White Paper

Healthcare
Diagnostic Imaging Systems

Accelerate Your Visualization Experience

Reduce Visualization Time with GE Healthcare and Intel

Authors

Beenish Zia
Lead Author, Intel Corporation

Shiran Afergan
Steve Johnson
Jared Pager
Co-Authors, Intel Corporation

S Antoine Aliotti
Yingpo Huang
Jerome Knoplioch
Yannick Leberre
Lionel Marmonier
Florentin Toulemon
Co-Authors, GE Healthcare

Table of Contents

Executive Summary

GE Healthcare, a division of General Electric (GE), is a global leader in medical technology and digital solutions innovation. GE helps clinicians make faster, more informed decisions through intelligent devices, data analytics, applications, and services.

GE Healthcare Advantage Workstation (AW) platforms provide radiologists quick access to imaging data in the office, in the hospital, at home, or wherever they need it.[1] Advantage Workstation Servers offer advanced visualization and diagnostic capabilities, enabling clinicians to share images quickly and easily.[2] They provide radiologists with fast access to medical images for a good overall visualization experience.

GE AW platforms take advantage of the power and performance provided by Intel® Xeon® processors, so when the opportunity arose for GE Healthcare and Intel to collaborate on an effort to further improve imaging performance, the decision was an easy one. This paper describes the various technical challenges we tackled and how, by working together, we were able to achieve an approximate 70% reduction in visualization time.[3] This performance improvement was a cumulative result of various hardware and software optimizations performed by the team, as described in the "Optimizing the Visualization Pipeline" section of this whitepaper. As shown, we were able to accelerate medical imaging visualization without a loss in accuracy.[4]

Importance of Reduction in Visualization Time

The average number of Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) scans taken every day has increased significantly over the last decade.[5] This has resulted in an increasing burden on radiologists, who must view more and more images every day. One way to reduce that burden is to cut the time it takes radiologists to wait for image studies to load. From the first click on the user interface to full availability of the complete set of scans, every second is critical. The more quickly radiologists can access and examine images, the sooner they can move on to the next patient's scans. By accelerating this process, we can significantly increase the number of images that radiologists can view each day. This can not only help reduce physician burnout[6], but it can help get results to patients more quickly so that the appropriate treatment can be started as rapidly as possible. The result is a win-win for both radiologist and patient.

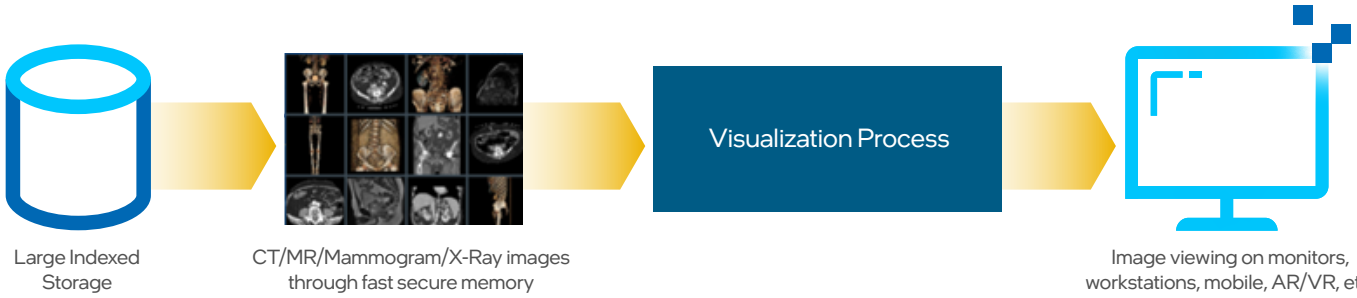


Figure 1. High-level Block Diagram of Medical Imaging Visualization flow.

Advantage Workstation (AW) Server Visualization

For GE AW servers, the high-level block diagram of visualization flow is shown in Figure 1. After a patient undergoes a radiology scan, the resulting images are initially stored in a large-indexed database. From there, they are accessed by an Intel Xeon processor-based AW server equipped with fast, secure memory. This server is responsible for running the visualization process, which is executed on Intel CPUs.

When the visualization process is complete, the images are ready for viewing in a variety of ways, including large viewing monitors, tablets, remote viewing stations (e.g., home laptops), Augment Reality (AR)/Virtual Reality (VR) devices, or even 3D printing.

The Visualization Pipeline

The visualization flow shown in Figure 1 consists of a sequence of steps that we call the visualization pipeline. In optimizing performance, we sought to accelerate each step of the pipeline process, with a goal of reducing processing time by a factor of 10x.

As shown in Figure 2, the visualization pipeline for 2D/3D volume images consists of three major steps. Together these steps cover the majority of the entire process, from the moment the radiologist accesses the user interface until the views are available for manipulation. The three steps are:

1. Initialization
2. Loading
3. Post-processing

Optimizing the Visualization Pipeline

To meet our objective of reducing visualization time for medical images, we needed to address two major questions: First, what hardware platform configuration would help us improve performance? And second, how could we optimize the code running on those systems to run as quickly and efficiently as possible, without sacrificing image quality?

Hardware Selection

Medical images require a large amount of storage space. A typical CT image slice is 512 x 512 x12 bits[7]. A thin-slice dataset can easily run to 500 images per study, and more extensive studies (such as CT angiography and cardiac CT) may require up to 2500 images. The result is that an average CT dataset runs from roughly 200MB up to 1GB per study. Similarly, MRI studies can require on average about 300MB of storage.[7]

In addition, the processing of these large datasets is demanding. Each step—from data ingestion, to processing, to visualization—is compute-intensive.

To handle these demands requires a powerful computing platform, and our team selected an Intel® Xeon® Scalable processor-based platform. To choose the specific Intel Xeon processor that would best meet our needs, we ran multiple tests on various hardware configurations. We compiled reference visualization code to serve as a baseline, and then ran trials to assess the impact of various cores, threads, and processor speeds. Because our engineering teams were spread across multiple countries around the world, we started with a remote cluster comprised of 2nd generation Intel Xeon Platinum 8260L processors. Eventually, after successful evaluation of the workload on the remote cluster, the GE Healthcare team obtained an on-premise 2nd generation Intel Xeon Gold 6256 processor-based system for further performance improvements. This decision was made because the application was more dependent on frequency than core counts; hence, for in-house evaluation, we chose Intel Xeon Gold 6256(12c/3.6GHz) over Intel Xeon Platinum 8260L (24c/2.4GHz), as it provided a cost-effective combination of frequency and core counts.

The use of remote clusters enabled our teams to test and evaluate scaling and frequency dependencies for specific segments of reference visualization code. As indicated in Figure 2, this code consisted of several different sections, each of which exhibited different behaviors.

Let's look at what happened when we examined volume rendering (VR), a particularly compute-intensive step in the reference visualization code. First, we ran the code on a 6-core Intel Xeon W-2135 processor-based baseline system. Next, we ran the code on a second generation Intel Xeon Platinum 8260L processor-based system with 24 cores (after making minor software changes). As shown in Figure 3, the results were as expected for moving non-optimized code from a 1-socket workstation to a 4-socket Intel Xeon Scalable CPU. **We saw an 87% reduction in rendering time.**[4]

After our basic testing established the benefits of moving from a single socket workstation to a 2nd generation Intel Xeon Scalable processor-based system, we moved to the next step of our investigation: software optimization.

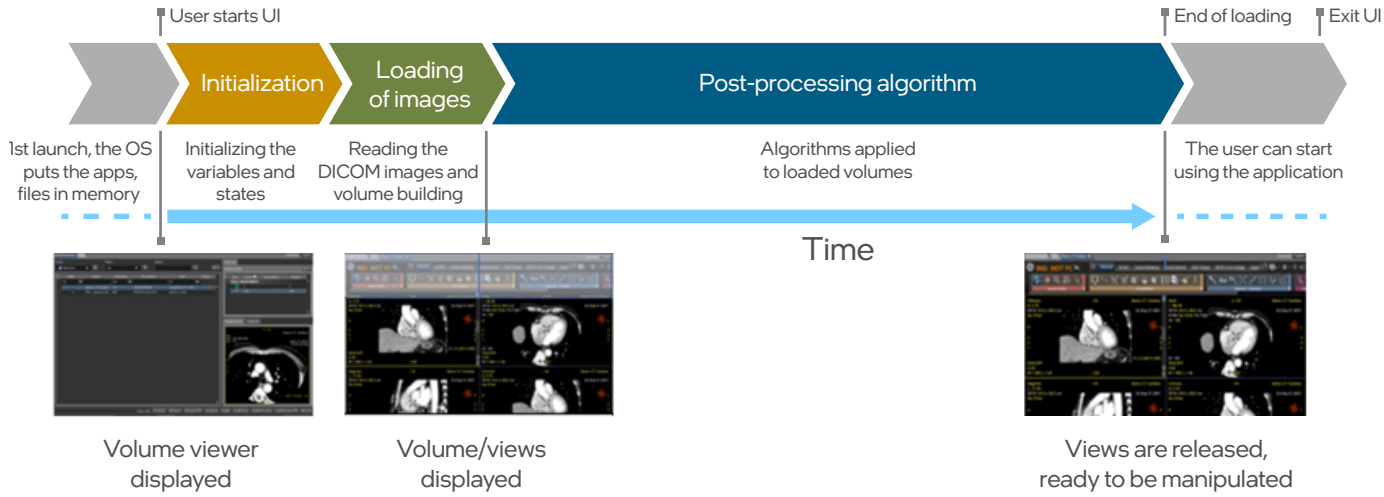


Figure 2. Volume Visualization Pipeline in a GE AW Server.

Software Optimization

Before we could make a final decision on hardware, we needed to ensure not only that the software involved in the visualization process ran well on those systems, but also that we could enhance the code to optimize its performance. To do this, we relied heavily on the **Intel® VTune™ Profiler**[8]. This Intel-optimized performance tool helps developers quickly analyze code performance, isolate problems and bottlenecks, and reach solutions more rapidly than would otherwise be possible.

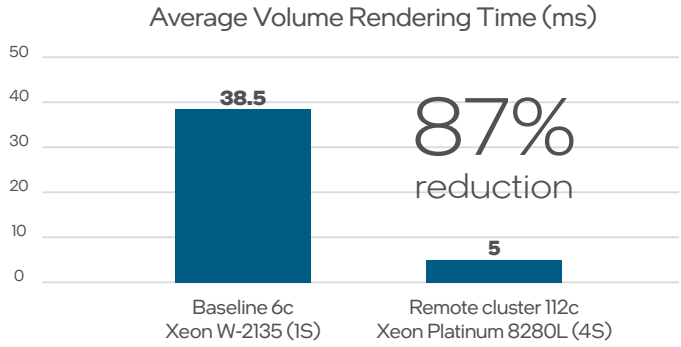


Figure 3. Volume rendering time comparison.

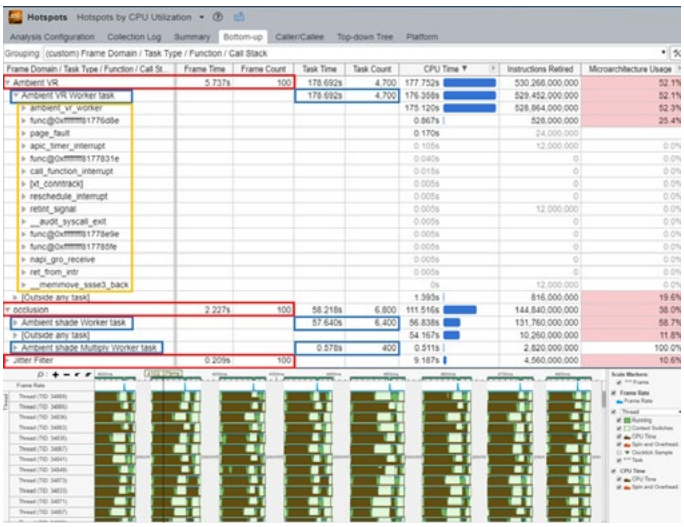


Figure 4. The Intel® VTune® Profiler user interface showing CPU utilization hotspots for the reference code.

Major software optimizations we investigated included:

- **Threading and Memory Allocations**
  We examined the code for opportunities to take better advantage of CPU threading. To do this, we used the OpenMP API[9], as well as Intel Threading Building Blocks[10] (Intel TBB), a C++ template library developed by Intel for optimizing parallel programming memory allocation on multi-core processors. Intel TBB enabled us to define tasks that could run in parallel, streamlining processing.

Figure 5 illustrates the results of our original tests, where we saw improved threading with OpenMP and memory allocations with the Intel TBB proxy. We were able to improve VR Rendering by 1.9x, occlusion by 3.0x, and jitter filter by a **remarkable 25x**.[4]

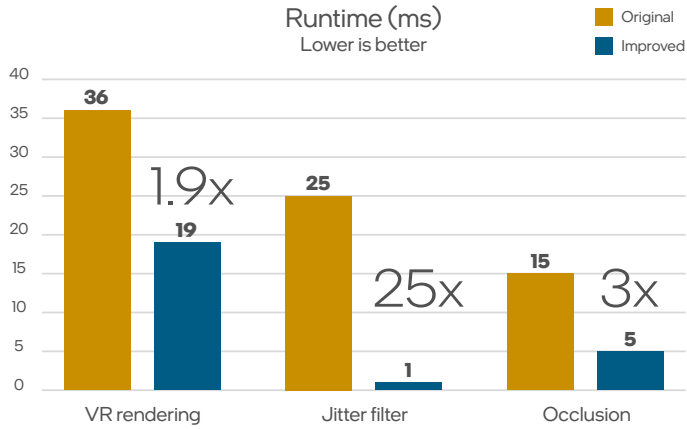


Figure 5. Original rendering runtime comparison, tested on an Intel Xeon Platinum 8260L CPU @ 2.40GHz, 24x2 physical cores.

Figure 7 shows the results of our second round of testing. Here we also saw improved threading, which had been a bottleneck for the original Software Development Kit (SDK). Axial plane processing performance increased by 2x, while sagittal plane processing performance saw **an improvement of 4x over previous results.**[4]

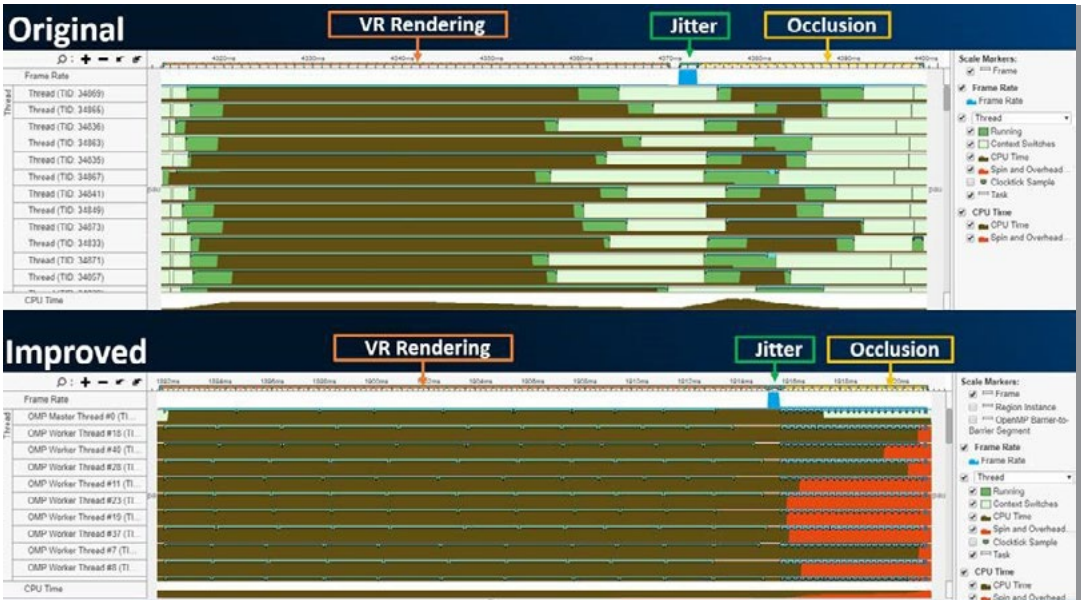


Figure 6. Snapshot of Intel® VTune Profiler showing original compared vs improved for thread utilization for same reference workload as in Figure 5.

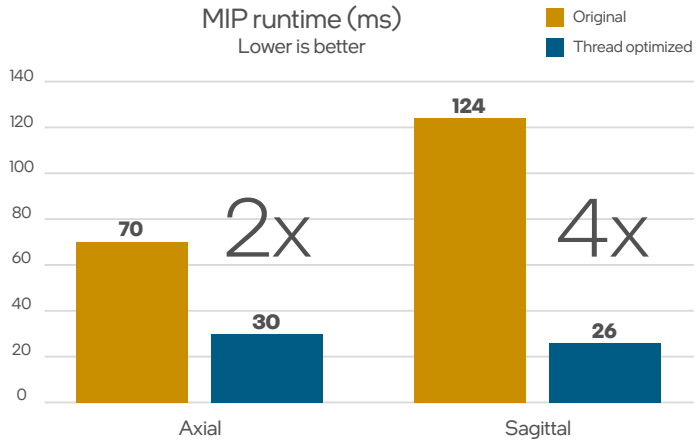


Figure 7. Rendering on the newer version. Threading and Memory Improvement.

- **Efficient Memory Usage**
  Next, we looked at Non-Uniform Memory Access (NUMA) design. By tuning the reference code using NUMA design principles, we were able to make better use of local memory and improve memory usage by 6x on larger input sizes[4], as illustrated in Figure 7. We tested rendering optimizations on an Intel Xeon Platinum 8260L processor-based system with the CPU running at 2.40GHZ.

  As shown in Figure 8, we were able to optimize code from the volume viewer, increasing memory efficiency for NUMA memory design by **at least 1.5x**.[4]

- **Post-processing Algorithm Optimizations**
  We also investigated how we could optimize post-processing algorithms. To do that, we compared three different approaches, as illustrated in Figure 9.

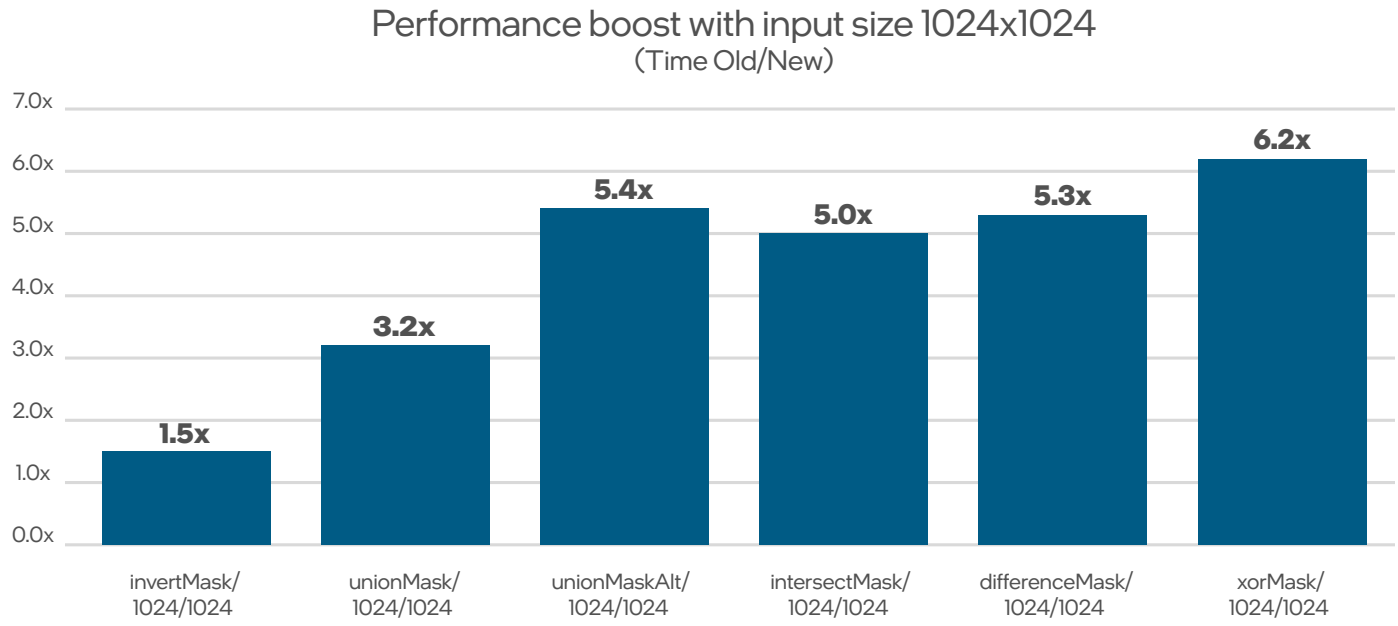
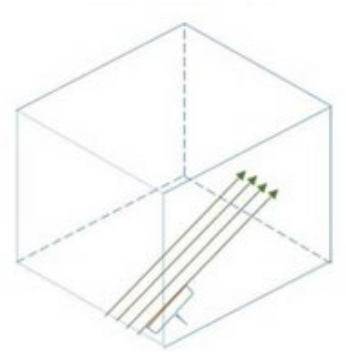


Figure 8. Volume Viewer. Performance boost with 1024 x1024 input size.



The Current Method

The original method used in the algorithm can process multiple voxels of a single ray in "parallel" using SIMD (Single Instructions/Multiple Data)[11].

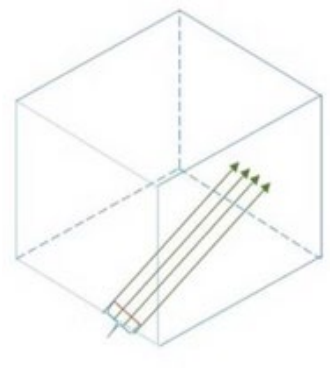
Advantages

- This is the most straightforward implementation, and it works well.
- This uses a workflow that takes advantage of some SIMD benefits.

Disadvantages

- There is almost no caching benefit, resulting in memory latency issues.

Note: A "voxel" is to a 3D model what a "pixel" is to a 2D image.



The Packet Method

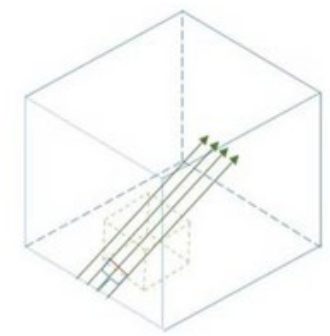
Instead of tracking multiple voxels on a single ray in parallel, the packet method traces a single voxel on multiple "nearby" rays in parallel. While this method can be implemented in Intel Advanced Vector Extensions[12] (Intel AVX), we recommend a minimum of Intel AVX2, due to the need to compare packed integers. (In some ways, this is the simplest method; each ray is independent, so there's no need to consider dependencies between voxels.)

Advantages

- Each "step" can use nearby data, increasing cache utilization.
- This method handles inter-voxel dependencies naturally.

Disadvantages

- Requires a minimum of Intel AVX2 for efficient implementation with 256-bit registers.
- Non-parallel rays quickly disperse from each other, decreasing performance benefits.
- The method still requires tracing the entire volume, so there is no "multi-packet" benefit.



The Multi-volume Packet Method

Several different methods fall into this category. Most of them involve dividing the sub-volumes, determining which rays pass through a given sub-volume, etc. In all cases, however, the idea is the same: divide the volume into several sub-volumes, then trace rays within that sub-volume. If the sub-volume is correctly sized, you can maximize access of cached volume data, regardless of how the rays relate to each other.

Advantages

- This method can be implemented using either the current method or the packet method.
- Any ray distribution can be improved with this method (as long as the ray intersections are easy to calculate).
- The volume can be sized to fit entirely in L1 or L2 cache, significantly improving memory latency. You will need to test to see which cache provides better performance.
- In volumes that are mostly "empty," this can eliminate traversing a part of the volume.

Disadvantages

- Rays that "terminate" prematurely can be difficult to work with.
- This method increases the complexity of the algorithm in general, especially in determining which rays intersect with a volume. Similarly, you will need to determine if a ray can be traced "simultaneously" in multiple volumes.
- Poorly defined volumes can result in poor SIMD utilization for rays that enter or exit the sub-volume quickly.

Figure 9. Analysis methods.

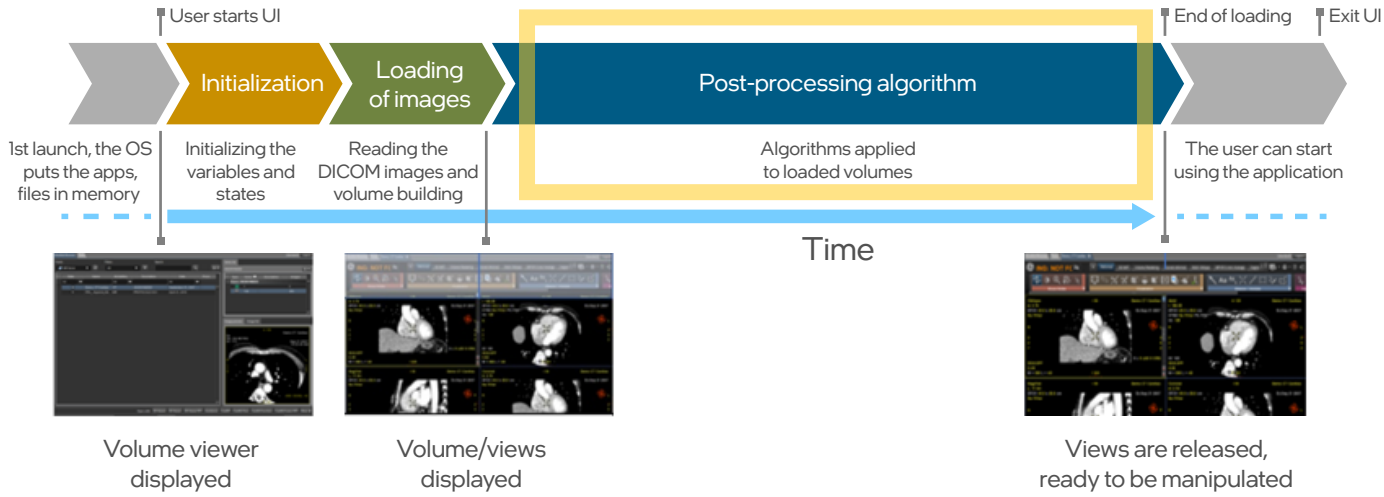


Figure 10. Post-processing algorithm.

By experimenting with the various pre-processing coding methods, we were able to record significant performance improvements. Let's look at an example: Maximum Intensity Projection (MIP) algorithm.

As shown in Figure 10, post-processing is the longest single step in the visualization pipeline. MIP is one of the early post-processing algorithms used after loading. MIP projects 3D data into a 2D space and is a necessary step before a user can rotate, turn, or otherwise manipulate images. MIP processing can be compute-intensive, and we knew that optimizing the MIP algorithm was critical to meeting our performance improvement objectives. Investigations into various optimization methods produced significant performance improvement.

Virtualized Setup Improvement

- MIP rendering single user (service test): **30-40% improvement.**[4]
- MIP rendering three parallel users (service test): **20% improvement.**[4]

Physical Machine

- MIP rendering benchmark testing (single user): **15% improvement** when camera's direction vector rotating on the axial plane, **30-40% improvement** on the sagittal plane.[4]

"High Level" Number

- Packet retracing rendering: **40% speed increase** for MIP.[4]

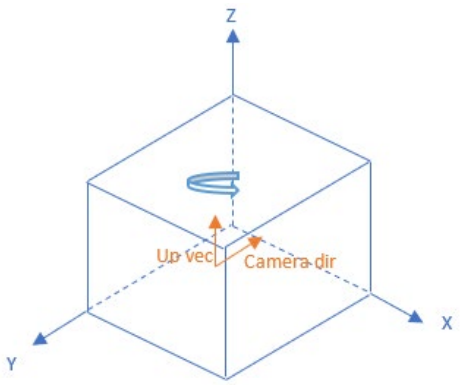
Conclusion – Working Together for a Better Future

In this paper, we have detailed the various methods we used to significantly reduce visualization time. We showcased improved performance in two distinct ways:

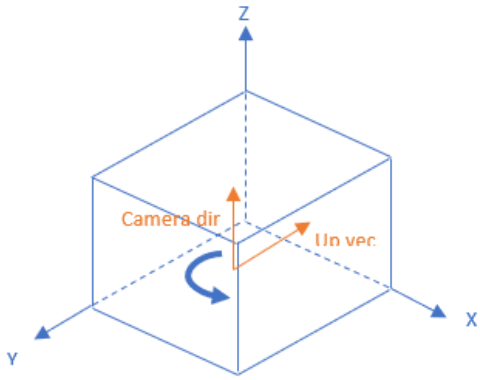
- **Hardware:** By moving processing to a more powerful Intel Xeon Scalable processors, we were able to achieve significant improvements.
- **Software:** By analyzing, experimenting, and tuning reference visualization code using Intel performance optimization tools, we were able to analyze code, optimize code for performance, isolate problems and bottlenecks, and reach solutions more quickly than would otherwise be possible. This resulted in a huge boost in performance, in addition to what the latest Intel hardware was able to achieve.

Thanks to collaborative teamwork, we were able to reduce the typical viewing time for the total visualization pipeline from ~17 seconds down to ~5 seconds. This represents close to a 70% reduction in visualization time needed.[4]

Note that these final or current visualization numbers were achieved on a 2S Intel Xeon Gold 6256 processor-based system, as most of the legacy code was not able to benefit from the high core count available on higher-end Intel Xeon Platinum processors; in addition, this CPU provided the team a more cost-effective solution. However, there is still room for further optimization and updating of the legacy code to achieve >70% reduction in time, and the team continues to collaborate on this.



Axial



Sagittal

Optimization of the visualization pipeline is just one part of the medical imaging workflow challenge. We at Intel and GE Healthcare plan to continue collaboration into the future to further reduce radiologist viewing times for medical images and improve the patient experience. Look for further whitepapers in the future.

To Learn More

GE Healthcare
gehealthcare.com/about

GE AW Server
gehealthcare.com/products/advanced-visualization/platforms/aw-server

Intel Xeon Scalable Processors
intel.com/xeon

Intel Software Tools for Developers
intel.com/software

1 GE AW Platforms - https://www.gehealthcare.com/products/advanced-visualization/platforms/aw-server

2 GE Healthcare AW Server - https://www.gehealthcare.com/products/advanced-visualization/platforms/aw-server

3 Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

4 See backup for configuration (config 1) details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. Refer to https://software.intel.com/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

5 https://appliedradiology.com/articles/the-radiologist-s-gerbil-wheel-interpreting-images-every-3-4-seconds-eight-hours-a-day-at-mayo-clinic

6 Radiologist Burn Out – GE healthcare - https://www.gehealthcare.com/feature-article/decreasing-radiologist-burnout

7 Society for Imaging Informatics in medicine - https://siim.org/page/archiving_chapter2

8 Intel® VTune profiler - https://software.intel.com/content/www/us/en/develop/tools/vtune-profiler.html

9 OpenMP - https://en.wikipedia.org/wiki/OpenMP

10 Intel® Thread Building Blocks - https://software.intel.com/content/www/us/en/develop/tools/threading-building-blocks.html

11 SIMD -https://en.wikipedia.org/wiki/SIMD#:~:text=Single%20instruction%2C%20multiple%20data%20(SIMD,on%20multiple%20data%20points%20simultaneously.

12 Advanced Vector Extensions

Configurations

Config 1: Head Node: 2nd Gen. Intel® Xeon® Platinum 8268OL, 12 X 64GB DDR4 DIMMS 60L, 12 X 512GB Intel® DCPMMs, 1 X P4510 (8TB) SSD, 1 X S3520 (480GB), Centos 7.6 OS, running Intel® Parallel Studio XE 2019 Update 4 Cluster Edition for Linux and Intel® OpenVINO 2019.2 toolkit.

Compute Node: 2nd Gen. Intel® Xeon® Platinum 8260OL,12 X 64GB DDR4 DIMMS 60L, 12 X 512GB Intel® DCPMMs, 1 X P4510 (8TB) SSD, 1 X S3520 (480GB), Centos 7.6 OS, running Intel® Parallel Studio XE 2019 Update 4 Cluster Edition for Linux and Intel® OpenVINO 2019.2 toolkit.Application was volume rendering, jitter, occlusion and MIP algorithm, all these applications were provided by GE Healthcare as reference visualization codes in C++.

Test performed by Intel Corporation in August 2019 and May 2020 for OpenMP optimizations.

Config 2: 2S 2nd Gen. Intel® Xeon® Gold 6256 (3.6GHz/12-core/205W), 12x 32GB RAM, 12x 128GB PMem, 2x 400GB SSD, 8x 1.92TB SSD read intensive. Linux OS, running Intel® Parallel Studio XE 2019 Update 4 Cluster Edition for Linux and Intel® OpenVINO 2019.2 toolkit.

Application was volume rendering, jitter, occlusion and MIP algorithm, all these applications were run by GE Healthcare as reference visualization codes in C++.

Tests performed by GE Healthcare in May 2020.

Notices & Disclaimers

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 complete information visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

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.

Refer to http://software.intel.com/en-us/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.