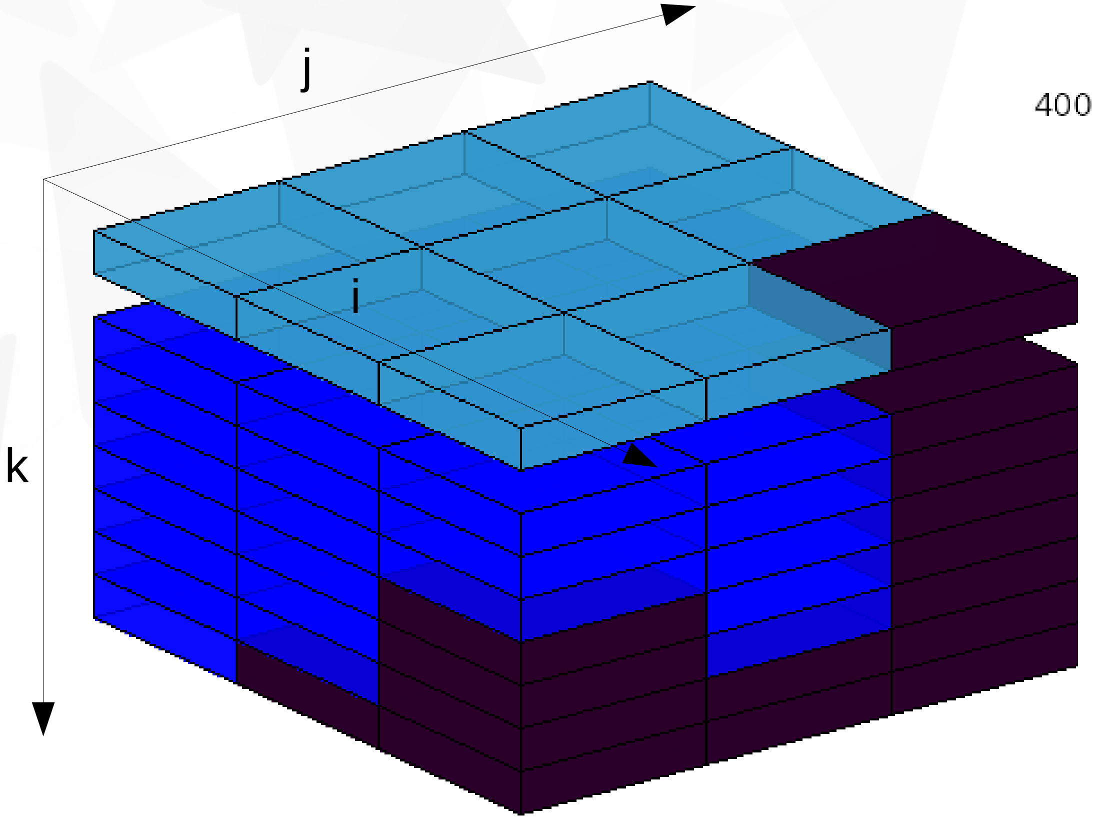
Refactoring for Xeon Phi

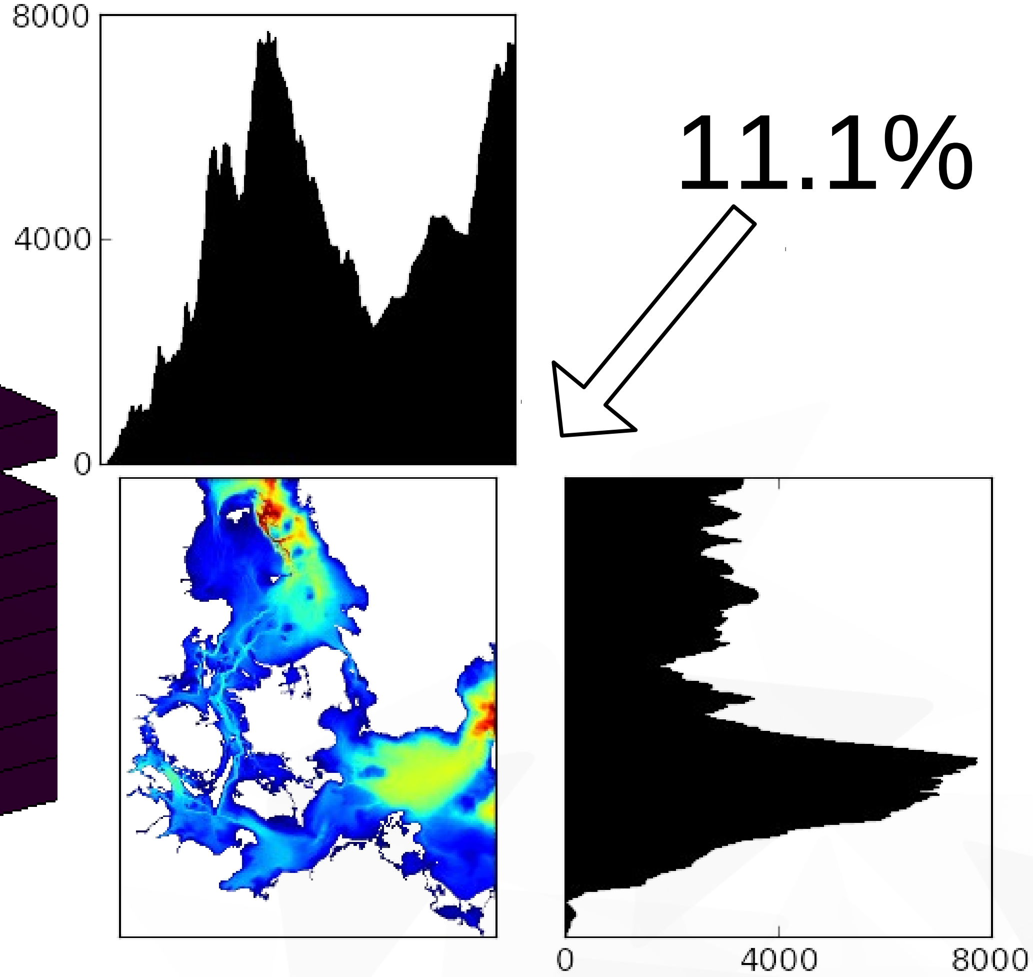
Jacob Weismann Poulsen, DMI, Denmark
Per Berg, DMI, Denmark
Karthik Raman, Intel, USA

HBM (3D ocean model) experiment

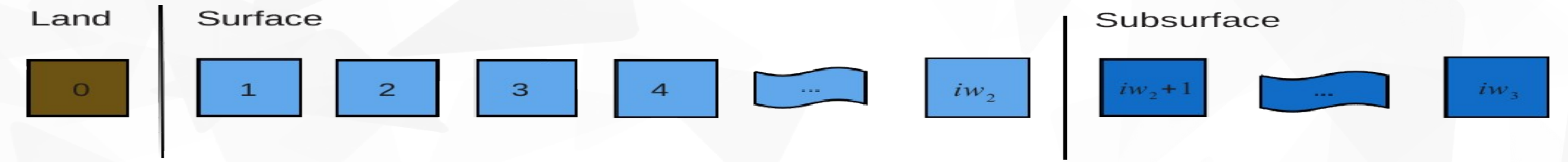
- 2S IVB (Q3, 2013) vs 1 KNC card (Q2, 2013)
  focus on native mode
- The most expensive and the most involved part
  - Tracer advection based on TVD scheme
  - Accounting for ≥ 44% of total runtime
  - 19 subroutines, SLOC: 5.5K/76K
- Establish kernels for every individual subroutine and analyze them (roofline approach). Moreover, analyze across all the subroutines to minimize dependencies preventing vectorization and halo swaps requiring thread barriers.
- Tools: Pen & paper, compiler reports, assembly code inspection and a profiler.

The data is sparse (highly irregular)





11.1%

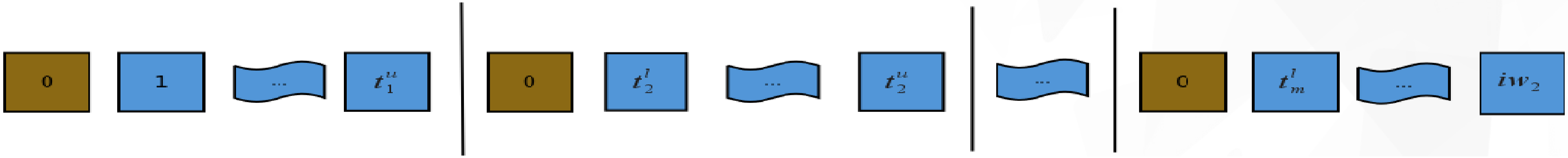


Insights – expose all parallelism (threads)

- SPMD instead of the usual loop based approach
  - a single openMP block with orphaned barriers around haloswaps will do (no explicit scoping).

```
!$OMP PARALLEL DEFAULT(SHARED)
call foo( ... );call bar(...); ...; ...
!$OMP BARRIER
call halo_update(...)
!$OMP BARRIER
call baz( ... );call quux(...); ...
!$OMP END PARALLEL
...
subroutine foo(...)
  ...
  call domp_get_domain(kh, 1, iw2, nl, nu, idx) ! Load balance
  do iw=nl,nu
    ! all threadlocal wet-points (:,:,:) are reached here
    ...
  enddo
end subroutine foo
```

  - Each thread will handle a subinterval of columns



Insights – expose all parallelism (SIMD)

- Loops are structured like this:

```
do iw=       ! horizontal - mpi/openmp parallelization
  do k=      ! vertical   - vectorization
    do ic=   ! innermost loop with number of tracers
```

- Step1: Get all outer k-loops to vectorize
- Step2: Ensure efficient SIMD code is generated

Premature abstraction is the root of all evil, e.g.

```
do k=2,kmax
  k1 = k+off1 ; k2 = k+off2
  t(1:nc,k) = t(1:nc,k) + A(k)*(B(1:nc,k1)-B(1:nc,k2))
enddo
```

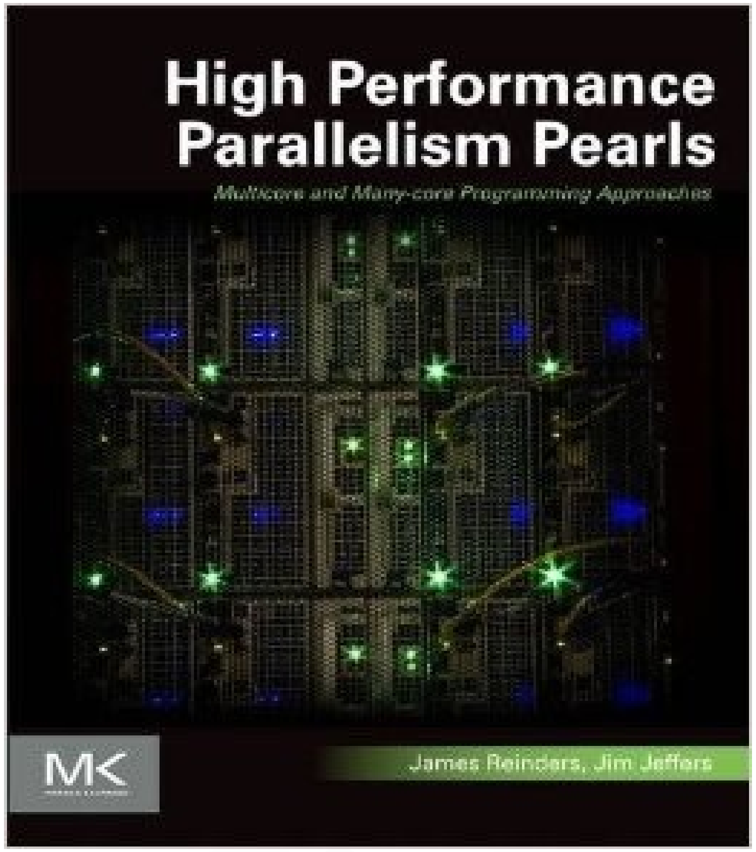
- Design choice for stencil codes: columns one-by-one using work arrays (tripcount) or whole stencil in one go (intensity) plus required remainder loops

Performance (memory bandwidth bound)

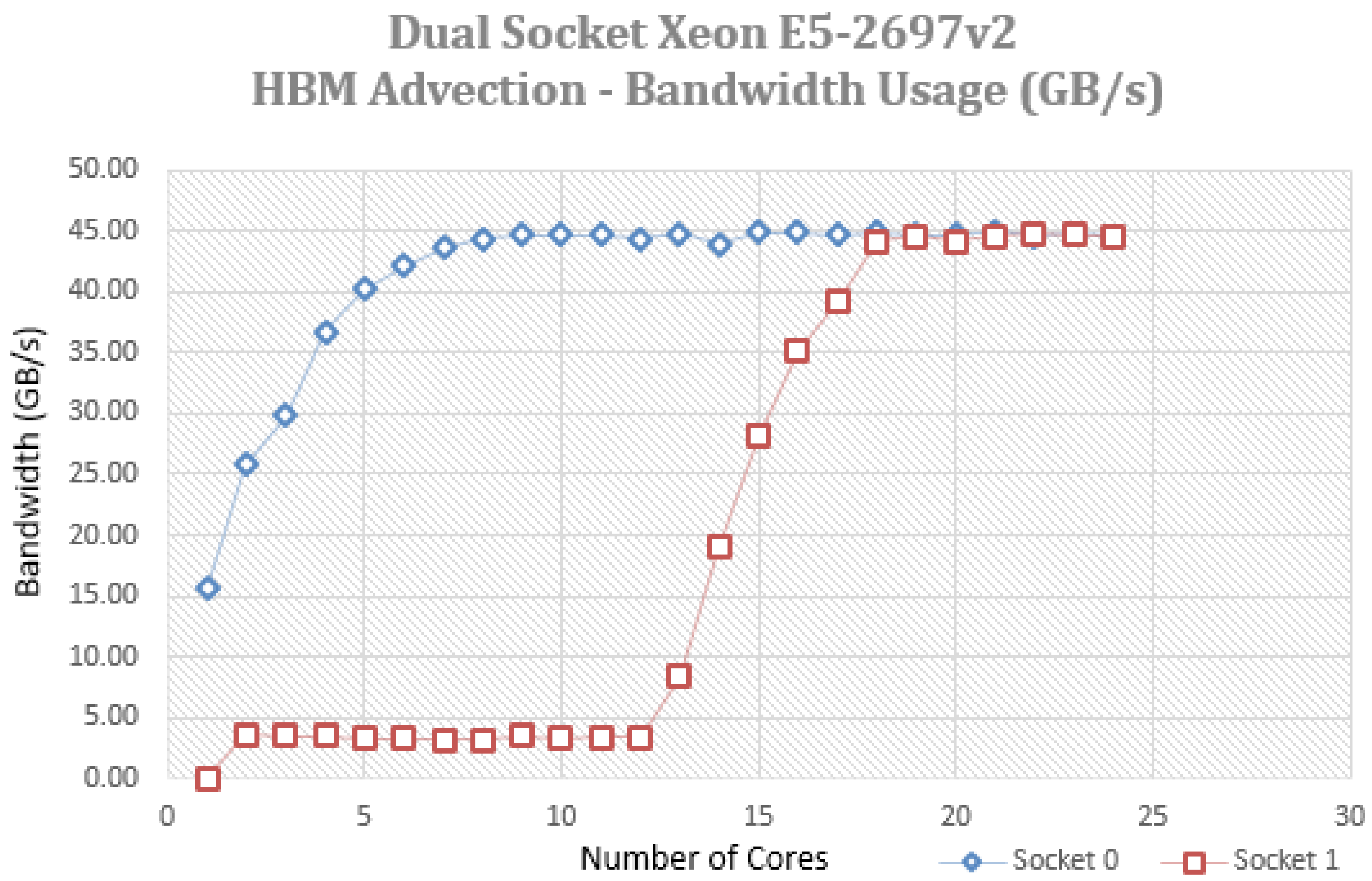
Advection (same algorithm)	2S-IVB 2697v2	KNC C0 7120A
Threads	48	240
Timing [sec]	81	72
Relative timing [%]	100	89
Stream Triad [GB/s]	86	177
Stream Triad [GB/s/watt]	0.21	0.63
BW sustained [GB/sec]	86	155
BW sustained [% peak]	100	88
BW sustained [GB/sec/watt]	0.21	0.55
Vector intensity sustained		7

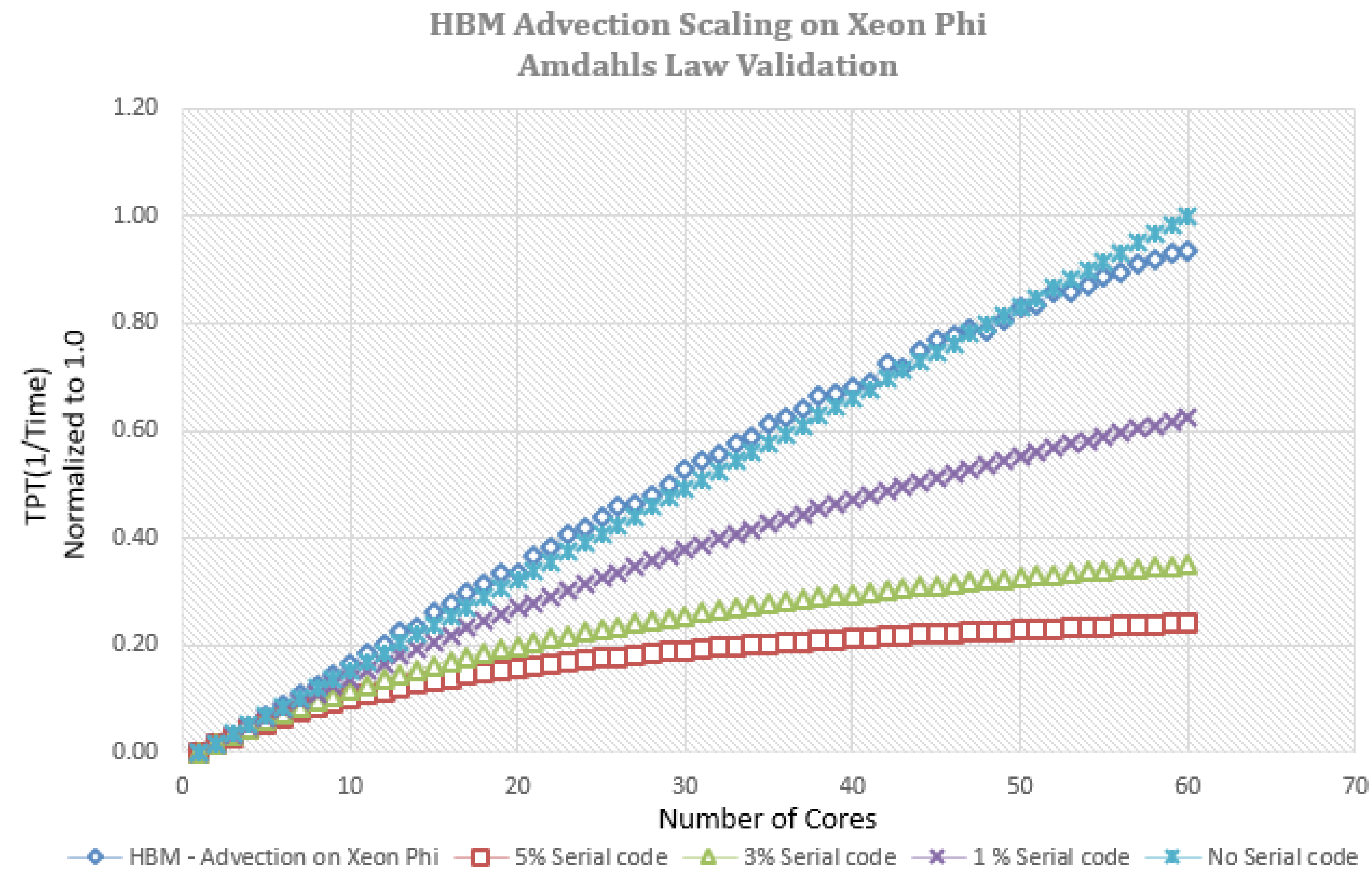
More information

- Experiment is documented in chapter 3 in a recent book (Morgan Kaufmann; 1 edition November, 2014; ISBN: 978-0128021187) on parallelism pearls for Xeon Phi.
- Code and testcase is available online: http://lotsofcores.com


High Performance Parallelism Pearls
Multicore and Many-core Programming Approaches
MK
James Reinders, Jim Jeffers

- The preparation work is documented in a technical report: http://www.dmi.dk/fileadmin/user_upload/Rapporter/tr12-20.pdf


Dual Socket Xeon E5-2697v2
HBM Advection - Bandwidth Usage (GB/s)
Bandwidth (GB/s)
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
0
5
10
15
20
25
30
Number of Cores
Socket 0
Socket 1


HBM Advection Scaling on Xeon Phi
Amdahls Law Validation
TPT(1/Time)
Normalized to 1.0
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0
10
20
30
40
50
60
70
Number of Cores
HBM - Advection on Xeon Phi
5% Serial code
3% Serial code
1 % Serial code
No Serial code

Acknowledgement

- Michael Greenfield, Intel
- Larry Meadows, Intel
- John Levesque, Cray
- Bill Long, Cray