Knights Landing Intel® Xeon Phi™ CPU: Path to Parallelism with General Purpose Programming

Avinash Sodani
Knights Landing Chief Architect
Senior Principal Engineer, Intel Corp.
Roadmap

Why parallelism? Why general purpose?

Knights Landing: Intel® Xeon Phi™ Processor Architecture

Performance, Applications, SW Tools and Support

Future Trends and Challenges
Computing demand continues to grow

- **HPC**
  - Solving bigger and more complex scientific problems to improve day to day lives

- **Cloud & online data and services**
  - Massive growth in online data and services, spurring growth in data centers

- **Machine Learning**
  - Promise of solving problems that are very hard to solve algorithmically

- **Genetics & Medical**
  - Cure for life threatening diseases. Deeper understanding to prevent diseases.

- **IoT**
  - Connected devices slated to grow over 20B by 2020 (Gartner). Drive backend datacenter needs

- **Data Analytics**
  - Growth across both traditional and emerging usages. Investment both at government and commercial levels
“Power-wall” slowed frequency increase over last decade
Core counts on exponential growth – much faster than single core performance
Exponential Growth in Data Parallelism

Massive flops per chip with vector and core count growth

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Exponential Growth in Data Parallelism

Massive flops per chip with vector and core count growth

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Exponential Growth in Data Parallelism

Massive flops per chip with vector and core count growth

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Parallelism is the way forward

**Trend** → Lots of thread- and data-level parallelism

Systems becoming highly parallel. More vectors, more cores per CPU, more CPUs per system

Single thread performance increasing at slower pace

Significant performance potential for applications that parallelize and vectorize
Plenty of solutions in play

Several parallel HW options. Vary with usage

- CPUs
- GPUs
- FPGA solutions
- Application specific accelerators

Different ways to program them

- MPI/OpenMP/TBB/etc.
- Language extensions with pragmas, etc.
- Different GPU programming models: CUDA, OpenCL, OpenACC, etc.
- Accelerator-specific API
- Research models that try to encompass both CPU and GPU programming
Software story is important

Software generally live for decades. Much longer than hardware

- Important to change software for parallelism in a manner that preserves investment
- They should continue to run and perform well on future hardware
- Choose programming models that lasts long

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Knights Landing: First Intel® Xeon Phi™ Processor

Enables extreme parallel performance with general purpose programming

First **self-boot** Intel® Xeon Phi™ processor that is **binary compatible** with main line IA. Boots standard OS.

**Significant improvement in scalar** and **vector** performance

Integration of **Memory on package**: innovative memory architecture for high bandwidth and high capacity

Integration of **Fabric on package**

Potential future options subject to change without notice. All timeframes, features, products and dates are preliminary forecasts and subject to change without further notification.

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Knights Landing Overview

**Chip:** up to 36 Tiles interconnected by 2D Mesh

**Tile:** 2 Cores + 2 VPU/core + 1 MB L2

**Memory:** MCDRAM: up to 16 GB on-package; High BW

**DDR4:** 6 channels @ 2400 up to 384GB

**IO:** 36 lanes PCIe Gen3. 4 lanes of DMI for chipset

**Node:** 1-Socket

**Fabric:** Intel® Omni-Path Fabric on-package (not illustrated)

**Vector Peak Perf:** 3+TF DP and 6+TF SP Flops

**Scalar Perf:** ~3x over Knights Corner

**Streams Triad (GB/s):** MCDRAM: 450+; DDR: ~90

---

**Note:** not all specifications shown apply to all Knights Landing SKUs

Source Intel: All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. KNL data are preliminary based on current expectations and are subject to change without notice. 1Binary Compatible with Intel Xeon processors using Haswell Instruction Set for 10nm TSM. 2Bandwidth numbers are based on STREAM-like memory access pattern when MCDRAM used as flat memory. Results have been estimated based on internal Intel analysis and are intended for informational purposes only. Any difference in system memory or software design or configuration may affect actual performance.
**KNL Tile:** 2 Cores, each with 2 VPU
1M L2 shared between two Cores

**Core:** New OoO Core. Balances power efficiency, parallel and single thread performance.

**2 VPU:** 2x AVX512 units. 32SP/16DP per unit. X87, SSE, AVX, AVX2 and EMU

**L2:** 1MB 16-way. 1 Line Read and ½ Line Write per cycle. Coherent across all Tiles

**CHA:** Caching/Home Agent. Distributed Tag Directory to keep L2s coherent. MESIF protocol. 2D-Mesh connections for Tile
Many Trailblazing Improvements in KNL. But why?

<table>
<thead>
<tr>
<th>Improvements</th>
<th>What/Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self Boot Processor</td>
<td>No PCIe bottleneck. Be same as general purpose CPU</td>
</tr>
<tr>
<td>Binary Compatibility with Xeon</td>
<td>Runs all legacy software. No recompilation.</td>
</tr>
<tr>
<td>New OoO Core</td>
<td>~3x higher ST performance over KNC</td>
</tr>
<tr>
<td>Improved Vector Density</td>
<td>3+ TFLOPS (DP) peak per chip</td>
</tr>
<tr>
<td>New AVX 512 ISA</td>
<td>New 512-bit Vector ISA with Masks</td>
</tr>
<tr>
<td>New memory technology: MCDRAM + DDR</td>
<td>Large High Bandwidth Memory → MCDRAM</td>
</tr>
<tr>
<td></td>
<td>Huge bulk memory → DDR</td>
</tr>
<tr>
<td>New on-die interconnect: Mesh</td>
<td>High BW connection between cores and memory</td>
</tr>
<tr>
<td>Integrated Fabric: Omni-Path</td>
<td>Better scalability to large systems. Lower Cost</td>
</tr>
</tbody>
</table>

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. For more information go to [http://www.intel.com/performance](http://www.intel.com/performance). Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Core & VPU

Balanced power efficiency, single thread performance and parallel performance

- 2-wide Out-of-order core
- 4 SMT threads
- 72 in-flight instructions.
- 6-wide execution
- 64 SP and 32 DP Flop/cycle

- Dual ported DL1 → to feed 2 VPU
- Two-level TLB. Large page support
- Gather/Scatter engine
- Unaligned load/store support

- Core resources shared or dynamically repartitioned between active threads

- General purpose IA core

Avinash Sodani CGO PPoPP HPCA Keynote 2016
KNL ISA

E5-2600 (SNB\(^1\))  E5-2600v3 (HSW\(^1\))

x87/MMX  x87/MMX
SSE*  SSE*
AVX  AVX
AVX2  AVX2
BMI  BMI
TSX  

KNL (Xeon Phi\(^2\))

x87/MMX
SSE*  SSE*
AVX  AVX
AVX2  AVX2
BMI  BMI

No TSX. Under separate CPUID bit

KNL implements all legacy instructions
- Legacy binary runs w/o recompilation
- KNC binary requires recompilation

KNL introduces AVX-512 Extensions
- 512-bit FP/Integer Vectors
- 32 registers, & 8 mask registers
- Gather/Scatter

Conflict Detection: Improves Vectorization
Prefetch: Gather and Scatter Prefetch
Exponential and Reciprocal Instructions

LEGACY

1. Previous Code name Intel® Xeon® processors
2. Xeon Phi = Intel® Xeon Phi™ processor
AVX-512 CD: Instructions for enhance vectorization

```
for(i=0; i<16; i++) { A[B[i]]++; }
```

```
index = vload &B[i]  // Load 16 B[i]
old_val = vgather A, index  // Grab A[B[i]]
new_val = vadd old_val, +1.0  // Compute new values
vscatter A, index, new_val  // Update A[B[i]]
```

```
index = vload &B[i]  // Load 16 B[i]
pending_elem = 0xFFFF // all still remaining
do {
    curr_elem = get_conflict_free_subset(index, pending_elem)
    old_val = vgather {curr_elem} A, index  // Grab A[B[i]]
    new_val = vadd old_val, +1.0  // Compute new values
    vscatter A {curr_elem}, index, new_val  // Update A[B[i]]
    pending_elem = pending_elem ^ curr_elem // remove done idx
} while (pending_elem)
```

Code is wrong if any values within B[i] are duplicated

AVX-512 Conflict Detection

- VPCONFLICT{D,Q} zmm1{k1}, zmm2/mem
- VPBROADCASTM{W2D,B2Q} zmm1, k2
- VPTESTNM{D,Q} k2{k1}, zmm2, zmm3/mem
- VPLZCNT{D,Q} zmm1 {k1}, zmm2/mem

Avinash Sodani CGO PPoPP HPCA Keynote 2016
KNL Memory Modes

Three Modes. Selected at boot

Cache Mode
- SW-Transparent, Mem-side cache
- Direct mapped. 64B lines.
- Tags part of line
- Covers whole DDR range

Flat Mode
- MCDRAM as regular memory
- SW-Managed
- Same address space

Hybrid Mode
- Part cache, Part memory
- 25% or 50% cache
- Benefits of both

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Flat MCDRAM: SW Architecture

MCDRAM exposed as a separate NUMA node

Memory allocated in DDR by default \(\rightarrow\) Keeps non-critical data out of MCDRAM.

Apps explicitly allocate critical data in MCDRAM. Using two methods:

- **“Fast Malloc”** functions in High BW library ([https://github.com/memkind/memkind](https://github.com/memkind/memkind))
  - Built on top to existing libnuma API
- **“FASTMEM”** Compiler Annotation for Intel Fortran

Flat MCDRAM with existing NUMA support in Legacy OS
Allocate into DDR

```
c/C++
float *fv;
fv = (float *)malloc(sizeof(float)*100);
```

Allocate into MCDRAM

```
c/C++
float *fv;
fv = (float *)hbw_malloc(sizeof(float) * 100);
```

Allocate into MCDRAM

```
Intel Fortran
DECLARE arrays to be dynamic
REAL, ALLOCATABLE :: A(:)
!
DEC$ ATTRIBUTES, FASTMEM :: A
NSIZE=1024
allocate array 'A' from MCDRAM
ALLOCATE (A(1:NSIZE))
```
KNL Mesh Interconnect

Mesh of Rings
- Every row and column is a (half) ring
- YX routing: Go in Y → Turn → Go in X
- Messages arbitrate at injection and on turn

Cache Coherent Interconnect
- MESIF protocol (F = Forward)
- Distributed directory to filter snoops

Three Cluster Modes
(1) All-to-All (2) Quadrant (3) Sub-NUMA Clustering
Cluster Mode: All-to-All

Address uniformly hashed across all distributed directories

No affinity between Tile, Directory and Memory

Most general mode. Lower performance than other modes.

Typical Read L2 miss
1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor
Cluster Mode: Quadrant

Chip divided into four virtual Quadrants

Address hashed to a Directory in the same quadrant as the Memory

Affinity between the Directory and Memory

Lower latency and higher BW than all-to-all. SW Transparent.

1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return
Cluster Mode: Sub-NUMA Clustering (SNC)

Each Quadrant (Cluster) exposed as a separate NUMA domain to OS.

Looks analogous to 4-Socket Xeon

Affinity between Tile, Directory and Memory

Local communication. Lowest latency of all modes.

SW needs to NUMA optimize to get benefit.

1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Avinash Sodani CGO PPoPP HPCA Keynote 2016
KNL w/ Intel® Omni-Path Fabric

Fabric integrated on package

First product with integrated fabric

Connected to KNL die via 2 x16 PCIe* ports
Output: 2 Omni-Path ports
  ▪ 25 GB/s/port (bi-dir)

Benefits
  ▪ Lower cost, latency and power
  ▪ Higher density and bandwidth
  ▪ Higher scalability

*On package connect with PCIe semantics, with MCP optimizations for physical layer

Avinash Sodani CGO PPoPP HPCA Keynote 2016
Pre-production KNL Performance and Performance/Watt

Significant performance improvement for compute and bandwidth sensitive workloads, while still providing good general purpose out-of-box throughput performance.

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. Source E5-2697v3: www.spec.org. KNL results measured on pre-production parts. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance. *Other names and brands may be claimed as the property of others.
MCDRAM Cache Hit Rate

MCDRAM performs well as cache for many workloads. Enables good out-of-box performance without memory tuning.
Deep Learning Training on KNL

Significant boost in deep learning training performance with KNL
Setting a trend for future increase with same programming model

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 purchases, including the performance of that product when combined with other products. KNL results measured on pre-production parts. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance
*Other names and brands may be claimed as the property of others
Programming for KNL

No different than programming a CPU

Same basics apply

- Exploit thread parallelism – Use all cores
  - Using parallel runtimes like MPI, OpenMP, TBB, etc.
  - Not always necessary to use all threads/core to get best performance
- Exploit the data parallelism – Vectorize!
- Utilize high bandwidth memory

Similar optimizations help both Intel® Xeon® and Xeon Phi™ processors
Tools support evolving rapidly

<table>
<thead>
<tr>
<th>Tools</th>
<th>Features</th>
</tr>
</thead>
</table>
| **Auto-Vectorize** | • New instructions that help vectorize loops, e.g., Vconflict  
• Aggressive vectorization and multi-versioning  
• Masking and predication |
| **Language constructs to express parallelism** | • OpenMP pragmas  
• Task level parallelism  
• Higher level language constructs/libraries |
| **Compiler hints to guide Optimizations** | • Compiler pragmas as hints for vectorization  
• Aliasing/alignment directives |
| **Feedback on code changes for parallelization** | • Meaningful and actionable compiler feedback about optimizations  
• Profiling tools to better understand the program behavior  
• Drive compiler optimization through runtime metrics |
Future Trends

Transistor density will increase
→ more cores and flops
→ more integration of system components

Power will continue to be a big challenge
- Intense focus on power efficient designs
- System power efficiency via integration
- More intelligent power management to better share power among components
- Usage-specific instructions and functionality for power efficiency

More parallel solutions in future
Some Future SW Challenges

- Better **load balancing** between different threads
  - More task based parallelization, instead of bulk synchronous model

- **Data locality** conscious coding
  - Utilize caches well. Good for both performance and power

- **Reducing memory capacity** per thread
  - This can limit utilizing all cores in a CPU due to capacity constraints

- **Algorithms** that minimize global communications

- Continue to **improve tools** that provide relevant and actionable feedback to programmer on parallelization opportunities
Summary

- More parallel machines in future
- Parallelizing applications critical for performance
- Choice of “how” to parallelize is important → Software has a long life time

Knights Landing Xeon Phi™ processor
Massively parallel CPU with general purpose programming

CPU + general purpose programming provides a stable base for parallel software