Efficient Parallel Programming on Xeon Phi for Exascale

Eric Petit,

Intel IPAG DCG,

Seminar at MDLS, Saclay, 29th November 2016

DCG
Data Center Group

Legal Disclaimers

Intel technologies features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

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.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at https://www-ssl.intel.com/content/www/us/en/high-performance-computing/path-to-aurora.html.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Intel, the Intel logo, Xeon, Intel Xeon Phi, Intel Optane and 3D XPoint are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

*Other names and brands may be claimed as the property of others.

Agenda

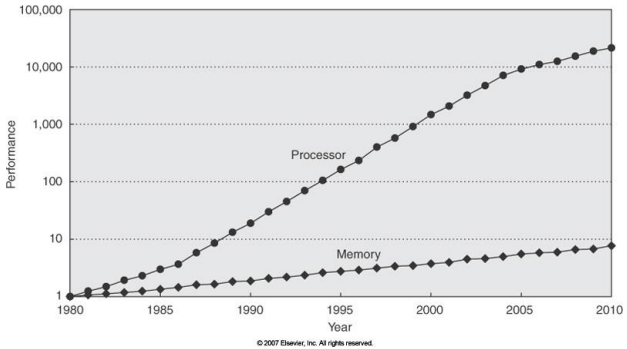
1. The path to exascale architectures and its challenges for the software stack
   1. Meet the power wall, freq wall, and memory wall
   2. Intel innovations for HPC systems
2. Some guidelines for code modernization
   1. The proto-application concept
   2. About the key Implementation strategies and optimization for efficient and scalable parallel programming
3. Conclusion

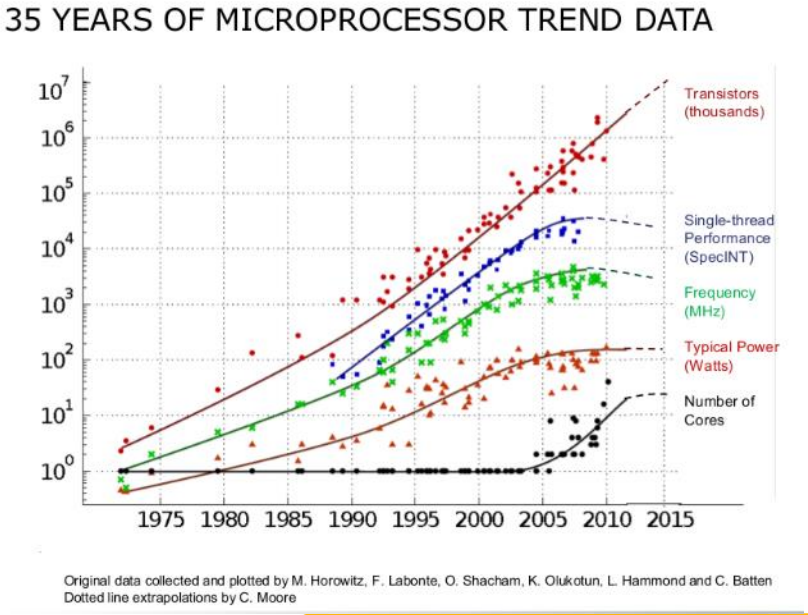
The path to exascale architectures and its challenges for the software stack

Meet the power wall, freq wall, and memory wall

After ~ 2005:
The number of transistors continues to increase, but:

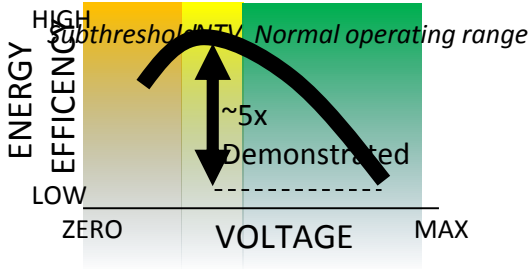
We have hit limits in
- Power (W and W/mm2)
- Frequency
- Memory gap is still here





Single core scalar performance is now only growing slowly

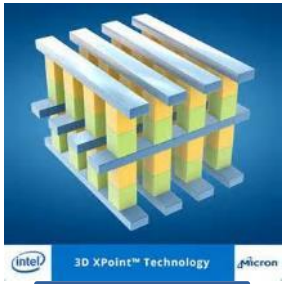
Some Promising (and existing) Intel Technologies



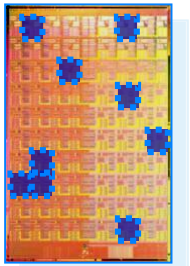
Near Threshold Voltage (NTV)

FPGA and ASICS

Omnipath integration

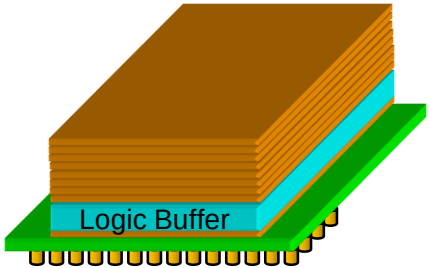


Fast storage



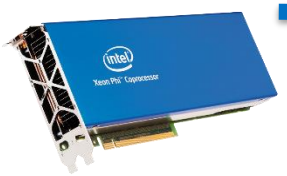
Fine-grain power management

Specialized circuits (SIMD, encryption, …)

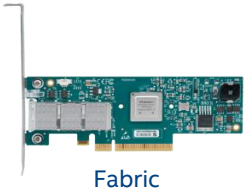


Stacked memory

Intel® Xeon Phi™ - A Paradigm Shift



Coprocessor



Fabric

Memory

Server Processor

Memory Bandwidth
450+ GB/s STREAM

Memory Capacity
Over 25x KNC

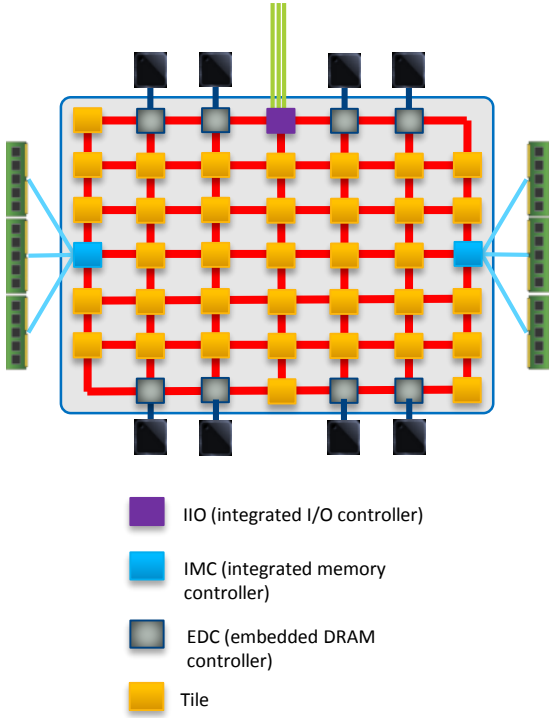
Power Efficiency
Over 25% better than card

I/O
200 Gb/s/dir

Density
3+ KNL with fabric in 1U

KNL processor

- Comprises a mesh connecting the tiles (in red) with the MCDRAM and DDR memories.
- Also with I/O controllers and other agents
- Distributed tag directory serves as connection point between tile and mesh
  - NUCA L2 cache
  - No L3 cache as in Xeon
- AVX512 instruction set



CPU-Fabric Integration

with the Intel® Omni-Path Architecture

KEY VALUE VECTORS

- ✓ Performance
- ✓ Density
- ✓ Cost
- ✓ Power
- ✓ Reliability

PERFORMANCE

Tighter Integration

Multi-chip Package Integration

Intel® OPA HFI Card

Future Generations
Additional integration, improvements, and features

Next generation Intel® Xeon® Phi™ coprocessor

Future Intel® Xeon® processor (14nm)

Intel® Xeon Phi™ processor

Next generation Intel® Xeon® processor

Intel® Xeon® processor E5-2600 v3

TIME

Building on the industry's best technologies

- Highly leverage existing Aries and Intel® True Scale fabric
- Adds innovative new features
- Re-use of existing OpenFabrics Alliance* software

Profond changes of exascale systems architectures

Increasing number of nodes, cores, accelerators

- Even if progress are made, some resources do not scale
  - Memory per core/thread, Bandwidth, Coherence protocol (NUCA caches…), Network interconnect, Fault tolerance
- Multiplication of hierarchical levels

  => Non uniformity
  - Different scales: bandwidth, memory size, performance
- Multiplication of the architectural designs

  => Heterogeneity
  - FPGAs, ASICs, Xeon, Xeon Phi => Capabilities, memory layout, concurrency…

The exascale programing challenges

Increasing complexity of parallel programing

Multiplication of the programing models

Weak AND strong scaling are required

⇒ **Evolutions are requested** for applications, runtimes and programing models to match the **architecture innovation**

⇒ **Co-Design and code modernization**

Some guidelines for code modernization

A pragmatic approach

- Start from real use cases and produce solutions that can be applied in production runs
- Target 100K's lines of codes application
  - Full rewriting is not an option
  - Compatibility with the existing software

  ⇒ Favor abstraction and library approach and piecewise refactoring through proto-applications

  ⇒ Mixing shared and distributed parallelism in a transparent way to the end-user

Why not experimenting in the original application?

Exascale technologies required deep code modification

Full application are complex and costly to execute at scale

- Difficulty to experiment ground breaking solutions
- Cost of the experiments (Time, PY, CPUs)
- Needs proof of concept demonstrating ROI to decide

Code and use-cases might not be easily shared with the community

Need a strong and daily support of the application developer

Portability and maintainability of the solution

- Over specialization
- Long term support?

The Proto-Apps Concept

Aka mini-app, proxy-app (NERSC trinity, Argonne CESAR…)

Objectives: Reproduce at scale the behavior of a set of HPC applications and support the development of optimizations that can be translated into the original applications

- Easier to execute, modify and re-implement

If you cannot make the application open-source, you can at least open-source the problems.

- Support community engagement
- Reproducible and comparable results
- Interface with application developers

Is that a benchmark?

- Yes and no…
- Benchmarks can be proto-application that are widely representative
- Proto-apps can becomes (good) benchmarks, with some software engineering
- Benchmarks and proto-apps are two different terminology translating a different finality of very similar objects

Building a proto-application

Build-up : ‘mini-app’ that mimic a full app with simpler physic

- All aspect are explored
- Minor involvement of the end-user
- No/Few IP issues
- No specific pb targeted
- Behavior at scale compare to the original?
- Representativeness? Use cases? Feed-back to the real code?

Strip down: ‘Proxy-app’ who extract a particular kernel from an application

- Target a specific issue
- Must be representative at scale
- Easy feed-back to the user
- Only a part of the application is addressed
- Problem coupling? use cases generation?
- Larger end-user involvement

=> IP (code and use case)

IMHO I prefer the second one, even building multiple proto-app from an application to expose the different problems

About the key methodologies for efficient and scalable parallel programming

Trendy optimizations these days:

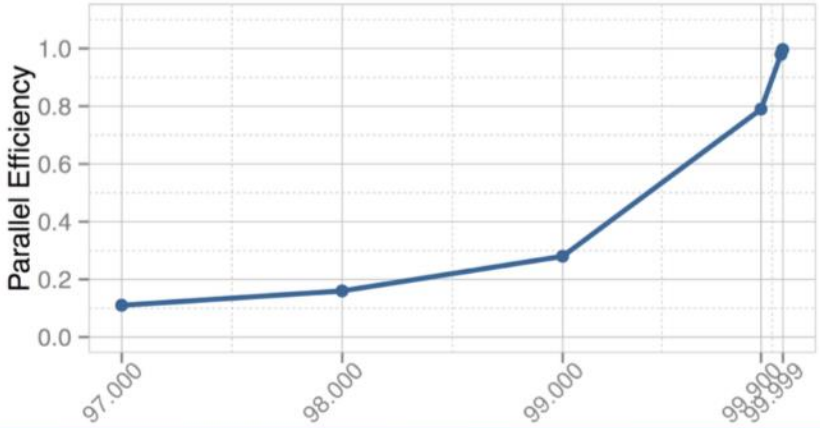
- MPI+X Hybrid parallelism
- Communication overlapping
- Vectorization

=> How to do safe choices? Will it pay off ? Up to what point?

A small paranthesis on MPI+X implementations

MPI+X is not trivial!

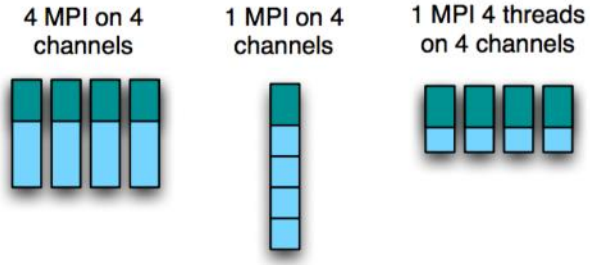
- Sequential section, runtime overhead on the critical path, synchronization Amdahl told you 99.999...% of your code must be parallel!!!
- With pure MPI usualy 100% of the code is parallel. The MP models constraint you to split or duplicate your work.
  - Loss in efficiency with work and memory duplication
  - Which is the reason why you might want to use shared memory model...



A small paranthesis on MPI+X implementations

(Real) multithreated communications are a requirement

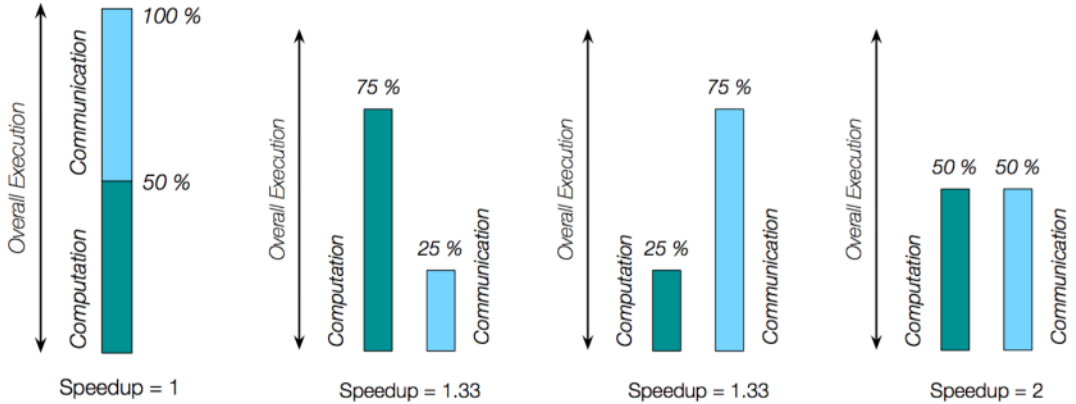
- All the operation of a communication must be parallelized (Amdhal...)
- There is some parallelism in the networks technologies, use it!
- Asynchronous communication allows comm/comp/comm overlap [2] [3]



[2]Note: The maximum speed-up of overlapping comm is 2 !

[3]Note on the note: the real gain in scalability must come from algorithmic changes in communication pattern: move away from bulk sync!

Limitation of Communication Overlap



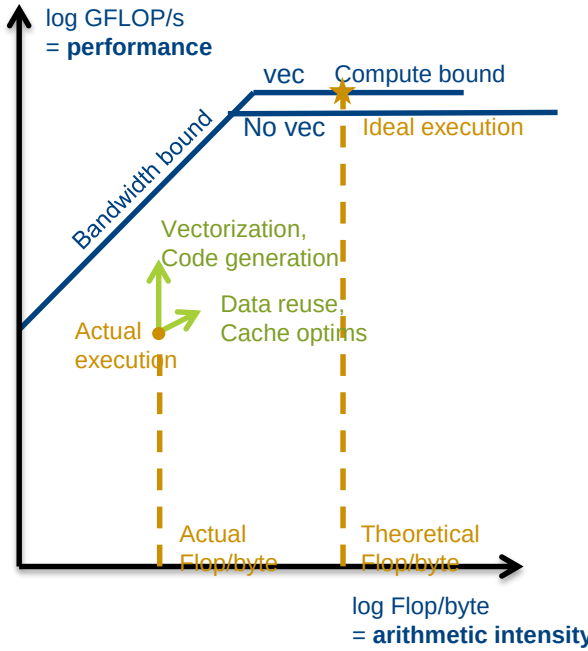
Maximal speedup: 2 $\times$

- Balanced computation and communication loads completely overlapped
- Overlap possibilities diminishing in strong scaling experiments
- Bounded performance gain
- Does not benefit to the scalability

$\Rightarrow$ Communication multithreading

Vectorization

- Vectorization is a core level feature!
- With bounded speed-up too, so it comes late in the priority list!
- Core hardware favor regular execution
  - Vectorization, pipeline, branch prediction, data prefetching…
- What about irregular applications?
  - Vectorization happens at core level
    - You need to find data parallelism in the dataset accessible to your core
    - Longer vector = higher need for data parallelism
    - Smaller memory per core = less available data parallelism
  - The roof line model (now in vtune) is a good indicator of the potential ROI.
    - Other tools like vector advisor, maqao etc.
    - Measure first, spend the effort after!



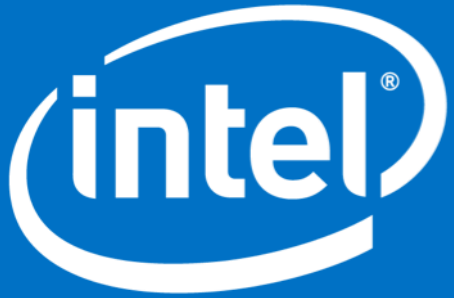
Conclusion

A new hope! But not for free…

- As usual software stack needs time to catch up, and HPC is highly competitive
  - Compare to many other domain, there is a need to have 35 years old code always running after the very the last piece of hardware
  - Require a strong comitment, and investment
  - => need to work closer with the specialists to make the turn
- Requirement for holistic approach to code modernization
  - Rethink top-to-bottom (discretization, algorithm, distributed parallelism, thread parallelism, vectorization)
  - But there are some permeability now… When working on the top layer, prepare (all) the next ones.
  - Proto-app as POC to explore and validate your choices
- Basic concepts prevail
  - Concurrency and locality (=> scalability and compute intensity)
  - Think and implement being architecture oblivious
  - Tune being architecture aware

  => If you do well today, it will pay-off for many years

=> New free lunch after frequency is scalability, do your best effort to subscribe!


intel