From Knights Corner to Knights Landing:

Prepare for the Next Generation of Intel® Xeon Phi™ Technology

Detect—and Correct—OpenMP* Inefficiencies

Optimized Simulations for HPC in Biomedical Research
Letter from the Editor
By James Reinders

Your Path to Knights Landing, the Next Generation of Intel® Xeon Phi™ technology
Prepare your application now for Knights Landing—the highly scalable, next-generation Intel® Xeon Phi™ processor/coprocessor that debuts this year.

OpenMP* Region Analysis with Intel® VTune™ Amplifier XE
Intel® VTune™ Amplifier XE can help OpenMP* users more easily understand where to invest their tuning efforts.

Walker Molecular Dynamics Laboratory Optimizes Biomedical Software
Walker Molecular Dynamics Laboratory turned to Intel® VTune™ Amplifier and other Intel® Software Development Products to optimize performance on both Intel® Xeon™ and Xeon Phi™ architecture.

Intel® Software Development Products Win HPCwire Awards
Two Intel® Software Development Products garner top honors from readers and editors alike.

Real-World Pearls of Wisdom in High Performance Parallelism
Leading experts from numerous industries and disciplines whip up delicious code in this “cookbook” of programming for better parallel performance.
LETTER FROM THE EDITOR


Are You Ready?

“It’s not technology that limits us. We’re the limitation. Our technology is an expression of our intelligence and creativity, so the limitations of our technology are a reflection of our own limitations.”

–Christian Cantrell, science fiction author, podcaster, and frequent social commentator on technology.

As software developers, we know this all too well. But it’s a new year and with it comes beginnings, hopes, and new avenues to overcome limitations.

Intel has been working for the last few years to create the next generation of Intel® Xeon Phi™ technology—codenamed “Knights Landing.” And this year, not only is Intel poised to ship it, developers are ready for all it promises. The first commercial systems to use this technology have already been announced and are expected to reach the market later this year.

“Knights Corner–Your Path to Knights Landing,” reveals what you can do now to prepare your application for the next-generation Intel Xeon Phi coprocessor. Programs that make good use of today's Intel Xeon Phi coprocessors (“Knights Corner”) will have a great path to the next generation and can be improved by a few techniques that prepare them for the future.

In our second feature, “OpenMP* Region Analysis with Intel® VTune™ Amplifier XE,” author Kirill Rogozhin shows us how Intel VTune Amplifier XE can help OpenMP developers more easily understand where to invest their tuning efforts.

In “Walker Molecular Dynamics Laboratory Optimizes Biomedical Software,” we also explore how scientific researchers were ready to take their experiments to the next level using the powerful Amber® molecular dynamics software. To enable scientists to run longer simulations and speed up research on Intel Xeon and Xeon Phi architecture, the lab turned to Intel VTune Amplifier and the latest features in Intel® MPI Library and Intel® Math Kernel Library (Intel® MKL).

As parallel programmers who push for the latest innovations and improvements, we’re always ready to hear and learn more from others. “Real-World Pearls of Wisdom in High Performance Parallelism” discusses my latest book co-edited with Jim Jeffers. The recently released book is a shared collection of the experiences of 69 developers from around the world.

It’s this community of shared experiences and information that continue to elevate what we can achieve in the Parallel Universe.

James Reinders
February 2015
In 2012, the world saw the first systems built on Intel's first many-core commercial product—the Intel® Xeon Phi™ coprocessor also known as “Knights Corner.” Today, Intel® Xeon® processors and these first-generation Intel Xeon Phi coprocessors contribute more FLOP/s (floating point operations per second) to the Top500 List than any other form of computation, including GPU accelerators. Tianhe-2, the world's fastest supercomputer on the Top500 List, uses Intel Xeon processors and Intel Xeon Phi coprocessors.

Made specifically for parallel processing, Knights Corner coprocessors have up to 61 x86 cores, each running up to 1.1 GHz. Each core has four hardware threads, which means that you can have as many as 244 threads of execution available per coprocessor for operation in parallel. That’s a lot.

While 1.1 GHz per core is lower than an Intel Xeon processor, a conscious trade-off has been made for very high parallelism versus single-threaded performance. A program designed for high degrees of parallelism will make the Knights Corner coprocessor shine—delivering up to 2.2 times the performance of processors and four times the power efficiency. Think two TeraFLOP/s at single-precision operation or one TeraFLOP/s double precision.
In short, the overall performance of the device can be quite a bit higher than with a processor, but it does demand a parallel program to get that performance.

Knights Landing: The Next Generation

Intel has been working for the last few years to create the next generation of Intel Xeon Phi technology—codenamed "Knights Landing." The first commercial systems to use this technology have already been announced and are expected to reach the market later this year.

The first was the Cori* Supercomputer at the National Energy Research Scientific Computing (NERSC) Center at the U.S. Department of Energy's Lawrence Berkeley National Laboratory. Another notable project is the Trinity* Supercomputer at the National Nuclear Security Administration (NNSA), a $174 million deal awarded to Cray that will feature Intel® Haswell processors and Knights Landing components.

“Many software developers have been inspired. They have been working hard on their applications, looking for that concurrency and examining how can they take advantage of new levels of computational power. The reward is a payoff in higher performance, whether the program runs on an Intel Xeon processor or an Intel Xeon Phi coprocessor.”

What’s New

Knights Landing brings significant enhancements to the Intel Xeon Phi family of coprocessors. For example:

> Knights Landing will have at least as many cores as a Knights Corner device, but it will be manufactured with Intel's 14-nanometer technology, which will add performance and power advantages. (The Knights Corner product line is made at a 22-nm process size.)

> You'll see a huge jump in performance, from one TeraFLOP/s for peak performance in the first generation to three TeraFLOP/s double-precision peak performance per single socket node.

> Knights Landing can perform as either a stand-alone, bootable processor (running the host OS) or a coprocessor (PCIe* end-point device). With the processor version, you won't be required to have a separate host processor as you do with Knights Corner. This versatility has a lot of implications. For one, all memory on a node or system can be dedicated to Knights Landing instead of being split between the many-core and multicore processing devices.
> With individual cores on the device supporting out-of-order execution, Knights Landing will improve performance and programmability—most notably, better capabilities for executing serial portions of programs.

> Large, high-bandwidth, on-package memory will boost applications that are very bandwidth-hungry and also have a substantial memory footprint. You can still use off-package memory; a tremendous boost for memory hungry applications will come with both on-device and off-device memory, just as with a regular processor.

> Versions will be available with integrated fabric, which will enable higher efficiency in cluster applications, such as those using the standard, portable MPI message-passing library.

> Knights Landing will offer five times the bandwidth (compared to DDR4) in one-third the space—with power efficiency more than 25 percent better than a discrete coprocessor and five times better than previous processors.

Knights Landing will standardize vector instructions around the Intel® AVX-512 specification. That's exciting, because as Intel Xeon Phi products become available in a processor format, it is valuable to have the instruction set completely synchronized with processors. (This is probably the most significant instruction difference between Knights Corner and Knights Landing. For testing purposes, a software emulator is available with full support for AVX-512 as implemented in Knights Landing.)
Another advantage relates to work under way with other vendors, including the GNU Project*, producers of the GNU Compiler Collection* (GCC*). The current version of GCC for Knights Corner cannot make use of these wide 512-bit vector instructions. That is set to change by the time Knights Landing is released.

**What Doesn’t Change: A Key Merit of Knights Landing**

Intel’s vision is to span from multicore to many-core architecture with consistent programming models, languages, tools, and techniques. **Our belief is this:** You should be able to write one program (using familiar compilers, libraries, abstract parallel programming models, and standards) and get top performance on both a multicore CPU and on the Many Integrated Core (MIC) architecture—in other words, first- and second-generation Intel Xeon Phi products.

You may need to modify your source code to support highly parallel operation, but that investment can be done in a consistent fashion. You are not forced to write in a different method when using lots of cores versus using a few *(Figure 1)*.

In keeping with this vision, Intel Xeon Phi products support critical, top-of-mind standards such as OpenMP* and MPI, Fortran, C and C++, and Intel® Threaded Building Blocks *(Intel® TBB)*.
These standards have been around a long time, have support from many vendors, and work on many different platforms. Many applications have been written using these standards, which were created for general-purpose devices, specifically CPUs of many different varieties. Because Knights Landing is designed to be very versatile—capable of doing anything you would expect a CPU to do—all of these standards are supported very well.

Knights Corner made good on this promise. Knights Landing adds a number of new innovations and improves on it.

Programming for High Levels of Parallelism

Systems based on Knights Landing are coming in the second half of 2015, but you can do a lot now to get ready for them. The most important considerations come down to one very important concept: concurrency.

As with any highly parallel device, the performance of each thread individually is lower than a CPU, but with hundreds of threads of execution, Knights Landing can deliver much higher overall performance than a CPU. It puts a lot of parallelism at your disposal, but you have to use it.

As Figure 2 shows, if your program uses only four or eight threads of execution, it will actually perform better on an Intel Xeon processor. Not until you start to use dozens of threads will you see an Intel Xeon Phi coprocessor outperform an Intel Xeon processor. More likely, you would use hundreds of threads to see this performance.

When fewer threads are used, the Intel® Xeon® processor wins on performance. With more parallelism, the Intel® Xeon Phi™ ultimately outperforms the processor. Design for this reality.
Figure 3 shows an actual customer example of relative performance for a single-threaded application written in Fortran using MPI. The Intel Xeon processor outperforms the Intel Xeon Phi coprocessor—Knights Corner, in this case.

However, when the program is tuned for parallelism, it's a much different story. The Knights Corner coprocessor far outperforms the Intel Xeon processor. Note that the optimizations done for Intel Xeon Phi coprocessors also improved performance on Intel Xeon processors (Figure 4). It's a win-win. This was a fairly straightforward piece of work with a customer. It didn't take a great deal of effort to find where we could expose more concurrency in the application and take advantage of it.

These results beg the question, Why didn't we do this work years ago? Why didn't we optimize our applications to get this performance boost on processor-based systems? Why are so few supercomputer workloads tuned this way?
The short answer is that having 61 cores to work on is very motivating. Once you get excited, port your application, and take advantage of 61 cores, you'll find that virtually every parallel platform using processors will benefit.

Many software developers have been inspired. They have been working hard on their applications, looking for that concurrency and examining how can they take advantage of new levels of computational power. The reward is a payoff in higher performance, whether the program runs on an Intel Xeon processor or an Intel Xeon Phi coprocessor.

So perhaps the first step toward Knights Landing readiness is the inspiration—proving to yourself that you can get your application to scale on a highly parallel device, and that would be Knights Corner.

The good news is that attaining higher levels of parallelism is easy to do using current techniques. OpenMP or Intel TBB lend themselves very well for this. Or you could use a certain number of MPI ranks on the device, along with OpenMP or TBB, to reach a very high thread count on a device. The next thing you know, you’re seeing performance above anything you could expect out of a CPU. But you have to have that parallelism.
Is Your Application Ready for Knights Landing?

It could be, but it’s unlikely. If it hasn’t been running on a highly parallel device, you probably haven’t shaken out all the impediments to scaling. There are three scaling criteria you need to look for the most:

- Thread scaling, or being able to scale as you increase the number of cores
- Vectorization scaling to ensure profitable use of vectorization for data parallelism
- Fabric scaling for cluster applications

There may be some cache-related optimizations as well, but many methods to attain thread scaling and vectorization require or incorporate the concepts of good cache utilization. It is best to focus on these three elements.

The best way to know if your application is ready for this scaling is to do some plotting on a high-core-count system, such as a Knights Corner system or an Intel Xeon EX processor-based system. Establish a performance and configuration discipline and cluster baseline for your workload at scale:

- For thread scalability, plot performance against the number of ranks per node.
- For vector scalability, plot vectorization profitability. Turn vectorization on and off, or compare performance with minimal amounts of vectorization versus maximum vectorization flags.
- For fabric scalability, plot performance against the number of ranks per node. Does performance go up when you increase the number of ranks?

Performance plotting is the first basic but very effective step in understanding whether or not you have opportunities to tune for better performance. Then you can take it from there.

For example, using OpenMP or TBB, you can incrementally increase your thread scaling to use more and more of the whole device as a rank. What happens to performance as you add more worker threads? On Knights Corner, about three threads per core is likely to be optimal, but it depends on the application. All require at least two threads per core to reach peak performance.
For vectorization scaling, most developers use intrinsic functions, autovectorization or the SIMD (single instruction, multiple data) directives included in the OpenMP 4.0 standard. Intel® compilers also include additional innovative reporting capabilities that help with vectorization.

The key point to remember is that once you understand these three important characteristics of your application, there's a lot you can do to improve its performance on all machines—such as those using Intel Xeon processors and Intel Xeon Phi coprocessors.

What Are the Best Systems to Prepare on?

Without a doubt, Knights Corner is the best system to use today to prepare for Knights Landing. If your application is a good fit for Knights Corner, you can tune the scaling dimensions to get the desired performance attributes, reap the near-term benefits, then prepare to be impressed with application performance on Knights Landing.

For example, if you're presently using optimizations that have to shuttle data back and forth to the CPU because Knights Corner doesn't have enough memory, you'll see a significant performance boost with the added memory and reduced data movement when running on Knights Landing.

“Knights Landing offers high-level source compatibility with the intrinsic functions of Knights Corner, while adding a number of enhancements for better performance on vector instructions.”

What If the Application Is Not a Fit for Knights Corner?

That's possible. The chief reason would be memory. While 16GB is a lot of memory for a card, it's not a lot of memory for a node. So there's a chance you may not get as much of your application running on Knights Corner as you'd like, due to limitations that apply to anyone designing to a card. Another limiting factor for applications can be high input/output (I/O) or communication requirements.
Furthermore, the relatively low serial performance of Knights Corner may call for material portions of your code to run on the host processor, which tends to be about 10 times faster. An unwanted side effect of such an approach is moving more data back and forth across the PCI bus.

If Knights Corner is not the right choice, then you need to understand the three scaling dimensions of your application on another system with a very large number of cores. Eight or 10 cores is nowhere near enough, because it doesn't require really tackling a very high degree of scaling—the effective use of threads and vectors.

Theoretically, you could have eight or 16 cores and use performance data from those few cores to predict performance for hundreds of cores. In reality, we've never seen this successfully done because it seems to fall short of inspiring enough attention to scaling.

You really have to put the application to the real-world test. If not using a 61-core Knights Corner system, it could be a very high-core count Intel Xeon system—an Intel Xeon processor EX system—if you have the budget for it. But if there's any way to use Knights Corner to prepare for Knights Landing, that's the way to go. Efforts to make your code ready on Knights Corner will be handsomely rewarded, not only on today's Intel Xeon processor-based machines but on Knights Landing in the future.

Looking to the Future

The enhancements in Knights Landing help with the considerations and limitations discussed earlier. For example:

- Additional versatility and memory (the ability to be a processor and to have processor-sized memory) enable many more applications and use models.
- The integrated fabric and onboard, high-bandwidth memory redefine the kinds of workloads Knights Landing can handle.
- Significantly higher serial code performance reduces the effects of moving back and forth between parallel and serial operations—and the work required to juggle those movements on large applications.

Knights Landing offers high-level source compatibility with the intrinsic functions of Knights Corner, while adding a number of enhancements for better performance on vector instructions. Enhancements to the Intel AVX-512 specification, such as better support for unaligned data, make it easier to program the vectorization capability of Knights Landing compared to Knights Corner. (If you've coded specifically for Knights Corner, there may be a little bit of work to do here.)

In short, if you have a Knights Corner application today, you're very well lined up for Knights Landing. And where your applications might be limited today by Knights Corner, Knights Landing will bring relief.
Inspired by 61 Cores

This theme recurs again and again. Having 61 cores to work with is very motivating. Find a Knights Corner machine or build out an Intel Xeon machine with a very high number of cores, be inspired by that, and you'll be well on the road to Knights Landing. For more inspiration, take a look at our new book, *High Performance Parallelism Pearls*, which has outstanding real-world examples, with actual code, contributed by 69 experts from around the world.

In summary:

> **Expose the concurrency.** To show workloads’ real value, Intel Xeon Phi requires them to have great concurrency across multiple dimensions. Getting ready for Knights Landing therefore means investing generically in exposing that concurrency in a way that allows our tools and hardware to take advantage of it, regardless of platform.

> **Find opportunities for improvement.** Using a high number of cores, plot the different dimensions of concurrency—thread scaling, vector scaling, and fabric scaling—to understand where you can tune your application to capitalize on high degrees of parallelism.

> **Use familiar and proven methods.** Intel is delivering on the promise of highly parallel computing using consistent programming models, languages, tools, and techniques. There is no need to adopt new methods or reinvent the wheel as you advance from a few cores to hundreds of cores.

The payoff for these investments is higher performance, whether the program runs on an Intel Xeon processor, Knights Corner, or the new Knights Landing, with lasting value for decades to come.
Resources

Knights Corner: Your Path to Knights Landing webinar by James Reinders

Intel® Xeon Phi™ Coprocessor High-Performance Programming, Jim Jeffers, James Reinders, 2013

High Performance Parallelism Pearls: Multicore and Many-Core Approaches, Jim Jeffers, James Reinders, 2014


Structured Parallel Programming, Michael McCool, Arch Robinson, and James Reinders, 2012

Online developer resources: software.intel.com/mic-developer

Try Intel® Compilers

Available in these software tools:

Intel® Parallel Studio XE Composer, Professional, and Cluster Editions
OpenMP* is a commonly used parallel programming model, especially in high performance computing (HPC). OpenMP constructs are intended to boost performance by distributing work among multiple threads. However, adding OpenMP directives into the code doesn't always result in the desired performance improvements—a programmer often needs to tune it. The tuning method is closely tied with performance analysis—you need to know the reason behind the inefficiency to fix the bottleneck.

The majority of well-known performance tools show performance-related information associated with functions/loops in a bottom-up or top-down style, rather than showing particular OpenMP parallel regions. As a result, a programmer loses the parallel region/parallel loop context. Without that context, understanding inefficiencies like imbalance or overhead becomes difficult.
Intel® VTune™ Amplifier XE 2015 can produce profiling results in terms of OpenMP constructs that the programmer operates with. It can show the parallel and serial time of an application, the difference between measured and ideal execution of a parallel region, statistics broken down by parallel loops, and per-region CPU utilization histograms. Users can more easily understand where to invest their tuning efforts by knowing how much gain in application wall-clock time they can theoretically achieve by fixing each problem. Additionally, overhead and spin time classification helps users understand the reasons of the inefficiency (e.g., waiting on a barrier due to load imbalance or spinning on a lock due to synchronization).

This article will describe what types of OpenMP inefficiencies the Intel VTune Amplifier can detect and how users could treat and address them.

BLOG HIGHLIGHTS

Improving MPI Communication between the Intel® Xeon® Host and Intel® Xeon Phi™

BY LOC-NGUYEN »

MPI Symmetric Mode is widely used in systems equipped with Intel® Xeon Phi™ coprocessors. In a system where one or more coprocessors are installed on an Intel® Xeon® host, Transmission Control Protocol (TCP) is used for MPI messages sent between the host and coprocessors or between coprocessors on that same host. For some critical applications this MPI communication may not be fast enough.

In this blog, I show how we can improve the MPI Intra-Node communication (between the Intel Xeon host and Intel Xeon Phi Coprocessor) by installing the OFED stack in order to use the Direct Access Programming Library (DAPL) as a fabric instead. Even when the host does not have an InfiniBand* Host Channel Adapter (HCA) installed, the DAPL fabric can still be used to transfer MPI messages via scif0, to a virtual InfiniBand* interface.

On an Intel® Xeon® E5-2670 system running the Linux* kernel version 2.6.32-279 and equipped with two Intel® Xeon Phi™ C0 stepping 7120 coprocessors (named mic0 and mic1), I installed MPSS* 3.3.2 and Intel® MPI Library 5.0 on the host. Included in the Intel MPI Library is the benchmark tool IMB-MPI1. For illustration purposes, I ran the Intel MPI Benchmark Sendrecv before and after installing the OFED stack obtained results for comparison. In this test used with two processes, each process sends a message and receives a message from the other process. The tool reports the bidirectional bandwidth.

More
Simple Configuration

Good news: You almost don't need any special configuration except the most recent versions of the Intel® Compiler and VTune Amplifier XE. To get all the features described in this article, use the following versions:

- Intel® VTune™ Amplifier XE 2015 Update 2
- Intel® Parallel Studio XE 2015 Composer Edition Update 2

To profile an OpenMP application, just run any typical analysis, such as Advanced Hotspots. The only additional setting you need is to set the following environment variable before running performance collection. For example (for Linux*):

```
export KMP_FORKJOIN_FRAMES_MODE=3
```

The reason is that some instrumentation of Intel® OpenMP runtime (related to per-barrier statistics) was still in the experimental state when this article was written. So you need to enable the functionality explicitly using the environment variable.

Please note that the experimental feature is a beta-quality feature that may or may not appear in a future production release. Data collected with the experimental feature enabled is not guaranteed to be backward-compatible with future releases.

Exploring Inefficiencies: Serial Code and CPU Utilization

We recommend you first look at the CPU utilization histogram on the Summary pane of your application results. It displays the elapsed time of your application, broken down by CPU utilization levels (Figure 1). It shows only useful utilization so the CPU cycles that were spent by the application burning CPU in spin loops (active wait) are not counted on the histogram.

![CPU utilization histogram](image1.png)
Ideally in parallel applications, most of the elapsed time should be concentrated in the “green” area, so the majority of the CPU cores are utilized. Figure 1 shows the result of our test running on an Intel® Xeon Phi™ coprocessor. It is clear that the majority of available computing resources are underutilized. There may be two main reasons:

> Big serial portion: Most of the code is running without parallelism at all.
> Poor efficiency of parallel regions: The code is parallel, but some bottlenecks limit scalability.

Now look at the “OpenMP Analysis” section on the same Summary pane (Figure 2).
This section contains the elapsed time of the application split into the elapsed time of the serial portion (outside of any parallel region) and the parallel part of the program. If the serial portion is significant, that would be a good place to start looking. Find ways to minimize serial execution by either introducing more parallelism or by doing algorithm or microarchitecture tuning for sections that seem unavoidably serial. For high thread-count machines, serial sections have a severe and negative impact on potential scaling (following the infamous Amdahl's Law) and should be minimized as much as possible. In our test, 93.4 percent of wall clock time is spent in a serial region, so this is definitely the bottleneck.

Serially executed code can be explored going to the Bottom-up tab, choosing the “/OpenMP Region/Thread/Function..” grouping and filtering by the Master Thread of the “[Serial - outside any region]” row (Figure 3). Here in the Bottom-up grid, we see that a serial region takes 21.571s, which is the majority of the elapsed time. But if we sort by CPU time, the first line is occupied by an OpenMP parallel region at line 179. It takes more than 164 seconds of CPU time compared to ~101 seconds in the serial region. Also, when we look at the serial region, it is essential to filter by Master Thread only. There are multiple OpenMP worker threads waiting or spinning on barriers when a serial region executes. We should exclude this time if we want to explore pure serial code. In our test, it takes only 20.713s of CPU time, which is several times less than the top-most parallel region. But on the highly parallel Intel Xeon Phi coprocessor, this kills performance; Amdahl's Law is strict here.

General efficiency of parallel regions can be checked with the “OpenMP Region CPU Usage Histogram” on the Summary pane. This is essentially the same as the usual CPU usage histogram, but the CPU usage is reported not globally but rather for the selected parallel region. Figure 4 shows the CPU usage histogram from the same Intel Xeon Phi coprocessor result as Figure 1, but only for the parallel region at line 153. Here we see a different picture—sometimes the CPU concurrency is close to the ideal of 224, but more time is still spent with 32-40 concurrent hardware threads.
The Parallel Universe

Exploring Performance Opportunities: Potential Gain

Let’s take a look at another test case. We have analyzed NAS Parallel Benchmarks (NPB) to explore possible problems in OpenMP parallel region execution.

Test setup:
- CPU: Intel® Xeon® processor E5-2697 v2 @ 2.70GHz, 24 cores/48 threads.
- OS: RHEL* 7.0 x64
- Compiler: Intel® Parallel Studio XE 2015 Composer Edition Update 2
- Workload: NPB 3.3.1, “CG - Conjugate Gradient, irregular memory access and communication” module, class B

The number of OpenMP threads is configured to 24 to match the physical core count. Figure 5 shows quite good CPU utilization for the benchmark run—the opposite of what we’ve seen in the previous example with a long serial region on an Intel Xeon Phi coprocessor. This is still not
ideal—significant time is distributed between 2 and 6 concurrently running cores. This means some parallel code is executed there, but it doesn't always load all 24 available cores. Serial time for NPB is negligible, so this is not a problem here (Figure 6). But look at the “Potential Gain” metric, highlighted in pink.

Potential gain estimates the difference in elapsed time between the actual measurement and an idealized execution of parallel regions, assuming perfectly balanced threads and zero overhead of the OpenMP runtime on work scheduling. So it is essentially the potential benefit of tuning, showing the maximum time that you might save by improving parallel execution. The potential gain metric can be more important than CPU or elapsed time, because it doesn’t focus you on the top time-consuming region; it focuses you on the region where you most likely get maximum results from tuning.

In our test, Potential Gain shows that optimization of all parallel regions to an ideal state would save us 3.975s, or 34.9 percent of total application runtime—a feasible opportunity for optimization.

Those were metrics of the whole application. Now, let’s go deeper, to the parallel region level. The five OpenMP top parallel regions are listed on the Summary tab, sorted by potential gain. In our test, the parallel region at line 514 is the source of almost all the application’s potential gain—3.958s (Figure 6). This is good for us—we have narrowed down the problem to a single parallel region.
Finding the Reason Behind Inefficiency

Once we’ve focused on a particular parallel region, click the link on the region name to navigate to a Bottom-up view grouped by OpenMP Region and the region of interest selected (Figure 7).

The Bottom-up grid view has different statistics about the parallel regions—elapsed time with potential gain, the number of OpenMP worker threads employed for the given region, and the number of instances of the region (how many times it was called; e.g., from an outer loop or calling function). CPU time is broken down into effective time (user code execution), spin time, and overhead time. We can see that spinning time is significant: 92.159s. Before investigating where it comes from, let’s take a quick look at the source code (Fortran)—see Figure 8. The parallel region at lines 514:695 contains multiple parallel loops designated by "!$omp do" constructs. This is bad news for us—metrics for the whole parallel region don’t tell which parallel loops are problematic.
Fortunately, VTune Amplifier XE can break down information not only by parallel region, but by OpenMP barrier as well. All 

```
#pragma omp for
```  

or 

```
!$omp do
```  

constructs have synchronization barriers, unless you include a “nowait” clause. Since VTune Amplifier can distinguish those barriers, we should be able to see CPU and wall clock time for each of the parallel loops inside the same parallel region. You might need to create a custom grouping such as “/OpenMP Region/ OpenMP Barrier Type/ OpenMP Barrier/..” to see the per-barrier data.

After grouping by OpenMP barriers, things become clearer. First, most of the potential gain, elapsed, and CPU time come from the parallel loop at line 572 that is highlighted in **Figure 9**.

![Per-barrier performance metrics](image)

Second, expanding the Spin Time column breaks it into categories, revealing that all our spinning is due to imbalance. Third, an “OpenMP Loop Schedule Type” column appears saying that the given loop is statically scheduled.

### Fixing Imbalance

Source code of the loop at line 572 is shown in **Figure 10**. There is no “Schedule” OpenMP clause so scheduling is static by default. The parallel loop suffers from imbalance, so it would be natural to try dynamic scheduling in the hopes that the OpenMP runtime will automatically redistribute the workload. Changed code is shown in **Figure 11**.

![OpenMP parallel loop at line 572, original version](image)

![Parallel loop at line 572 with dynamic scheduling](image)
Analyzing and Fixing Scheduling Overhead

After changing scheduling to dynamic, the parallel loop performance has become even worse. Elapsed time increased from the original 10.445s to 11.102s (Figure 12). However, details in the grid show a different picture now—the imbalance disappeared, so it was really fixed. But another column is highlighted in pink—74.99s of CPU time goes to scheduling overhead. It means that the OpenMP runtime library performs some internal processing too heavily.

Look at the column “OpenMP Loop Chunk.” It has changed from the original 3125 to 1. This means that each iteration is scheduled individually—work items for worker threads are too small, and the OpenMP library has to schedule them too often. This parallelism is too fine-grained.

Grain size 1 is the default for dynamic scheduling, the minimal value. To prove our theory about fine granularity, we changed the grain size to 20 (Figure 13).
Now see the performance results in Figure 14. Both imbalance and scheduling overhead take only about 1s. The barrier of the loop at line 572 has gone down in the hotspot list, because its potential gain has dropped to 0.077s from the original 3.133s. Elapsed time of the given loop is 8.928s versus the original 10.445s. Overall parallel region CPU time has decreased from the original ~250s to ~213s. So we have won some portion of the potential gain, though only about half of the potential 3 seconds.

![Performance metrics of the optimized loop](image)
A More Effective Way to Analyze Performance

Performance analysis of OpenMP applications is more natural with VTune Amplifier XE. You can investigate performance efficiency and bottlenecks in a “top-down” way: Start from general CPU utilization analysis and check what code is purely serial. You can see CPU utilization of the whole application or a particular parallel region. The potential gain metric helps you focus on the most interesting parallel regions in terms of potential benefit. Per-barrier data breakdown allows you to easily work with multiple parallel loops in the same parallel region.

The Bottom-up grid provides clear OpenMP statistics: scheduling type, chunk size, number of OpenMP threads, and parallel region instances. Performance metrics, such as detailed categorization of spin and overhead time, are intended to guide programmers in root-cause performance problems. This helps you understand what limits performance: imbalanced load, non-optimal granularity, waiting on locks, or something else.

These all are “synergy” features of the Intel OpenMP runtime library and VTune Amplifier XE. Both are available for free download as part of the Intel® Parallel Studio XE 2015 Professional Edition. You can download VTune Amplifier XE individually and additional online materials at the product site. Check out what performance issues the tools can find in your OpenMP application.

Try Intel® VTune™ Amplifier XE

Available in these software tools:
Intel® Parallel Studio XE 2015 Professional and Cluster Editions >
Intel® System Studio 2015 >
Intel® Media Server Studio Professional Edition >
Walker Molecular Dynamics Laboratory Optimizes Biomedical Software

Intel® Software Development Products increase application performance and productivity

Overview

The Walker Molecular Dynamics Laboratory at the San Diego Supercomputer Center (SDSC) focuses on advanced classical Molecular Mechanics (MM) and hybrid Quantum/Molecular Mechanical (QM/MM) simulation research. Up to 20,000 scientific researchers in academia and industry use Walker’s Amber® molecular dynamics software. It enables simulation of biomolecular systems for research, including drug discovery, biocatalysis, enzyme engineering, and advanced biomaterial development. The Walker Laboratory applies many of its research and development results to Amber software—providing users with the most advanced techniques for molecular dynamics simulations.

The broader research focus of the Walker Laboratory encompasses the fields of computational chemistry, molecular biology, and high performance computing (HPC). The lab is particularly interested in the development of efficient algorithms for parallel and accelerated computation of classical MM and hybrid QM/MM techniques. The research also focuses on the use of MM and QM/MM Molecular Dynamics algorithms for the determination of physical and chemical properties of enzymes.
Amber provides researchers with:

1. A set of molecular mechanical force fields for the simulation of biomolecules. These are in the public domain and are used in a variety of simulation programs.
2. A package of molecular simulation programs, including source code and demos.

The software and force fields are used for the simulation of enzymes, lipid membranes, advanced bio-materials, and catalysts. This impacts drug discovery—providing the ability to predict drug-binding affinity and accurately determine drug permeability in cells, as well as interaction with lipid-bound proteins.

“The Intel Math Kernel Library and Intel MPI Library implementations provide very large speed increases with minimal effort on the part of the developer, and are well worth the investment from an end-user perspective.”

– Professor Ross Walker
Associate Professor, Walker Molecular Dynamics Lab, SDSC, UCSD

The Challenge

Walker Laboratory sought to improve the vectorization of a mixed precision model within Amber’s highly optimized molecular dynamics engine (PMEMD) in order to achieve higher performance on Intel® architectures. It wanted to boost parallel scalability and build a solid foundation for support of Intel® Xeon Phi™ coprocessor offload acceleration. In addition, it sought to accommodate the use of Intel® Many Integrated Core Architecture (Intel® MIC Architecture) in Amber to meet customer and user requirements.

Based on previous experience, Walker Laboratory began making changes to Amber to enable the software to execute on the latest Intel® processors. But, this resulted in poorer performance than expected and it was difficult to determine how to make performance improvements. Initial assumptions regarding the way single- to double-precision casting would work within the code proved to be incorrect, and engineers were at a loss to explain why they were not achieving the anticipated performance improvements.
Amber is a computationally intensive application. As such, users demand that the software run on the most advanced HPC computer architectures available. Walker Laboratory approached Intel and, through the Intel® Parallel Computing Centers program, began a project to enable Amber to run on the Intel Xeon Phi coprocessor, and then to optimize performance on both Intel® Xeon™ and Xeon Phi processor architecture.

“The Intel VTune Amplifier XE is an invaluable tool for identifying hotspots when optimizing code. Its user interface is easy to use and informative, quickening the pace of development. Without access to Intel VTune's line-by-line performance counters, we would never have been able to identify the reasons why our mixed-precision code was running slower than our original double-precision code.”

– Dr. Perri Needham
Postdoctoral Researcher, Walker Molecular Dynamics Lab, SDSC, UCSD

The Solution

Working closely with Intel application engineer Ashraf Bhuiyan, the Walker Laboratory team was introduced to Intel® VTune™ Amplifier, as well as the latest features in Intel® MPI Library (Intel® MPI) and Intel® Math Kernel Library (Intel® MKL). Introducing support for Intel MPI provided an immediate improvement in parallel performance of between 10 and 20 percent, depending on MPI process count. Making use of Intel's optimized routines within Intel MKL also provided a performance boost over the modified version of pubFFT that Walker Laboratory originally used.

In addition, the use of Intel VTune Amplifier XE allowed the Walker Laboratory team to dig deeply into the code and identify a number of performance bottlenecks. In particular, it allowed them to identify where excessive casting of single-precision to double-precision code was occurring in the code, thus preventing its hybrid precision model from delivering
the expected speed improvements. With Intel VTune Amplifier XE, Walker Laboratory was able to locate the specific lines in the code where this was occurring and then escalate this to Intel® compiler engineers to determine why the compiler was unnecessarily casting. Without Intel VTune Amplifier XE’s ability to provide a line-by-line breakdown of timings, it would not have been possible to identify the location of this performance hotspot.

Ashraf helped with the Intel VTune Amplifier XE environment setup to identify hotspots in Amber, as well as with the tuning of Intel MPI for optimal performance. He developed the current working version for the offload code; joint development is still ongoing. Amber v14 now supports Intel MPI, which is configured using a flag when building. In addition, Ashraf assisted in ensuring the correct paths and environment for Intel MPI support and coordinated provision for Intel Xeon Phi coprocessor development hardware and associated software at SDSC.

## Results

### Performance Gains

Based on the tuning work done with Intel VTune Amplifier XE and the provision for using Intel MPI and Intel MKL, Walker Laboratory was able to deliver early performance improvements for explicit solvent particle mesh ewald simulations of approximately 20 percent for parallel scaling and 10 to 15 percent (depending on simulation specifics) serial improvement and implicit solvent generalized born simulations speedups of between 2.7 and 4.0x from the use of Intel MPI, the Intel compilers, and Intel MKL.

The following table shows detail performance improvements for sample data sets.

Provisional offload support for Intel Xeon Phi coprocessor adds about a 10 percent performance improvement for explicit solvent simulations of > 400,000 atoms, and work is ongoing, aided by Intel VTune Amplifier XE, to obtain longer-term performance gains from the Intel Xeon Phi coprocessor architecture.

<table>
<thead>
<tr>
<th>Model</th>
<th>System</th>
<th>System size</th>
<th>NANOSECONDS/DAY</th>
<th>NANOSECONDS/DAY</th>
<th>Speedup (INTEL/GNU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized Born</td>
<td>TRPCAGE (304)</td>
<td>304</td>
<td>63.57</td>
<td>176.12</td>
<td>2.77x</td>
</tr>
<tr>
<td></td>
<td>MYOGLOBIN (2,492)</td>
<td>2,492</td>
<td>1.17</td>
<td>4.43</td>
<td>3.79x</td>
</tr>
<tr>
<td></td>
<td>NUCLEOSOME (25,095)</td>
<td>25,095</td>
<td>0.01</td>
<td>0.04</td>
<td>4.00x</td>
</tr>
<tr>
<td>Particle Mesh Ewald-Nve Ensemble</td>
<td>JAC - 4fs (23,558)</td>
<td>23,558</td>
<td>13.09</td>
<td>15.36</td>
<td>1.17x</td>
</tr>
<tr>
<td></td>
<td>JAC -2fs (23,558)</td>
<td>23,558</td>
<td>6.88</td>
<td>8.01</td>
<td>1.16x</td>
</tr>
</tbody>
</table>
Productivity

Intel VTune Amplifier XE identified hotspots in the code, reducing development time. Specifically, it identified regions in the hybrid single/double precision model where unnecessary casting was occurring, thus reducing performance. Intel VTune Amplifier XE allowed Walker Laboratory to experiment with different code layouts to investigate why the compiler was optimizing the way it was. It also highlighted the key hotspots in the code as a function of different input settings, such as different cutoff sizes and different FFT grid sizes.

Note that there are four variants of the pme_direct_modget nb_energy. Intel VTune Amplifier XE gave Walker Laboratory the ability to split one routine into four function-specific routines and avoid if statements in inner loops, while still being able to track the cost of each function and prioritize further optimization.

The identification of specific code hotspots
Intel VTune Amplifier XE was critical in locating specific lines in the code that were taking more CPU time than predicted. Walker Laboratory looked further into this by finding the compiled machine code corresponding to these lines. It identified that the compiler was casting single-precision to double-precision code and back again without an explanation. Identification of the specific locations of these hotspots in the compiled executable allowed Walker to escalate this to Intel compiler engineers to investigate the problem.

Improved MPI performance enables simulations in less time and, more importantly, the ability to run simulations for longer. This allows users to carry out more simulations on different systems of interest, such as a larger range of potential drugs. It also helps them to improve the convergence of individual simulations and to look for rare events and longer timescale motions. This ultimately leads to better opportunities for scientific discovery.

Conclusion

Amber currently supports GPUs and is broadening support to include Intel Xeon Phi coprocessor architecture. Walker Laboratory anticipates that the flexibility of the Intel Xeon Phi coprocessor will allow users to run either multiple simulations simultaneously via the native mode or provide comparable speedups to the Amber GPU engine through offload mode.

Walker Laboratory is confident that future optimization and performance results, and in particular, a focus on the latest Intel Xeon Phi coprocessors, will enable substantially faster simulations. Amber users will benefit by being able to run longer simulations and conduct scientific research at a quicker pace. It will also open up new approaches to conducting research, such as providing the opportunity for real-time feedback from simulations allowing for interactive computation and computational experimentation on the desktop.
About

The research focus of the Walker Molecular Dynamics Laboratory—based at the San Diego Supercomputer Center at the University of California San Diego—encompasses the fields of computational chemistry, molecular biology, and high performance computing. The lab is particularly interested in the development of efficient algorithms, parallel- and GPU-accelerated computation of Quantum Mechanical and hybrid Quantum/Molecular Mechanical (QM/MM) techniques, as well as improvements in the computational efficiency and accuracy of classical MM dynamics simulations. The research also focuses on the use of MM and QM/MM Molecular Dynamics algorithms for the determination of physical and chemical properties of protein-based systems.
The Walker Laboratory is funded through a combination of grants from the National Science Foundation (NSF); National Institutes of Health (NIH); the University of California; Intel; NVIDIA; Microsoft; and the UK Foreign and Commonwealth Office, Department for Business, Innovation & Skills. Practical applications of the techniques developed in the lab include the development of next-generation viral inhibitors, improvements in bioethanol production, drug discovery, and advanced algorithms for high performance of scientific application on supercomputers.

Learn more: www.wmd-lab.org

About Intel® Software Development Products

Intel has been providing standards-driven tools for developers in the high performance computing industry for more than 25 years. Its industry-leading software tools include Intel® Parallel Studio XE Composer, Professional, and Cluster Editions.

Learn more: http://intel.ly/perf-tools

Learn more about Intel® software development tools at http://intel.ly/perf-tools.
Intel® Software Development Products
Win HPCwire Awards


“HPCwire goes through a very thorough process to get feedback from the community that we serve on who they think the impact makers are each year,” said HPCwire senior editor Tiffany Trader. “Once we’ve determined the nominations, based on community feedback, we put it out to our readers, who engage and tell us who they think are making the biggest impacts in the various award categories.”

Trader said HPCwire, a portal for science, technology, and business professionals interested in high performance and data-intensive computing, strives for objectivity in choosing candidates and award recipients. “Obviously, as an editorial group, we want to be careful not to endorse any one solution over another,” she said.

“We recognize Intel as a dominant figure in the HPC ecosystem with a portfolio of tools that play a significant role in addressing the needs of the community,” Trader explained. “Clearly, Intel is doing important things in parallel computing and HPC and getting attention for it from those out in the industry.”
Winner of the Best HPC Software Product or Technology – Editor’s Choice:
**Intel® Parallel Studio XE Cluster Edition – High Performance MPI Hybrid Cluster Development Tool Suite**

Intel® Parallel Studio XE Cluster Edition simplifies and speeds HPC and cluster application development, debug, and tuning. This advanced, comprehensive [C++ and Fortran](https://software.intel.com/en-us/articles/optimization-notice) tool suite helps scale development efforts with standards-driven compilers, programming models, and tools. These tools accelerate performance with built-in, intuitive parallel models, vectorization support, multifabric MPI library, and advanced MPI error checking and profiling.

Winner of the Best HPC Cluster Solution or Technology – Reader’s Choice:
**Intel® MPI Library**

Intel® MPI Library 5.0 focuses on making applications perform better on Intel® architecture-based clusters—implementing the high performance Message Passing Interface Version 3.0 specification on multiple fabrics. It enables maximum end user performance even with changes or upgrades to new interconnects, without requiring changes to the software or operating environment.

Intel® MPI Library 5.0 is part of Intel® Parallel Studio XE Cluster Edition.

---

**Try a 30-day free version of Parallel Studio XE Cluster Edition >**
Developers preparing for the next generation of Intel® Xeon Phi™ technology—codenamed “Knights Landing”—have all the more reason to learn new ways to exploit higher levels of parallelism from processors and coprocessors for optimization and high performance.

*High Performance Parallelism Pearls: Multicore and Many-core Approaches*, released last November, walks users through real-world examples—including source code—of how to leverage parallelism on processors and coprocessors using a standard set of programming methods.

Building off the momentum of their previous book, Intel® Xeon Phi™ Coprocessor High-Performance Programming, authors and Intel parallelism evangelists James Reinders and Jim Jeffers found an opportunity to create a “cookbook” of the inventive ways that numerous experts spanning a diversity of industries get the most from Intel® multicore and many-core processors.
Experts from around the globe

The roster of 69 contributors comprises a Who's Who of HPC experts from around the world representing academia (University of Tennessee, KU Lenven in Belgium, King Abdullah University of Science and Technology in Saudi Arabia); research institutions (Oak Ridge National Laboratory, Zuse Institute Berlin, CERN Norway, Danish Meteorological Institute); private consulting firms (QuickThread Programming, LLC, TechEnablement.com), and, of course, Intel Corporation.

As it covers a breadth and depth of subjects, those seeking parallel programming “pearls” of wisdom need not dig much. The 502-page book is separated into 28 chapters, with titles such as:

- Better Concurrency and SiDM on HBM
- Concurrent Kernal Offloading
- Heterogenous Computing with MPI
- Integrating Intel Xeon Phi Coprocessors into a Cluster Environment
- Portable Performance with OpenCL™

Each chapter shows in detail how to code for high performance. The source code sets from each example also are available for download on the book’s website, lotsofcores.com, which includes news and updates on the book from the authors.

With the industry striving for open standards to push for innovation, it’s no surprise that a second volume of High Performance Parallelism Pearls is in the works.

Buy High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches now on Amazon.com or the Elsevier Store.

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