

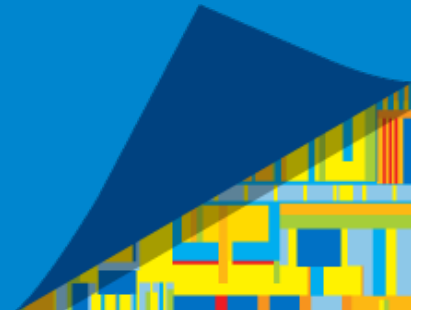


# Enabling Technology Trends in High Performance Computing

**Karl W. Schulz, Ph.D.**  
**Enterprise High Performance Computing Group, Intel**

MVAPICH User Group Meeting ♦ August 2015, Columbus, Ohio

Acknowledgements: Paul Besl, Chris Cantalupo, Boyd Davis, CJ Newburn, Ravi Murty, James Reinders, Bob Chesebrough



# Legal Disclaimer and Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2014, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

## Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

# Outline

- Exascale challenges and motivation
- Current generation many core architecture
- Potential software items of interest
- Future hardware directions
- Sidebar - Community Supported HPC Repository

# Motivation

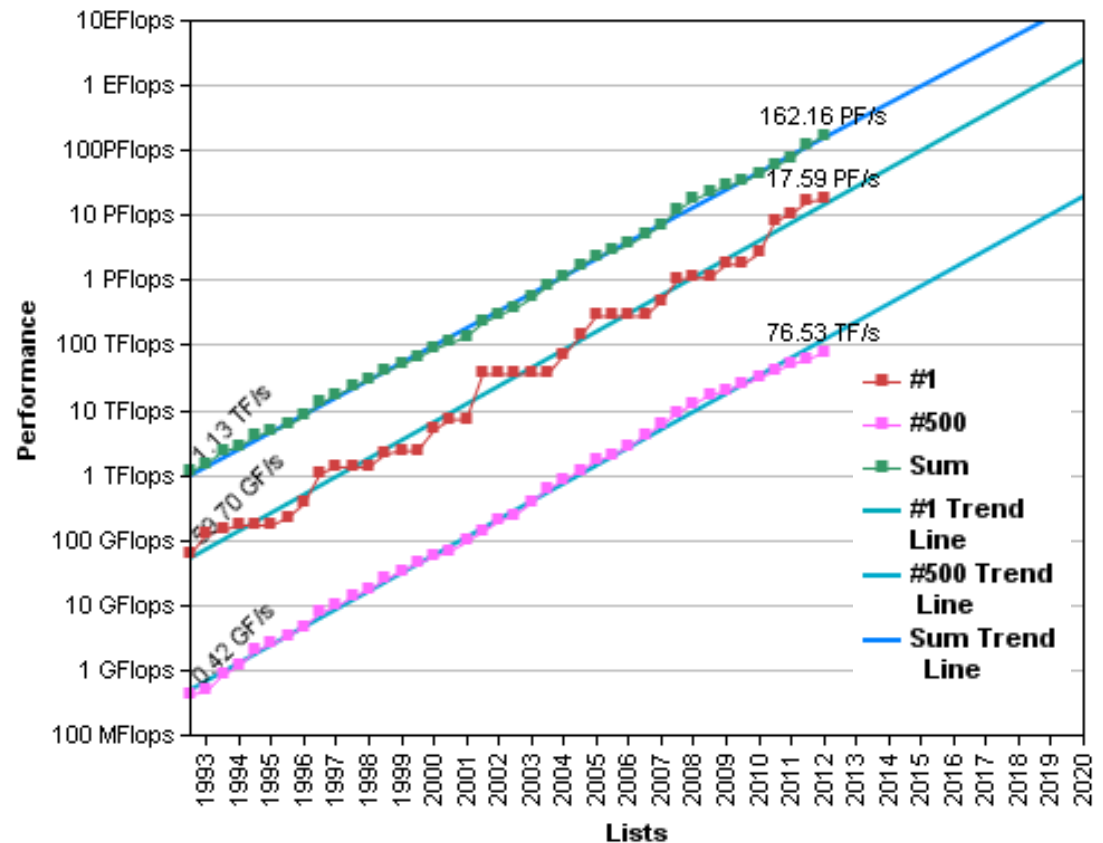
Compute power appetite for computational science is insatiable:

- we always need more:
  - higher fidelity simulations
  - parametric design
  - multi-physics coupling
- problem is further exacerbated with emerging disciplines like UQ:
  - sampling algorithms
  - high-order (intrusive) applications
  - ensemble methods

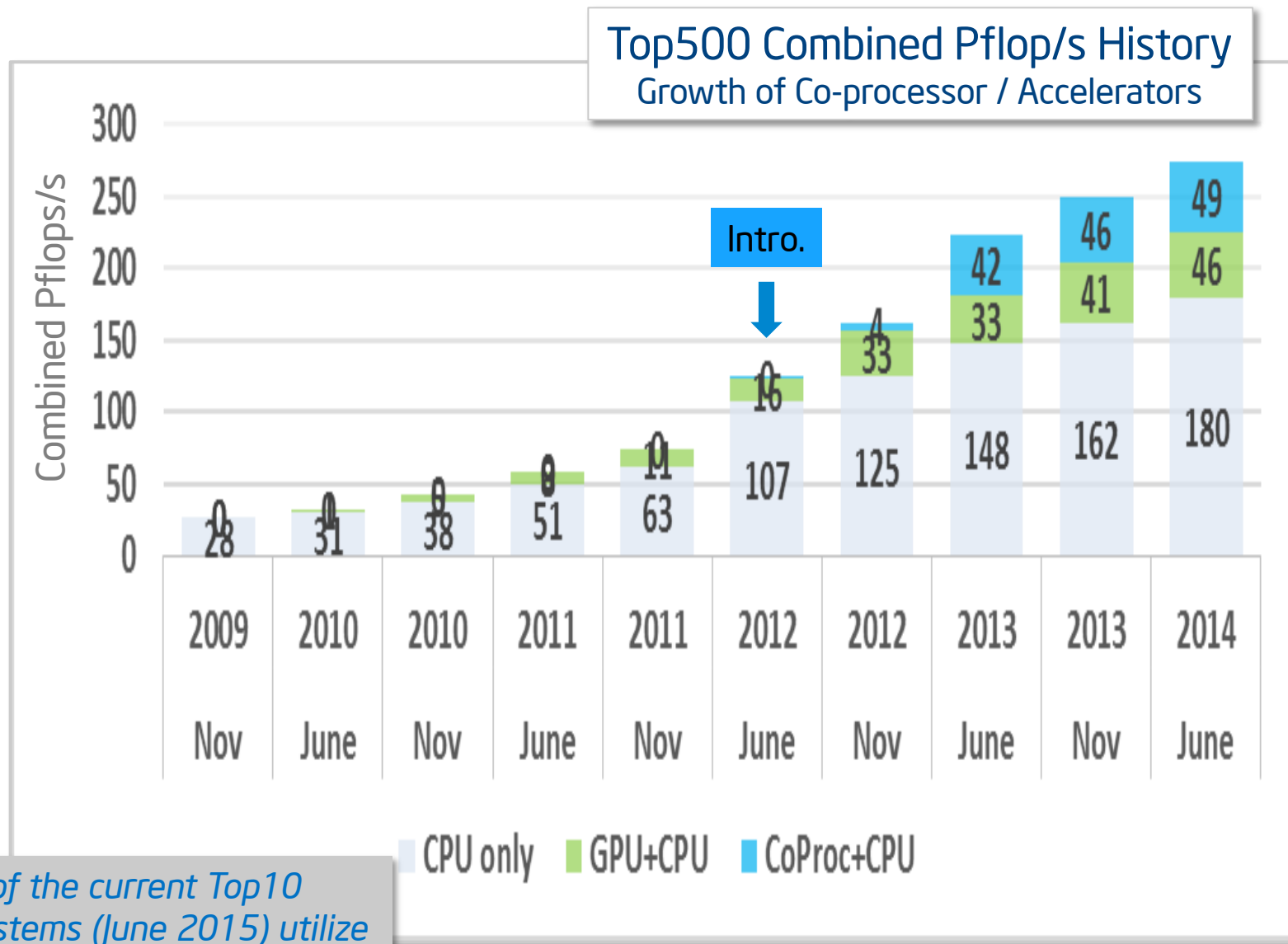
Top500 has been outpacing Moore's Law for the last 20 years helping to keep pace with computational demand

But, there is a limit with CMOS technology and the total power budget we can apply to future systems

Top500 History



# June 2014 Top 500 List



Source: [www.top500.org](http://www.top500.org)



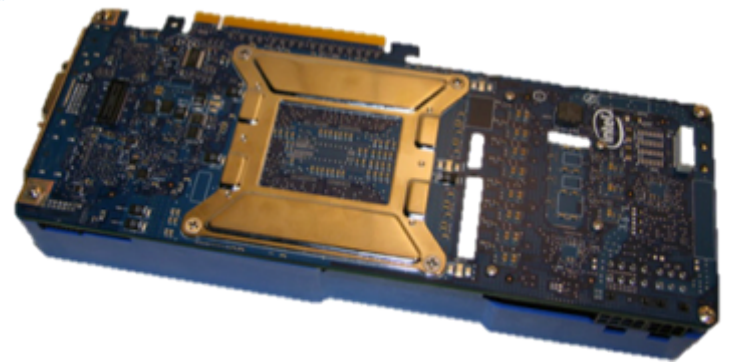
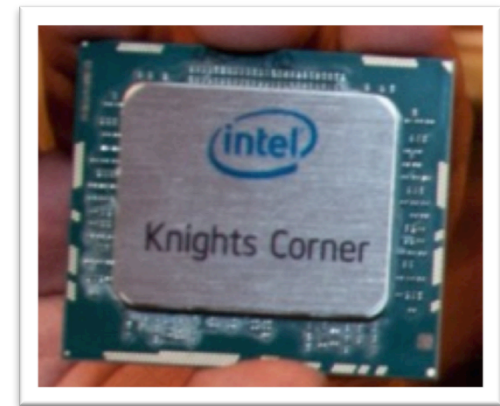
# Intel® Xeon Phi™ Architecture - Knights Corner

# Exascale Motivated Architectures

## *Current generation Xeon Phi (Knights Corner)*

### Basic Design Ideas:

- Leverage x86 architecture (a CPU with many cores)
- Use x86 cores that are simpler, but allow for more compute throughput
- Leverage existing x86 programming models
- Dedicate much of the silicon to floating point ops., keep some cache(s)
- Keep cache-coherency protocol
- Increase floating-point throughput per core (the power efficiency argument)
- Implement as a separate device
- Strip expensive features (out-of-order execution, branch prediction, etc.)
- Widened SIMD registers for more throughput (512 bit)
- Fast (GDDR5) memory on card



# Parallel Programming Models for Xeon Phi

## On-Node: Predominant parallel programming model(s):

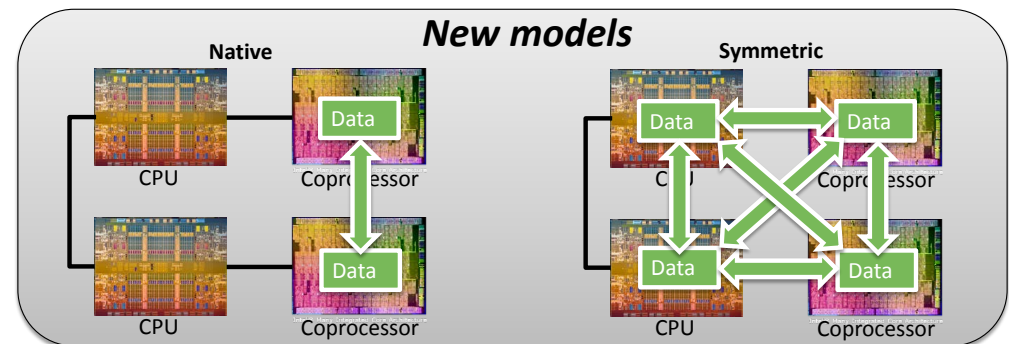
- Fortran: OpenMP, MKL
- C: OpenMP/Pthreads, MKL, Cilk
- C++: OpenMP/Pthreads, MKL, Cilk, TBB

## Off-Node:

- thru host (who then uses MPI)
- MPI directly

## Other conveniences

- MIC has familiar Linux-like environment
- you can login into it via ssh
- you can run “top”, debuggers, your native binary, etc
- can access your files through NFS, Lustre
- can run host-only, host+offload, native, or symmetric



MVAPICH2-MIC  
available to run  
native MPI code on  
Phi



# Example of Offload Execution

```
!dec$ offload target(mic:0) in(a, b, c) in(x) out(y)
!$omp parallel
!$omp single
    call system_clock(i1)
!$omp end single
!$omp do
    do j=1, n
        do i=1, n
            y(i,j) = a * (x(i-1,j-1) + x(i-1,j+1) + x(i+1,j-1) + x(i+1,j+1)) + &
                    b * (x(i-0,j-1) + x(i-0,j+1) + x(i-1,j-0) + x(i+1,j+0)) + &
                    c * x(i,j)
        enddo
        do k=1, 10000
            do i=1, n
                y(i,j) = a * (x(i-1,j-1) + x(i-1,j+1) + x(i+1,j-1) + x(i+1,j+1)) + &
                        b * (x(i-0,j-1) + x(i-0,j+1) + x(i-1,j-0) + x(i+1,j+0)) + &
                        c * x(i,j) + y(i,j)
            enddo
        enddo
    enddo
!$omp single
    call system_clock(i2)
!$omp end single
!$omp end parallel
```

*Kernel of a finite-difference  
stencil code (f90)*

# Things we are learning from current generation Xeon Phi...

- Performance tuning efforts on Xeon Phi repeatedly leading to improvements for host Xeon as well
- Wider vector units - we need to make sure we are taking advantage of them:
  - Un-vectorized loops lose 4x performance on Sandy Bridge and 8x performance on Phi
  - Quick sanity check for evaluating code readiness for running on Phi
    - compile your application with and without vectorization
    - if a significant performance difference is not observed, likely want to spend some quality time with the compiler (more on that later)
- Thread affinization even more important on Phi with 200+ threads to manage - *each application is a beautiful unique flower in this regard*
- OS scaling and SMP limits with Linux pushed with 60+ cores (eg. boot and jitter)

# Software: updates for an old friend

- We know OpenMP has been a popular defacto standard for shared-memory parallelism in computational science apps
- Not originally intended to support notion of “offloading” kernels
- Also did not target SIMD vectorization
- Latest OpenMP 4.0 standard endeavors to address some of these issues

## New items in OpenMP 4.0

- Support for accelerators (or heterogeneous devices)
- Thread affinity
- SIMD support for vectorization
- Thread cancellation
- Fortran 2003 support
- Extended support for:
  - Tasking
  - Reductions
  - Atomics

*From: E. Stotzer, “OpenMP 4.0 Acceleration”,  
IXPUG’14, Austin, TX*



# OpenMP 4.0 - Support for Offload

- Use target construct to:
  - transfer control from the host to the device (a la previous offload stanza)
  - establish a device data environment (if not done yet)
- Host thread waits until offloaded region is completed
- Newer Intel compiler toolchains have support for OpenMP 4.x semantics

```
#pragma omp target map(to:b[0:count]) map(to:c,d) map(from:a[0:count])  
{  
#pragma omp parallel for  
    for (i=0; i<count; i++) {  
        a[i] = b[i] * c + d;  
    }  
}
```

host  
target  
host

From: E. Stotzer, "OpenMP 4.0 Acceleration",  
IXPUG'14, Austin, TX

# OpenMP 4.0 - Support for SIMD directives

In this example, the programmer asserts:

```
#pragma omp simd reduction(+:sum)
for(i = 0; i < *p; i++) {
    A[i] = B[i] * C[i];
    sum = sum + A[i];
}
```

- $*p$  is loop invariant
- $A[]$  does not overlap with  $B[]$  or  $C[]$
- $sum$  not aliased with  $B[]$  or  $C[]$
- $sum$  should be treated as a reduction
- Allow compiler to reorder for better vectorization
- Vector code should be generated even if efficiency heuristic does not indicate a gain in performance

# Additional Enhancements for current generation Xeon Phi

- **hstreams is a library for concurrent streams in a heterogeneous context**
  - originally introduced with MPSS 3.4 as preview release
  - enables users to enqueue computation and communication in streams, enabling concurrency among the host and one or more coprocessors, or among tasks within a coprocessor
- **Performance improvements for hstreams coming in next MPSS release (3.6)**
  - includes ability to feed streams to cores on same host
  - offers improved load balancing using host/Phi
  - net effect, can get improved dgemm performance using host+Phi vs. current automatic offload (AO)

## Coming Hardware

- Next Generation Intel® Xeon Phi™: *Knights Landing*
- Next Generation Fabric: *Intel® Omni-Path*

# Overview of Knights Landing

## Platform Memory

Up to 384 GB DDR4 (6 ch)

## Compute

- Intel® Xeon® Processor Binary-Compatible
- **3+ TFLOPS<sup>1</sup>, 3X ST<sup>2</sup>** (single-thread) perf. vs KNC
- 2D Mesh Architecture
- Out-of-Order Cores

Over 60 Cores

## On-Package Memory

- Over 5X STREAM vs. DDR4<sup>3</sup>
- Up to **16 GB** at launch

Integrated Intel® Omni-Path

Processor Package

**Omni-Path**  
(optional)

- 1<sup>st</sup> Intel processor to integrate

**I/O**

Up to 36 PCIe 3.0 lanes



# Overview of Knights Landing (KNL)

- Microarchitecture based on 14 nanometer manufacturing process
  - Based on Intel® Silvermont with HPC enhancements
  - 4 Threads / Core
  - Deep out-of-order buffers
  - Gather/scatter in hardware
  - Advanced branch prediction
  - 2D Core mesh architecture
  - Binary compatible with Intel® Xeon® Processors
- Standalone bootable processor (running the host OS) and a PCIe coprocessor
- Integrated high performance on-package memory (MCDRAM library)
- Flexible memory modes for the on package memory including cache and flat
- Support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512)
- 70+ cores, 3+ TeraFLOPS of double-precision peak theoretical performance per single socket node
- No MPSS on KNL bootable processor - everything running natively
- Binary compatibility with Xeon (except TSX)

# Overview of Knights Landing (KNL) - On Package Memory

- AKA 'on-package memory', 'high-bandwidth memory', MCDRAM
  - Partnership with Micron Technology
- > 400 GB/s
- Up 16GB (at launch)
- NUMA support
- Over 5x energy efficiency, 3x density vs. GDDR5
- Multiple usage models, including 'L3 cache' and 'flat'
- Direct memory allocation can be coordinated with use of *memkind* library

*Note: Introduces a new degree of freedom exposed for applications and MPI*

# A Heterogeneous Memory Management Framework

- The memkind library

- Defines a plug-in architecture
- Each plug-in is called a “kind” of memory
- Built on top of jemalloc: the FreeBSD OS default heap manager
- Partition is defined by functions that provide inputs for operating system calls
- High level memory management functions can be over-ridden as well
- Available via github:  
<https://github.com/memkind>

*memkind library can be used today to simulate high-bandwidth memory*

- The hbwmalloc interface

- The high bandwidth memory interface
- Implemented on top of memkind interface
- Simplifies memkind plug-in (kind) selection
- Uses all kinds featuring on package memory on the Knights Landing architecture
- Provides support for 2MB and 1GB pages
- Select fallback behavior when on package memory does not exist or is exhausted
- Check for existence of on package memory

# MCDRAM as Cache

- **Upside**

- No software modifications required.
- Bandwidth benefit.

- **Downside**

- Latency hit to DDR.
- Limited sustained bandwidth.
- All memory is transferred DDR -> MCDRAM -> L2.
- Less addressable memory.

# Flat Mode

- **Upside**

- Maximum bandwidth and latency performance.
- Maximum addressable memory.
- Isolate MCDRAM for HPC application use only.

- **Downside**

- Software modifications required to use DDR and MCDRAM in the same application.
- Which data structures should go where?
- MCDRAM is a limited resource and tracking it adds complexity.

# Quick hbwmalloc Examples (C/Fortran)

## Allocate 1000 floats from DDR

```
#include <hbwmalloc.h>
float *fv;
fv = (float *)malloc(sizeof(float)*1000);
```

## Allocate 1000 floats from MCDRAM

```
#include <hbwmalloc.h>
float *fv;
fv = (float *)hbw_malloc(sizeof(float)*1000);
```

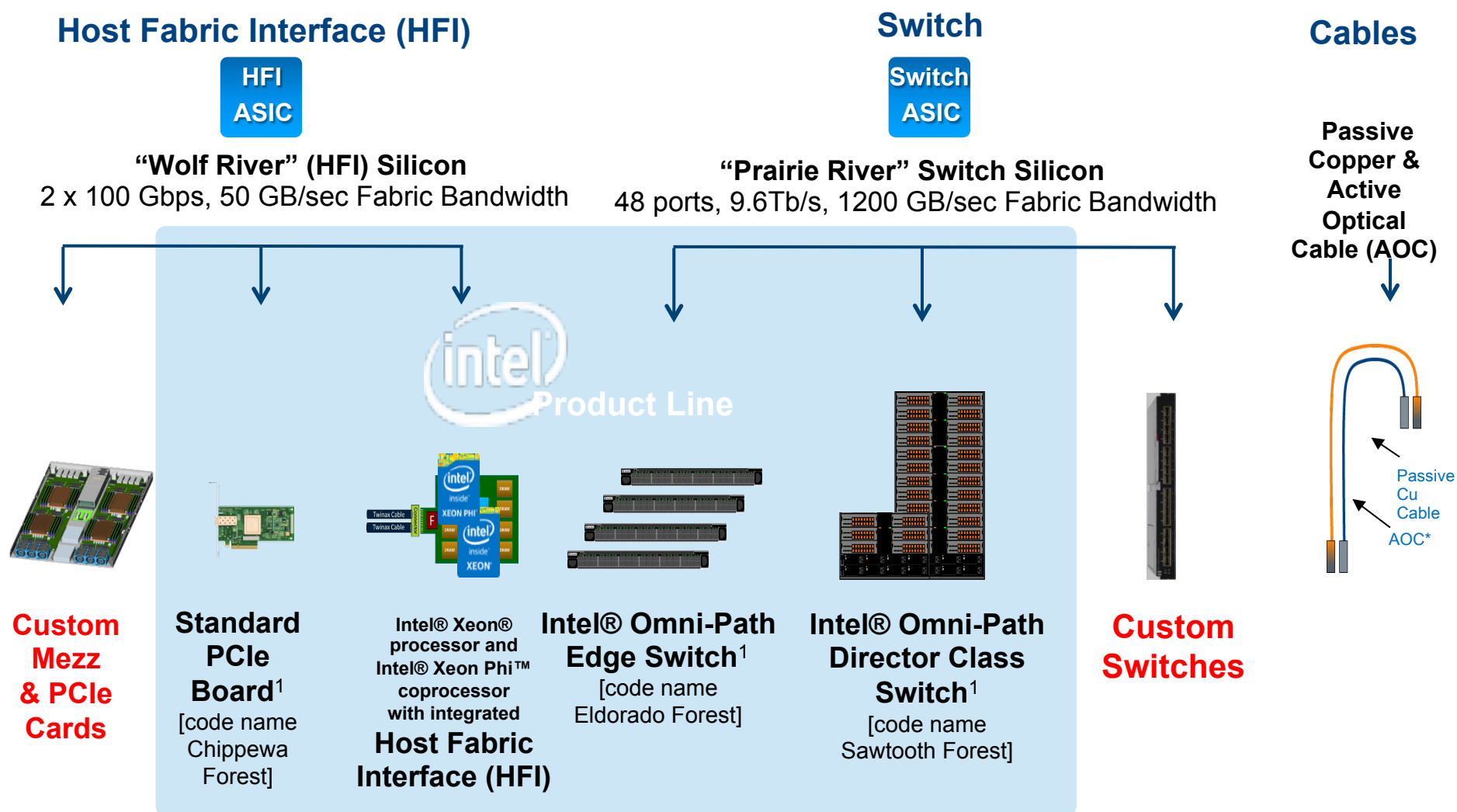
Similar example in  
Fortran using Intel  
Parallel Studio  
toolchain

```
c      Declare arrays to be dynamic
c      REAL, ALLOCATABLE :: A(:), B(:), C(:)

c      !DIR$ ATTRIBUTES FASTMEM :: A

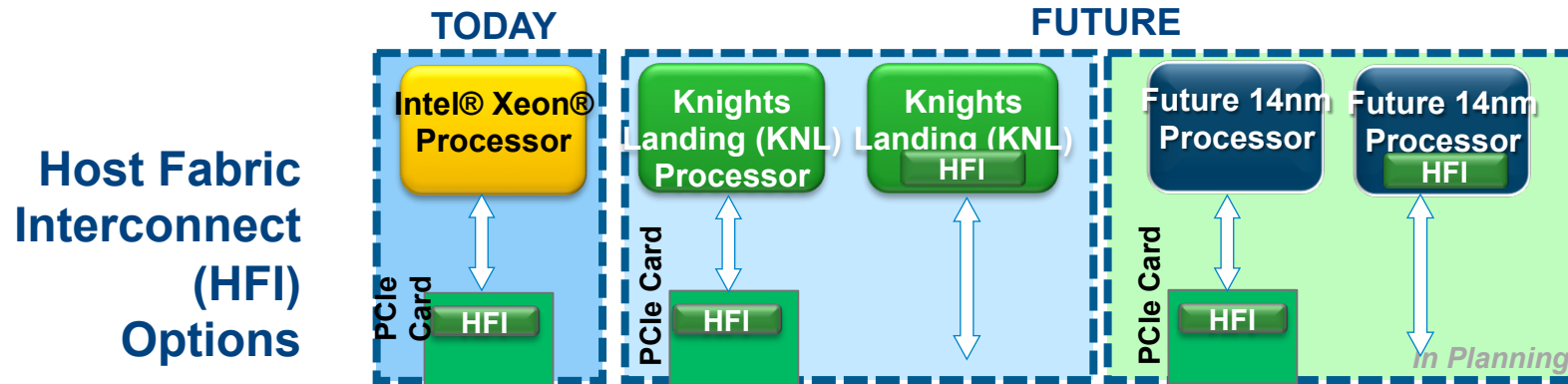
c      NSIZE=1024
c      allocate array 'A' from MCDRAM
c      ALLOCATE (A(1:NSIZE))
c
c      Allocate arrays that will come from DDR
c
c      ALLOCATE (B(NSIZE), C(NSIZE))
```

# Intel® Omni-Path Architecture Product Portfolio



<sup>1</sup> Will be available as both a reference design and Intel-branded product

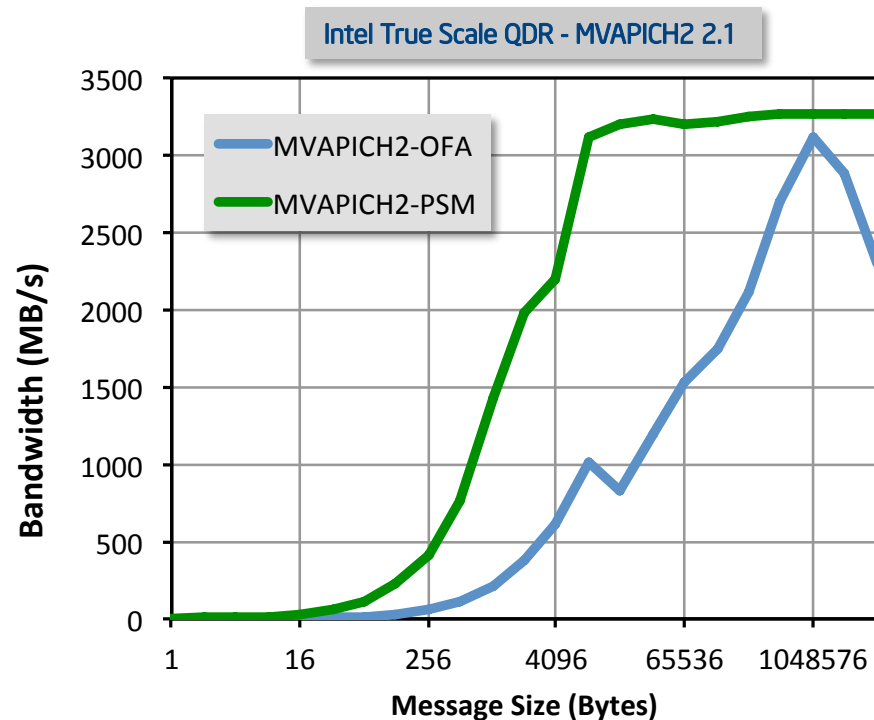
# Host Layer Optimization: Higher Messaging Rates, Low End-to-End Latency



- High MPI & PGAS messaging rate
  - 4th generation PSM tuned MPI Library
  - Improves HPC application performance & scaling
- Low end-to-end latency that stays low at scale
  - Connectionless implementation with hardware assist for Verbs
  - Improves application performance at scale
  - Excellent file system throughput

# Omni-Path + MVAPICH2

- Good news is that MV2 should work out of the box with PSM (Sayantan's talk has more details)
- I have personally run with MV2 over Omni-Path between
  - KNL<-> KNL
  - KNL <-> Haswell
- Quick reminder on effect of using PSM on today's True Scale hardware





# Announced Future Systems

*Supercomputers with Knights Landing*

Announced  
April '14



System name: **Cori**

**>9300** *Knights Landing nodes*  
*Open science system*

\*Source: NERSC.gov announcement April 29, 2014 . DOE--National Energy Research Scientific Computing Center  
Other brands and names are the property of their respective owners. Knights Landing: Next generation Intel® Xeon Phi™ processor and Intel® Xeon Phi™ coprocessor

HPC Wire  
July '14



System name: **Trinity**

*Mix of Haswell and Knights Landing*


\*Source: nnsa.energy.gov announcement July 10, 2014 and HPC Wire July 10, 2014

# Announced Future Systems

*Follow-on hardware generations*

Announced  
April '15



System Features	Mira	Aurora 
Compute Nodes	49,152	>50,000
Processor	PowerPC A2 1600 MHz	3rd Generation Intel Xeon Phi
System Memory	768 TB	>7 PB DRAM and persistent memory
System Interconnect	IBM 5D torus interconnect with VCSEL photonics	2nd Generation Intel Omni-Path Architecture with silicon photonics
File System Capacity	26 PB GPFS	>150 PB Lustre
Intel Architecture (x86-64) Compatibility	No	Yes
Peak Power Consumption	4.8 MW	13 MW

Source: <http://aurora.alcf.anl.gov>



Sidebar: Interest in Community Supported  
HPC Repository?

# Motivation for Community Effort

Many sites spend considerable effort aggregating a large suite of open-source projects on top of their chosen Linux distribution to provide a capable HPC environment for their users

- necessary to obtain HPC focused packages that are either absent or do not keep pace from Linux distro providers
- local packaging or customization frequently tries to give software versioning access to users (e.g. via modules or similar equivalent)

They frequently leverage a mix of external and in-house tools for:

- provisioning, software installations/upgrades, config management schemes, and system diagnostics mechanisms.
- although the functionality is similar, the implementations across sites is often different which can lead to duplication of effort

On the developer side, many successful projects must engage in continual triage and debugging regarding configuration and installation issues on HPC systems

*from ISC'15 BoF Session*



# Motivation (cont.)

Q: is there enough overlap in the software packages being assembled, the deployment use cases, and management strategies such that a community driven effort could help to provide a centralized element for HPC systems?

- Are there opportunities for sufficient reuse (say if minimal packaging guidelines and procedures can be established)?
- Would a community model be of interest?
- What is on the wish list for such a community effort?

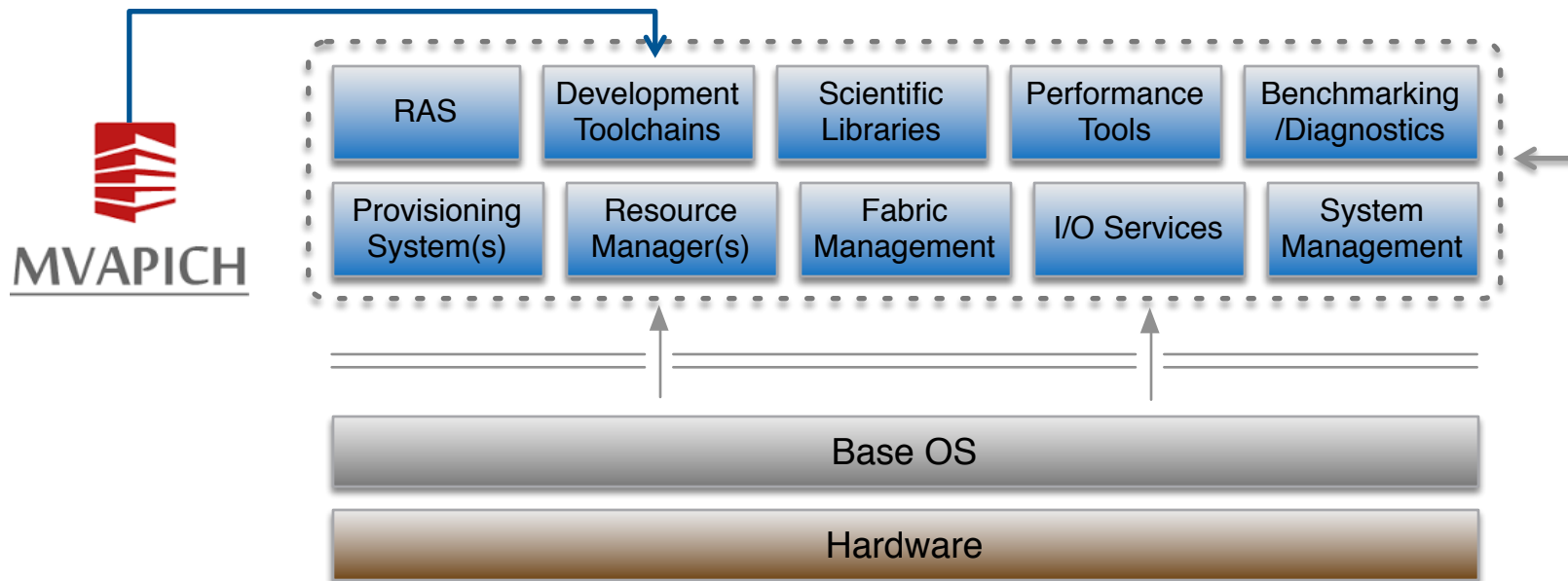
*from ISC'15 BoF Session*



# Community Mission and Vision

Mission: to provide an integrated collection of HPC-centric components that can be used to provide a full-featured reference HPC software stack

*Components should range across the entire HPC software ecosystem.*



from ISC'15 BoF Session

# Community Mission and Vision

## Vision

- to provide a collection of pre-packaged binary components that, when combined with a supported base operating system, can be used to install and manage HPC systems throughout their lifecycle
- to foster a nimble release cycle capable of supporting relevant new hardware configurations in a timely fashion
- to provide additional distribution/integration mechanisms for leading research groups releasing open-source software
- to allow and promote multiple system configuration recipes that leverage community reference designs
- to foster identification and development relevant interfaces between supported components that allows for simple component replacement and customization

*Related BoF submitted to SC'15 - please  
ping if interested*

# Summary

- Reviewed some of the trends and directions Intel is taking on path towards exascale
- Call for interest/community involvement in establishing HPC oriented repositories
- And speaking of community....

I would like to commend DK and his team:

- they consistently go well above and beyond the role of traditional academic software providers
- MVAPICH has evolved into production software that is supporting science in virtually all disciplines on systems around the world
- performance is critical and the team consistently delivers novel methods to improve performance on fast-changing hardware
- the HPC community benefits tremendously from this effort:

```
MPI_Send(&THANK_YOU,100000,MPI_INT,OSU,42,MPI_COMMUNITY);
```



# Thanks for your Time!

Questions? [karl.w.schulz@intel.com](mailto:karl.w.schulz@intel.com)

