_f_nvz-1),
   (col_f_nvr-1)*(col_f_nvz-1)) :: Ms

do index_ip = 1, mesh_Nzm1
  do index_jp = 1, mesh_Nrm1
    index_2dp = index_jp+mesh_Nrm1*(index_ip-1)
    tmp_vol = cs2%local_center_volume(index_jp)
    tmp_f_half_v = f_half(index_jp, index_ip) *
   tmp_vol
    tmp_dfdr_v = dfdr(index_jp, index_ip) * tmp_vol
    tmp_dfdz_v = dfdz(index_jp, index_ip) *  tmp_vol

    tmpr(index_jp,1) = tmpr(index_jp,1) +
   Ms(index_jp,1,index_ip,index_2D)*
   tmp_f_half_v
    tmpr(index_jp,2) = tmpr(index_jp,2) +
   Ms(index_jp,2,index_ip,index_2D)*
   tmp_f_half_v
    tmpr(index_jp,3) = tmpr(index_jp,3) +
   Ms(index_jp,3,index_ip,index_2D)*
   tmp_f_half_v
    tmpr(index_jp,5) = tmpr(index_jp,5) +
   Ms(index_jp,4,index_ip,index_2D)*          tmp_dfdr_v
+  Ms(index_jp,2,index_ip,index_2D)*          tmp_dfdz_v
```

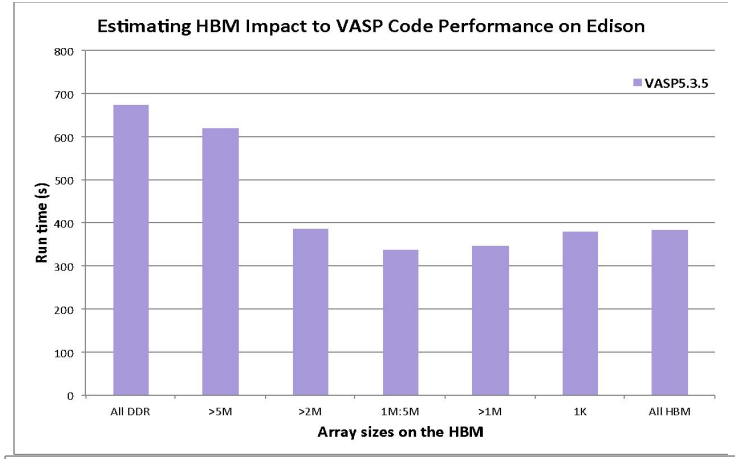
EMGEO (NERSC LEAD Scott French, Thorsten Kurth, Tareq Malas)

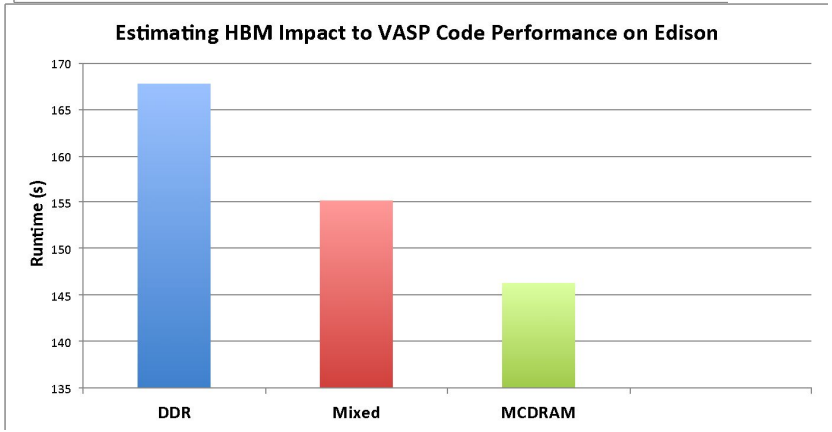
```
subroutine ell_spmv(mat, ind, x, z, m, n, ndiag)
  implicit none
  ! --
  integer :: m, n, ndiag
  integer, dimension(ndiag, m) :: ind
  complex*16, dimension(n) :: x
  complex*16, dimension(m) :: z
  complex*16, dimension(ndiag, m) :: mat
  ! --
  integer :: i, j
  complex*16 :: ztmp
!$omp parallel do private(ztmp)
  do i = 1, m
    ztmp =  (0.0d0, 0.0d0)
    do j = 1, ndiag
      ztmp = ztmp + mat(j,i) * x(ind(j,i))
    end do
    z(i) = ztmp
  end do
```

```
!$omp parallel do private(ztmp)
  do i = 1, 2 * nx * ny
    ztmp =  (0.0d0, 0.0d0)
    do j = 1, ndiag
      ztmp = ztmp + mat(i,j) * x(ind(i,j))
    end do
    z(i) = ztmp
  end do
!$omp parallel do private(ztmp)
  do i = 2 * nx * ny + 1, m -  nx * ny
    ztmp =  (0.0d0, 0.0d0)
    ! stride 1
    ztmp = ztmp + mat(i,1 ) * x(i - 2)
    ztmp = ztmp + mat(i,2 ) * x(i - 1)
    ztmp = ztmp + mat(i,3 ) * x(i)
    ztmp = ztmp + mat(i,4 ) * x(i + 1)
    ! stride nx
    ztmp = ztmp + mat(i,5 ) * x(i - 2 * nx)
    ztmp = ztmp + mat(i,6 ) * x(i - nx)
    ztmp = ztmp + mat(i,7 ) * x(i)
    ztmp = ztmp + mat(i,8 ) * x(i + nx)
    ! stride nx * ny
    ztmp = ztmp + mat(i,9 ) * x(i - 2 * nx * ny)
    ztmp = ztmp + mat(i,10) * x(i - nx * ny)
    ztmp = ztmp + mat(i,11) * x(i)
    ztmp = ztmp + mat(i,12) * x(i + nx * ny)
    z(i) = ztmp
  end do
```

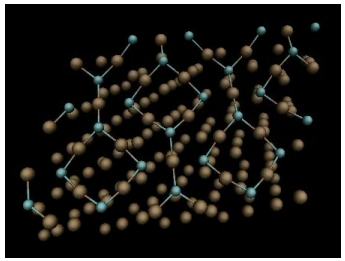
Vector loads when vectorized in `i`

VASP (NERSC LEAD Zhengji Zhao)





Estimating the performance impact of HBW memory to VASP code via FASTMEM compiler directive and the memkind library on Edison



Test case: benchPdO2

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

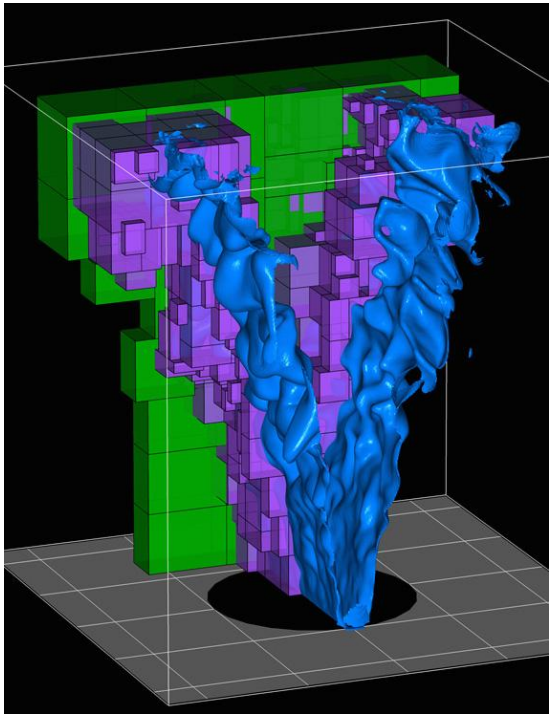
Boxlib (NERSC LEAD Brian Friesen)

Block AMR Framework.

Added “tiling” to improve data locality and improve OMP scaling on Xeon-Phi.

SAMPLE WALL TIME MEASUREMENTS (SEC)

0	Iteration #	Outer-loop-level threading	Loop tiling
1	1	3.76	0.58
2	2	3.87	0.69
3	3	3.86	0.70
4	4	3.86	0.69
5	5	3.84	0.68



Now exploring in transit analysis using specialized analysis ranks and burst buffer.

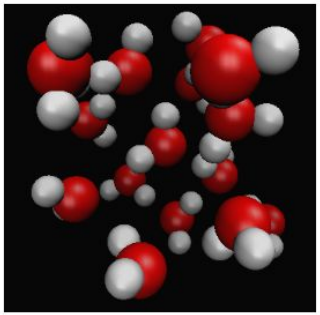
Quantum ESPRESSO (NERSC Lead Taylor Barnes / Jack Deslippe)

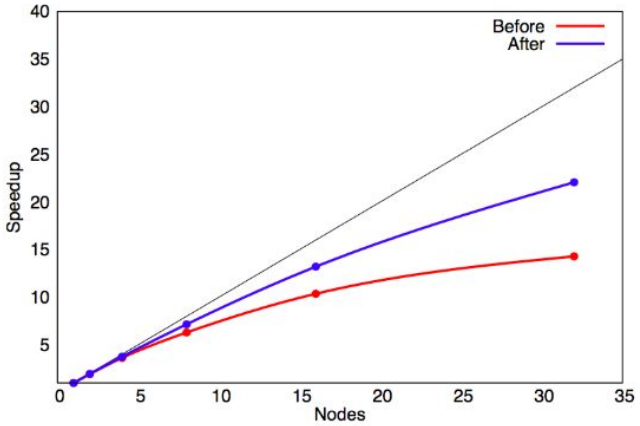
How improve a code where most FLOPs occur in libraries?

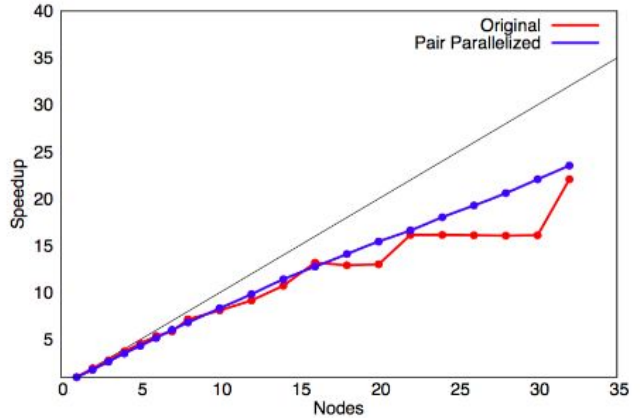
Targeting Exact Exchange Problems. Characterized by many parallel FFTs.

Strategy:

Improve on-node performance by
increase the on-noe FLOP density
and reducing inter-node communication.
Moving individual FFTs to a single node shared memory model and exploiting new band-pair parallelism with MPI.







Quantum ESPRESSO (NERSC Lead Taylor Barnes / Jack Deslippe)

How improve a code where most FLOPs occur in libraries?

Targeting Exact Exchange Problems. Characteriz many parallel FFTs.

Strategy:

Improve on-node performance by
increase the on-noe FLOP density
and reducing inter-node communication.
Moving individual FFTs to a single node shared memory model and exploiting new band-pair parallelism with MPI.

Exploit parallelism not used by default in app.