Scalability of a cross-platform multi-threaded non-sequential optical ray tracer

Alan W. Greynolds*
Ruda-Cardinal Inc., 1101 N. Wilmot, Tucson, AZ USA 85712

ABSTRACT

The GelOE optical engineering software implements multi-threaded ray tracing with just a few simple cross-platform OpenMP directives. Timings as a function of the number of threads are presented for two quite different ZEMAX non-sequential sample problems running on a dual-boot 12-core Apple computer and compared to not only ZEMAX, but also FRED (plus single-threaded ASAP and CodeV). Also discussed are the relative merits of using Mac OSX or Windows 7, 32-bit or 64-bit mode, single or double precision floats, and the Intel or GCC compilers. It is found that simple cross-platform multi-threading can be more efficient than the Windows-specific kind used in the commercial codes and who's the fastest ray tracer depends on the specific problem. Note that besides ray trace speed, overall productivity also depends on other things like visualization, ease-of-use, documentation, and technical support of which none are rated here.

Keywords: Ray tracing, software

1. INTRODUCTION

Design optimization increases the computational workload of engineering calculations by orders-of-magnitude. Multiple-core computers are now the norm (even on laptops) so that all computationally intensive algorithms should be transparently multi-threaded if possible. Ray tracing is still the work-horse of optical engineering where it's usually sequential for image analysis and non-sequential for illumination or stray light calculations. The question is how well do various implementations scale with the number of threads, cores, and/or processors. Therefore, the following is more computer than optical science.

2. THE MULTI-CORE DUAL-BOOT HOST COMPUTER

The benchmarking computer system is a 12-core Apple Mac Pro (mid-2010 model) whose hardware components are summarized in Table 1. It is set up to boot two separate "duel"ing 64-bit operating systems: the native Mac OSX 10.6 (Snow Leopard) and Microsoft Windows 7 (Professional). To both operating systems, the hardware appears to have 24 "CPU"s but they are not completely independent.

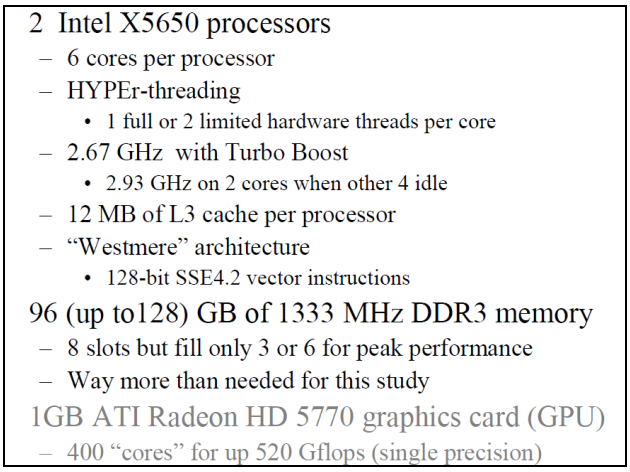
2.1 Cross-platform (including Linux) system software

Two different Fortran-200x compilers with OpenMP support were installed on both operating systems, the $700 Intel and the free GCC (Gnu Compiler Collection). Although they are capable of auto-parallelization, this feature tended to be erratic and ineffective so it was disabled by simply using their default "fast" optimization. However, both can successfully auto-vectorize small loops within threads like scalar dot products. The Intel compiler tends to be better at in-lining procedures and performing inter-procedural optimizations.

On both platforms, all graphs were done by the latest 4.4 version of the venerable GnuPlot software [1] (whose first version came out in 1986!). The well-known Ghostscript package was then used to convert the Postscript output to PDF.

* awgreynolds@ruda.com; 520 546-1495 x202; www.ruda.com

Table 1. Summary of 12-core Apple Mac Pro computer system's hardware components.



3. RAY TRACING APPLICATIONS TESTED

Unless stated otherwise, all codes tested here are 64-bit applications and traced rays to double precision floating point accuracy.

3.1 Windows commercial

The commercial multi-threaded ones are **FRED** 10.70 from Photon Engineering (Tucson, AZ) and a 2011 version of **ZEMAX** from Radiant-ZEMAX (Redmond WA), both written in Microsoft C/C++. For comparison purposes two single-threaded 32-bit codes with Visual Fortran kernels will also be tested; a LUM calculation in **CodeV** 10.2 from Synopsys formerly Optical Research Associates (Pasadena CA) and **ASAP** 2005 from Breault Research (Tucson AZ). All of these were selected because they were readily available to the author, and although FRED, CodeV, and ASAP don't support direct import of ZEMAX non-sequential models, they do indirectly via GelOE.

3.2 In-house cross-platform GelOE

3.2.1 Background

Pronounced like the dessert, GelOE is based on a proprietary "geometry-element" or just "gel", a flexible and efficient unit of geometry that bridges CAD and optical representations. It has been specifically designed to handle/merge arbitrarily complex CAD and optical models [2]. For example, within 16GB of RAM it can trace/track 25 million rays through a model with 10 million distinct surfaces. It should be noted that this simulation took over 1000 CPU hours!

Developed simultaneously on Mac OSX (PPC/Intel) and Windows (32/64-bit), GelOE is a command-line (i.e. console or terminal or batch) application. However, it can optionally spawn processes for interactive editing and graphics. Written entirely in modern Fortran (meaning according to post-1995 standards), GelOE consists of approximately 40 thousand actual statements contained within 900 separate procedures. It extensively makes use of modern Fortran's fast concise array (vector/matrix) syntax and dynamic storage in "modules" instead of static in "common blocks". Also, not a single "statement label" is used (i.e. no "goto"s) leading to highly structured program flow. However, the latest Fortran 200x object-oriented features resulted in slower runtimes and thus have been avoided. Table 2 is a basic overview of GelOE's internal operations and the following is a brief history of its development:

2005: Started as file conversion utility and pre/post-processor for ASAP
2006: Added its own ray tracing (non-sequential and differential)
2007: **Multi-threaded** using a few simple OpenMP directives
2008: Added GeWiz (wizard-style GUI front-end for file conversions) [3]

Table 2. Typical GelOE internal operations which can be thought of as just file conversions (although some can be quite complex).

`geloe [options*] import_list [options*] export_list [options*]`

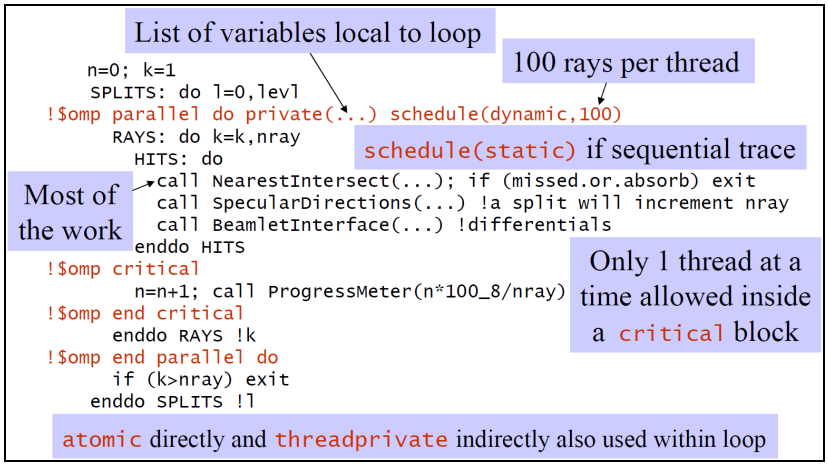
File Import*	ZEMAX CodeV ↓	ASAP-like scripts ↓	CAD ↓	Source rays ↓
Internal Databases	ZEMAX-like → ↓	Optical → ↓	Gel Objects → ↓	**Ray trace** ↓
File Export*	ZEMAX	CodeV ZEMAX	Graphics ASAP	Results Graphics

* currently there are over: 90 separate options (switches), 35 import file types supported, and 30 export types

3.2.2 Multi-threading with OpenMP

OpenMP is a cross-platform SMP (Shared Memory Parallelism) protocol for Fortran and C/C++ compilers [4]. Directive-based, it makes it trivial to support both serial and parallel versions with the exact same source code and has been a well-established standard since 1997 (latest 2011 version is 3.1). Table 3 is the source code for the GelOE ray trace loop but collapsed down from over 200 lines (not counting the hundreds of lines in called procedures) to show just the highlights. With OpenMP (and any multi-threaded programming) it's important to hoist outside of parallel loops as many updating operations on shared variables as possible. It's also crucial to select the right thread "schedule" to balance the load among threads, i.e. use the most efficient "static" schedule if possible, otherwise experiment to find the optimum "chunk" of work per thread. It should be noted that the quality of any OpenMP implementation is both compiler and operating system dependent. However as will be shown later, for a small amount of effort it can be very competitive with compiler/OS-specific "hand-coded" threading.

Table 3. Collapsed GelOE ray trace loop with OpenMP directives in red (just comments if OpenMP not enabled or supported).



4. TEST METHODOLOGY

When comparing the ray trace applications, it's important to "level the playing field" by making sure they are working the exact same problem. By starting from the same ZEMAX non-sequential files, the exact same geometry is guaranteed

if its creation for each application is by file transfer, i.e. as "hands-free" as possible. This is relatively easy because GelOE can accurately import then optimally translate and export a useful subset of ZEMAX entities including 26 surface, 42 object, 12 source, and 3 detector types. Also by loading the incoherent source from a .SDF or .DIS binary file, each application starts with the exact same rays. However, it was necessary to increase the original ZEMAX number of rays so that the fastest time is at least a few seconds. And finally the ray traces were forced to operate under the exact same conditions by turning off any polarization, intermediate detectors, and quadrant mirroring (coatings, ray filing/drawing were already off). Assuming no involved tweaking or "cheats" (like the ASAP SEARCH command), all the applications should be fairly timed while calculating the same illumination pattern.

A few of definitions are needed in order to properly interpret the results in the following sections. First, "Mrays/sec" is just an abbreviation for millions of rays traced per second, not counting any ray creation or analysis time. Second, "strong scaling" measures the departure from an ideal linear speedup with a percent efficiency defined as:

$$\%Efficiency = 100\frac{one\_thread\_time}{\#threads \times elapsed\_time} \tag{1}$$

And finally, the "0" thread refers to a serial (non-parallel) version of GelOE which is compiled separately without any multi-threading capability.

The specific test procedure includes several important steps. On Windows, run the *TMonitor* utility [5] to disable the erratic Turbo Boost and then set the process priority to higher than normal to block other processes. On both platforms, automatically cycle from 24 to 0 threads then back to 24 using shell scripts (macros for ZEMAX and FRED) and show "both" curves to highlight any variations in lieu of averaging over many time-consuming separate runs.

5. FIRST TEST CASE

The first test case is the file "Digital_projector_flys_eye_homogenizer.ZMX" [6], originally contributed by Michael Pate who sadly passed away last year. It was selected solely because it started with the most rays of any "non-trivial" ZEMAX non-sequential sample file. For our purposes the number of rays was increased even further from one to five million and fetched from a previously created .SDF binary file.

5.1 Computations

Figures 1, 2, 3, and 4 are both 3D system visualizations and the computational results from ZEMAX, GelOE, ASAP, and FRED, respectively. The important thing to note here is the excellent agreement between all of them. A CodeV LUM calculation [7] is also shown in Figure 5 even though it cannot use the exact same source and is not strictly a non-sequential ray trace because it does not include the rays that go directly from the source to the first lenslet array. Even so its results turn out to be very close to the others.

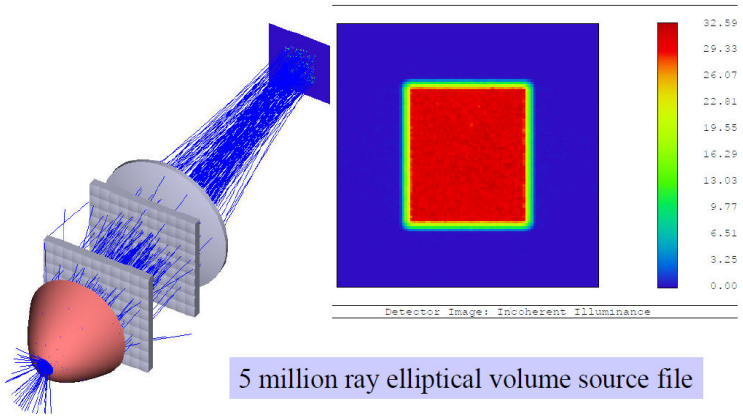


Figure 1. ZEMAX output for "Digital_projector_flys_eye_homegenizer.ZMX" sample file.

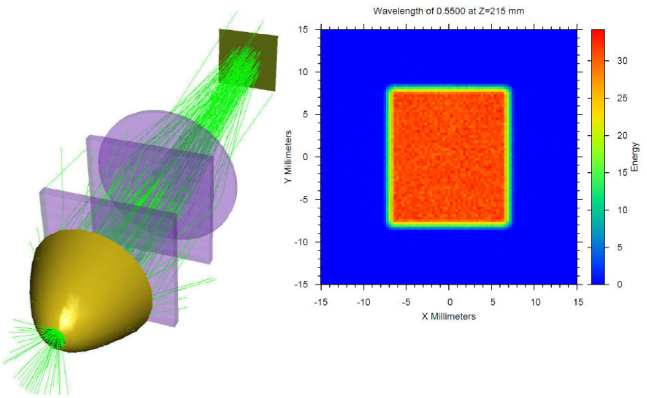


Figure 2. GelOE output after importing directly from .ZMX (and .SDF) file.

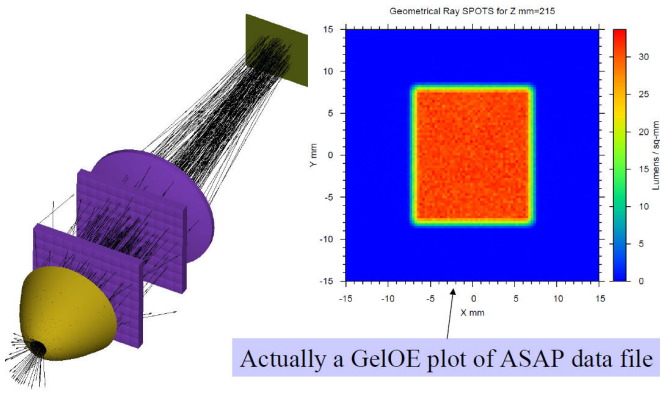


Figure 3. ASAP output after GelOE translated .ZMX to .INR and .SDF to .DIS files.

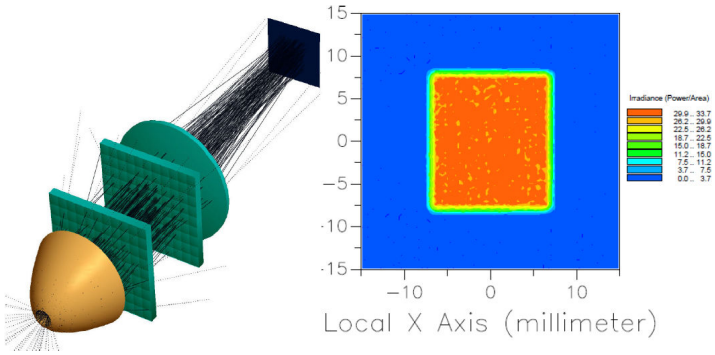


Figure 4. FRED output (GelOE then ASAP used for import).

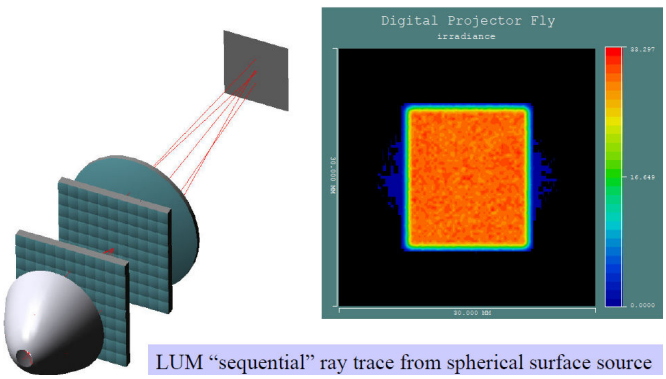
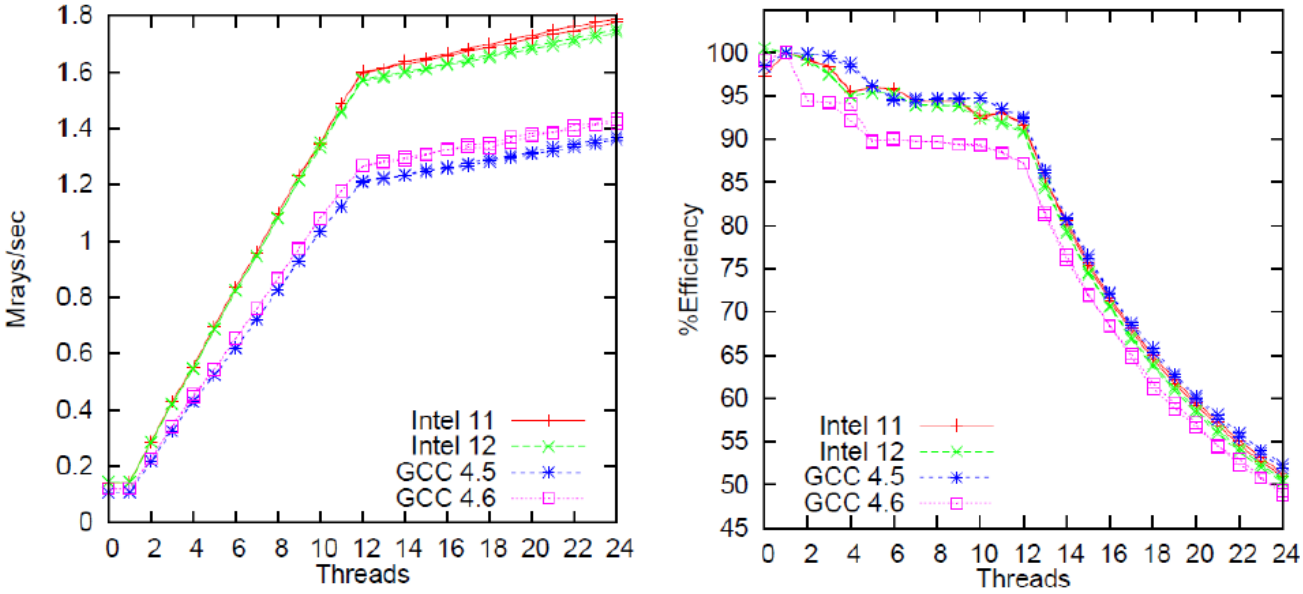


Figure 5. CodeV output after GelOE translated .ZMX file to .SEQ

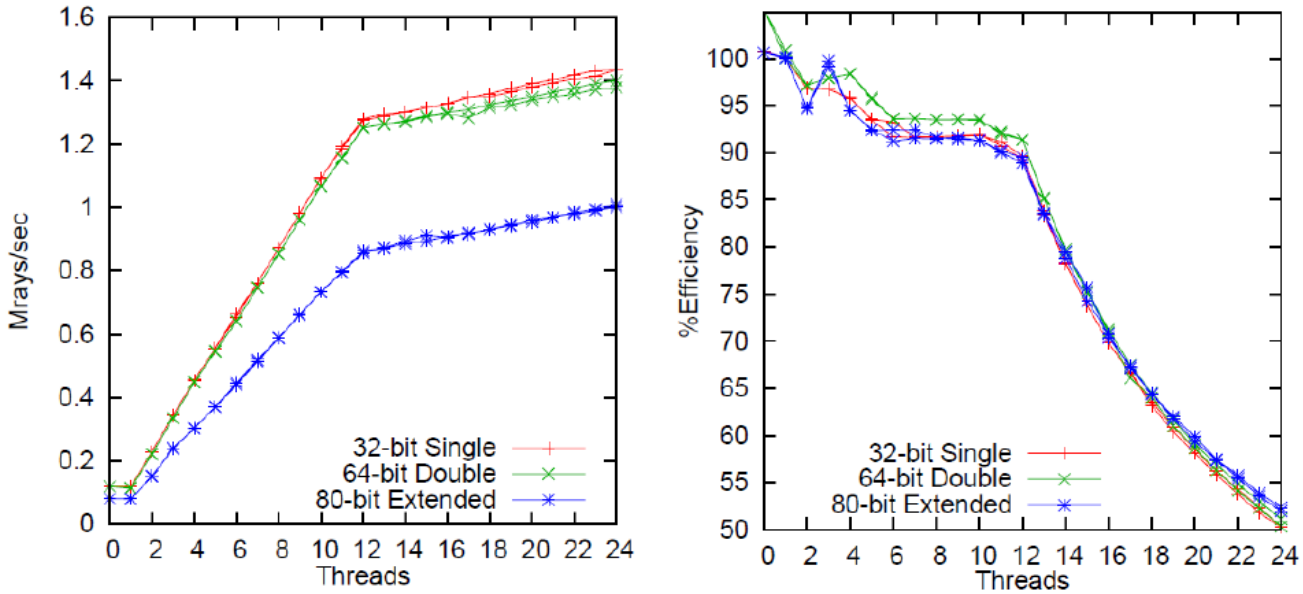
5.2 Timings

Starting with timings of GelOE on Mac OSX, Figures 6a and 6b are the results of using the two most recent releases of the two compilers. In all cases, the speedup with number of threads is very linear (greater than 85% efficient) up though 12 threads but drops off significantly past where HYPEr-threading has to be used.



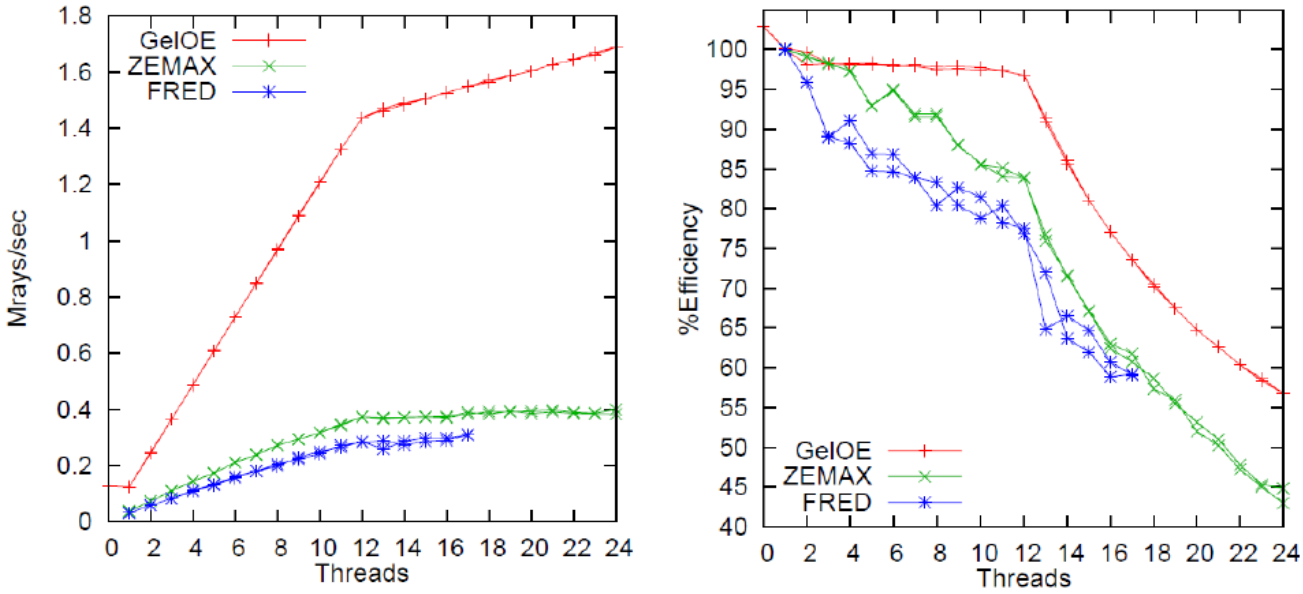
Figures 6a and 6b. Comparison of compilers on Mac OSX (HYPEr-threading kicks in after 12 threads)

Also, Figures 7a and 7b show the effects of GelOE using three different hardware-supported floating point precisions: 32-bit single (7 significant digits), 64-bit double (15 digits), and 80-bit extended (18 digits). The software-emulated 128-bit quad precision is excruciatingly slow but supports 33 significant digits which can be very useful at times.



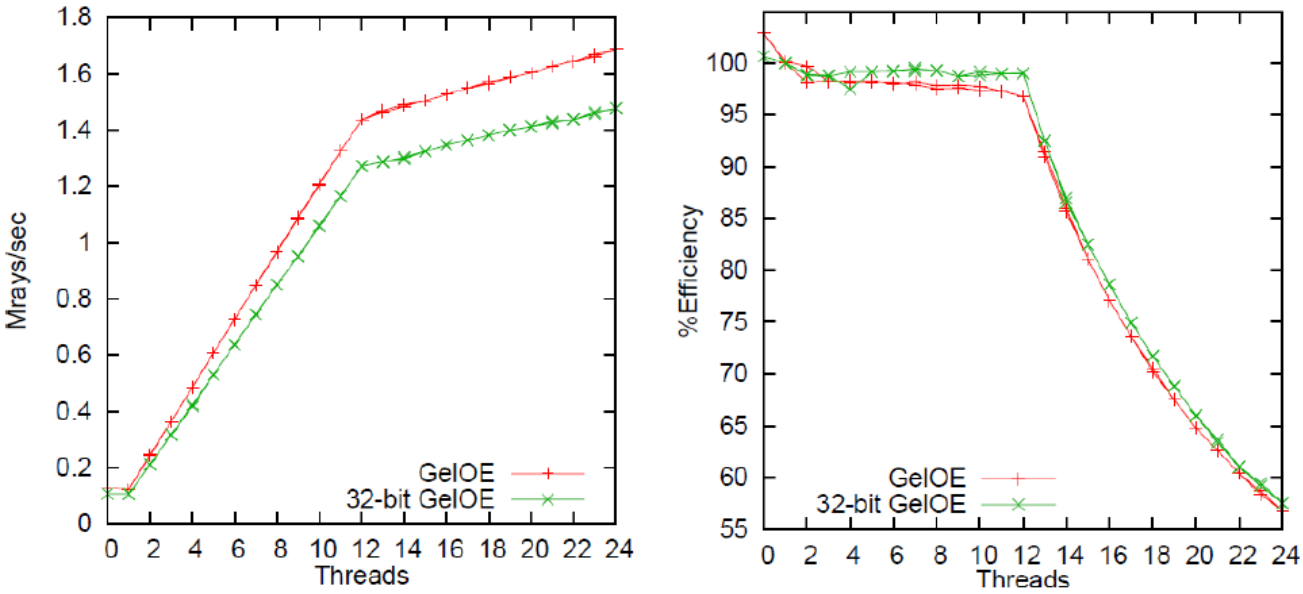
Figures 7a and 7b. Floating point precision comparison (software emulated 128-bit Quad precision 17 to 32 times slower than Single).

Moving to the Windows 7 platform, Figures 8a and 8b compare the speed and efficiency, respectively, of the three multi-thread applications.



Figures 8a and 8b. Multi-threaded application comparison on 64-bit Windows 7 (ASAP ~20% faster than ZEMAX single thread).

To be honest, GelOE has an advantage here since it can be compiled specifically for the host computer's CPU architecture while the commercial codes must still be able to run on the oldest AMD 64-bit CPUs. Figures 9a and 9b show what happens when GelOE is also compiled for really old 32-bit CPUs (going all the way back to the "686" Pentium Pros of the mid 1990's) but still run on the 64-bit host. The speed hit is not too bad (a little over 10%) while the efficiencies are virtually identical.



Figures 9a and 9b. 32-bit addressing and generic x86/87 instructions versus host 64-bit.

With significant additional user inputs, it's possible to set up the non-sequential applications (e.g. ASAP SEARCH and ZEMAX Consider lists) to perform a single-threaded sequential ray trace like CodeV. The non-sequential and sequential ranked results are shown in Table 4 (Turbo Boost not disabled here). The top three ray trace algorithms turn out to be written in Fortran while the last two in C/C++. CodeV is the fastest because it uses a computational shortcut to determine which cell of the packed regular-Cartesian lenslet arrays that a ray intercepts. This is different from at least GelOE which treats each cell as possibly completely independent with respect to position and/or orientation.

Table 4. Application comparison on Windows 7. Single thread times for 5 million rays ("sequential" ray trace results in parentheses).

Application	Elapsed time [sec]	Krays/sec
CodeV	(21)	(238)
GelOE	**35** (25)	**142** (200)
ASAP	**108** (67)	**46** (75)
ZEMAX	**128** (112)	**39** (45)
FRED	**148** (68)	**34** (74)

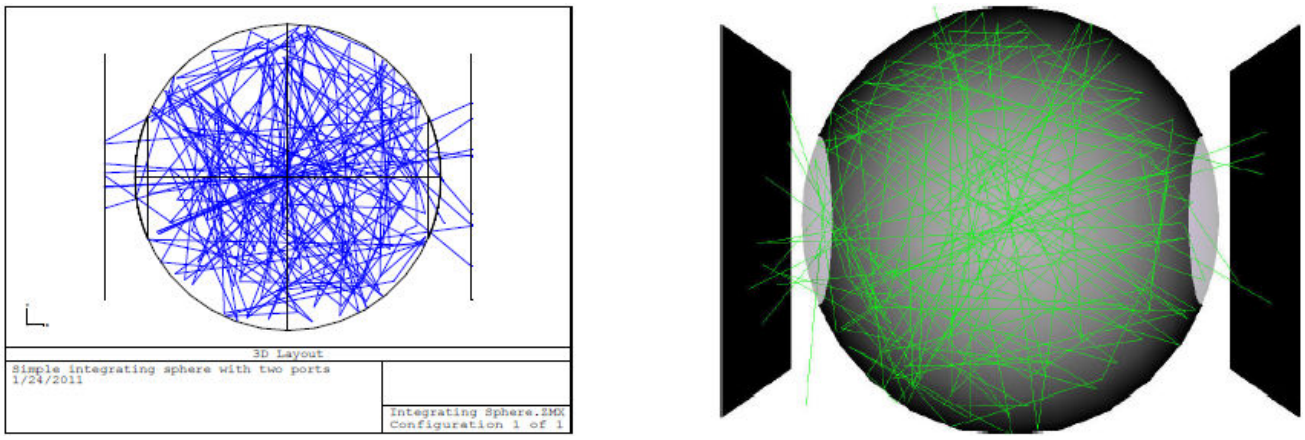
5.3 Summary

There are several general conclusions that can be made at this point. HYPEr-threading is of some benefit but it is not the same as having another separate core. The Intel compiler generated code is approximately 25% faster than the GCC which is surprisingly close given their different prices and origins. Single precision is no faster than double as has been the case for Intel CPUs for a long time. Mac OSX a bit faster but scaling is better on Windows 7.

For this one test case, GelOE scales better than ZEMAX, ZEMAX better than FRED. In terms of raw speed, GelOE is 3.3 to 3.9 times faster than ZEMAX and 3.7 to times 5.0 faster than FRED.

6. SECOND TEST CASE

The next case is the file "Integrating Sphere.ZMX", taken from the same ZEMAX installation directories as before and selected for being significantly different from the first test case. It consists of a sphere with a 100% Lambertian interior and two transparent exit ports (Figures 10a and 10b). A one million ray isotropic point source (previously created .SDF binary file) is placed at its center. On the average it takes 12 Monte-Carlo scatter bounces for a ray to escape and reach one of the detector planes with a few rays requiring nearly 200.



Figures 10a and 10b. ZEMAX and GelOE plots of "Integrating Sphere.ZMX" test case.

Due to differences in random number generators, it's not possible for all four applications to produce exactly the same final results but as seen in Figure 11, they are statistically "identical" (taking also into account differences in graphical color palettes).

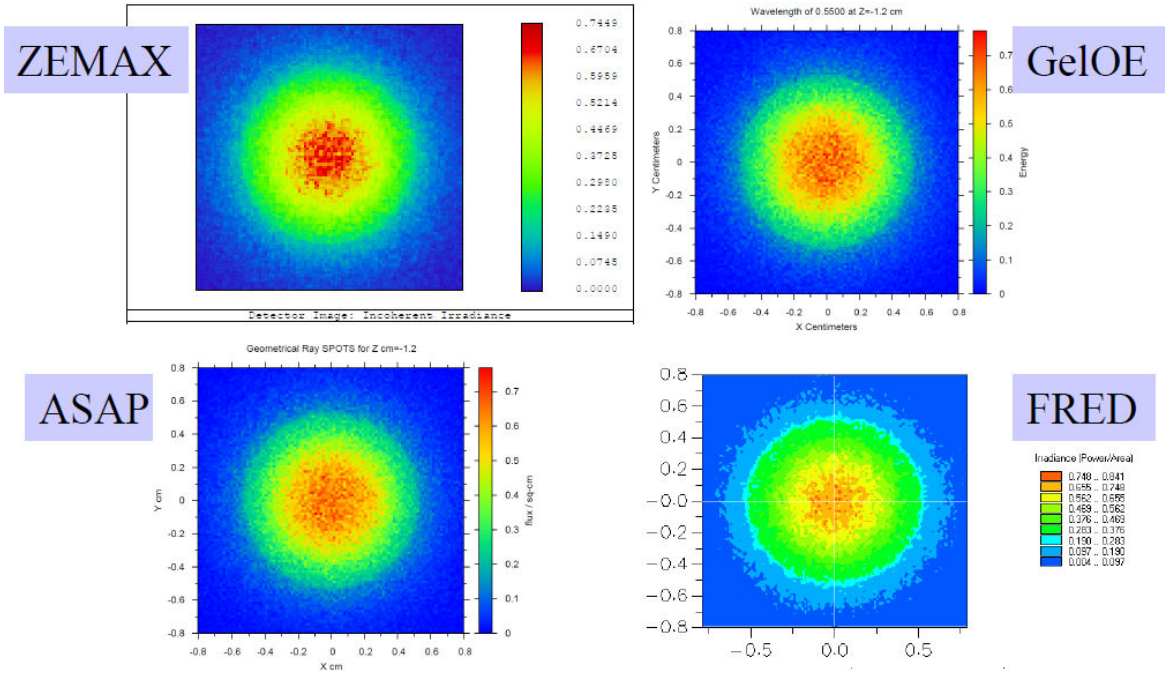
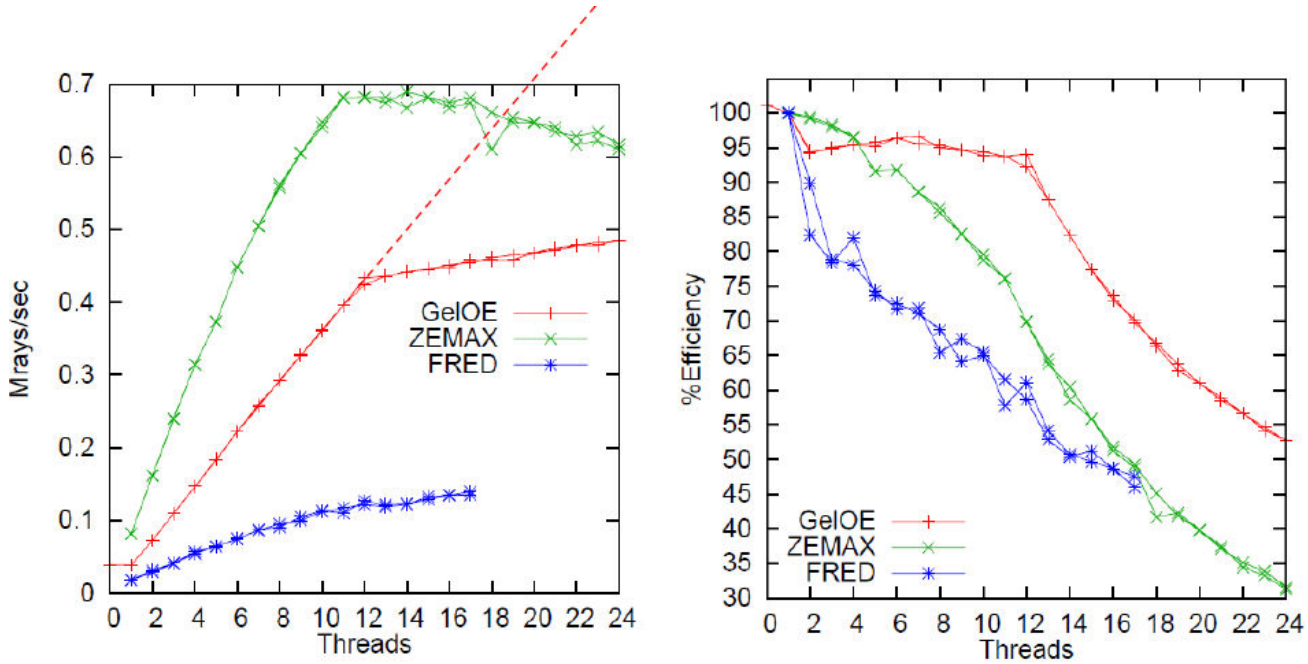


Figure 11. Output distributions for one port (other one statistically identical in all cases).

Figures 12a and 12b show that for this test case ZEMAX is now faster than GelOE. However, GelOE's multi-threading efficiency still remains superior overall to the commercial applications.



Figures 12a and 12b. Multi-threaded application comparison on Windows 7 (ASAP virtually same speed as GelOE single thread).

7. CONCLUSIONS

For this second test case the tables have turned, ZEMAX is now 1.7 to 2.1 times faster than GelOE (and 4.7 to 6.1 times faster than FRED). However, GelOE scales significantly better than ZEMAX (and FRED) so that for 24 or more *actual* cores (i.e. full threads), GelOE would probably end up being faster than ZEMAX.

Overall, it can be concluded that simple OpenMP cross-platform multi-threading can be more efficient than the Windows-specific kind used in the commercial codes. Who's the fastest ray tracer depends on the problem and we have only scratched the surface with these two test cases. It also needs to be pointed out that besides ray trace speed, overall productivity in using any software also depends on other things like visualization, ease-of-use, documentation, and support.

8. CURRENT AND FUTURE DEVELOPMENTS

The whole computer industry is moving rapidly towards pervasive parallel computing. For example, the latest Fortran-2008 standard includes "Coarrays", a built-in parallelism using an elegant array-like syntax. Unfortunately, stable and efficient implementations are not yet in the most popular compilers. On the hardware side, Intel's newest CPUs have 10 (and soon to be more) cores and their "Sandy Bridge" architecture includes AVX, an even more powerful set of vector instructions.

Virtually every engineering workstation today has the potential to be a "supercomputer" thanks to computer gaming. Current graphics cards can have hundreds to thousands of parallel floating point units (albeit as opposed to the Intel CPU, double precision can be significantly slower than single). Unfortunately, these "GPU"s (Graphical Processing Units)" have not yet been tapped for optical engineering in part due to competing programming standards: OpenCL (ATI, Apple), CUDA (NVIDIA), DirectCompute (Microsoft), OpenACC (Portland Group), and HMPP (CAPS France) to name the major ones. Although GPUs are already being applied to accelerating CGI ray tracing (e.g. NVIDIA's OptiX framework), it's an open question whether they can also be used effectively for optical ray tracing and engineering tasks that usually require double precision floats.

ACKNOWLEDGEMENTS

Thanks to Photon Engineering for providing a temporary FRED Optimum license.

REFERENCES

[1] P. K. Janert, *Gnuplot in Action: Understanding Data with Graphs*, Manning Publications, Greenwich CT (2009).
[2] A. W. Greynolds, "Stray light computations: Has nothing changed since the 1970's?", Proc. SPIE, Vol. 6675 (2007).
[3] A free version of the GeWiz (GelOE Wizard) optical file translation utility can be obtained by sending an e-mail to software@ruda.com (specify if for a 32 or 64 bit version of Windows).
[4] R. Chandra et al, *Parallel Programming in OpenMP*, Academic Press, London (2001).
[5] http://www.cpuid.com/softwares/tmonitor.html
[6] http://www.zemax.com/kb/articles/91/1/Flys-Eye-Arrays-for-Uniform-Illumination-in-Digital-Projector-Optics/Page1.html
[7] Optical Research Associates, *CodeV 10.3 Reference Manual,* Vol. IV, 25-13 (2011).