FAQ: Why is VirtualLab Fusion not using 100% of the CPU?

Version 0.1

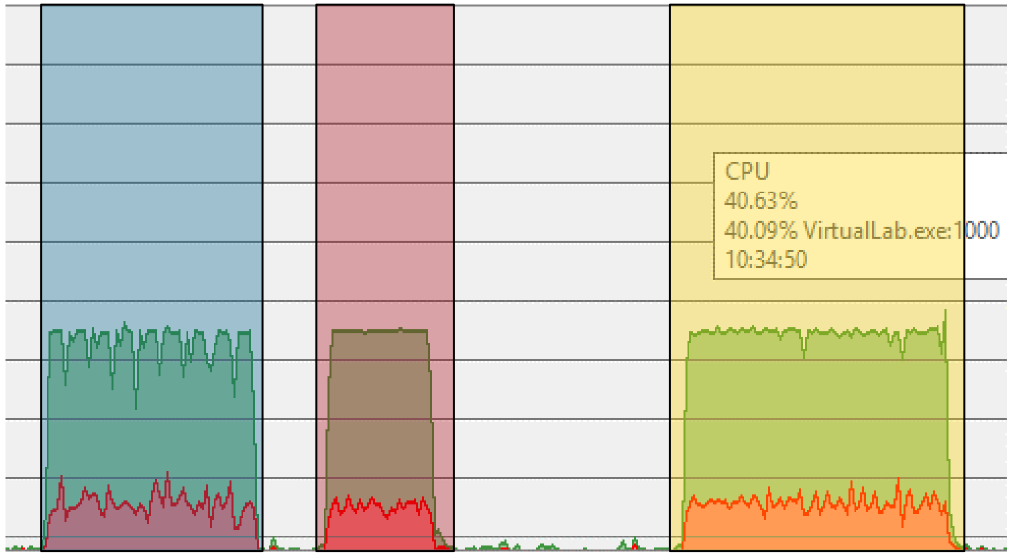
Introduction

- Modern CPUs always have more than one core.
- People often wonder why VirtualLab Fusion does not use all cores of a CPU with 100 %, even if it is the only running application.
- The following slides discuss possible causes.

Cause 1: Simultaneous Multithreading

What You Observe

- The overall CPU usage is around 50 %. If you look at the CPU usage of all cores with e.g. the Task Manager of Windows, you see that all cores run with around 50 %.



- Example for an algorithm which never uses more than 50 % of the CPU (here: the FMM of VirtualLab Fusion with ~500 orders [red], ~750 orders [yellow] and ~1000 orders [blue; canceled])

Reason

- Users often wonder why VirtualLab Fusion uses only 50 % of all cores visible in Windows Task Manager or Process Explorer.
- This is because of Simultaneous Multithreading (SMT), most commonly known under the Intel marketing term “hyper-threading”.
- In short: Windows recognizes for processors with SMT two logical cores for every physical core. However both logical cores are used simultaneously only in special cases, for example when one process is waiting for another process running on the other logical core.

Quote from Wikipedia

- According to Intel, the first hyper-threading implementation used only 5% more die area than the comparable non-hyperthreaded processor, but the performance was 15–30% better. Intel claims up to a 30% performance improvement compared with an otherwise identical, non-simultaneous multithreading Pentium 4. Tom's Hardware states: "In some cases a P4 running at 3.0 GHz with HT on can even beat a P4 running at 3.6 GHz with HT turned off." Intel also claims significant performance improvements with a hyper-threading-enabled Pentium 4 processor in some artificial-intelligence algorithms.
- Overall the performance history of hyper-threading was a mixed one in the beginning. As one commentary on high-performance computing from November 2002 notes:
- “Hyper-Threading” can improve the performance of some MPI applications, but not all. Depending on the cluster configuration and, most importantly, the nature of the application running on the cluster, performance gains can vary or even be negative. The next step is to use performance tools to understand what areas contribute to performance gains and what areas contribute to performance degradation.”
- As a result, performance improvements are very application-dependent; however, when running two programs that require full attention of the processor, it can actually seem like one or both of the programs slows down slightly when Hyper-Threading Technology is turned on.
- https://en.wikipedia.org/w/index.php?title=Hyper-threading&oldid=842570531#Performance_claims

Conclusion

- Thus it is quite usual that a program uses only half of the logical cores on a processor with SMT.
- Even if VirtualLab Fusion would use all logical processors instead of only 50% in average, it would run at most 30% faster, not the expected 100%.
- Assigning calculation tasks to actual CPU threads is done by the .Net ThreadPool programmed by Microsoft or by other external components like the Intel MKL. This assigning can be quite complex, as for example the thread pool has to take care that there are no more threads than CPU cores and there are not only worker threads but for example also I/O threads.
- Further technical details about tasks and threads can be found on Stack Overflow and in the MSDN Magazine.
- As a consequence, VirtualLab Fusion shows its CPU usage relative to the number of physical cores, not logical cores.

Cause 2: More than 64 Logical Cores

What You Observe

- On a multi-CPU machine with more than 64 logical cores you see in the Task Manager that not all logical cores are being used.

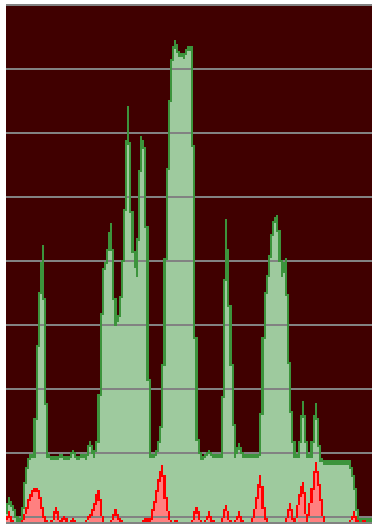
Reason & Solution

- If there are more than 64 logical cores, Windows groups them internally into processor groups for easier scheduling.
- By default, applications can only use one processor group. However, you can configure VirtualLab Fusion to use more than one processor. The corresponding config file is available upon request from service@lighttrans.com.
- But note that with this configuration file VirtualLab Fusion might run slower on machines with less than 64 logical cores.

Cause 3: “The Curse of Many Cores”

What You Observe

- Part of the algorithm runs in parallel. But most time only one core is used.
- Note that performance analysis tools like the Task Manager measure only the CPU usage averaged over some seconds or part of a second. Thus if parallelized and non-parallelized parts of the algorithm alternate quickly, you see an average CPU usage of e.g. 30 %.

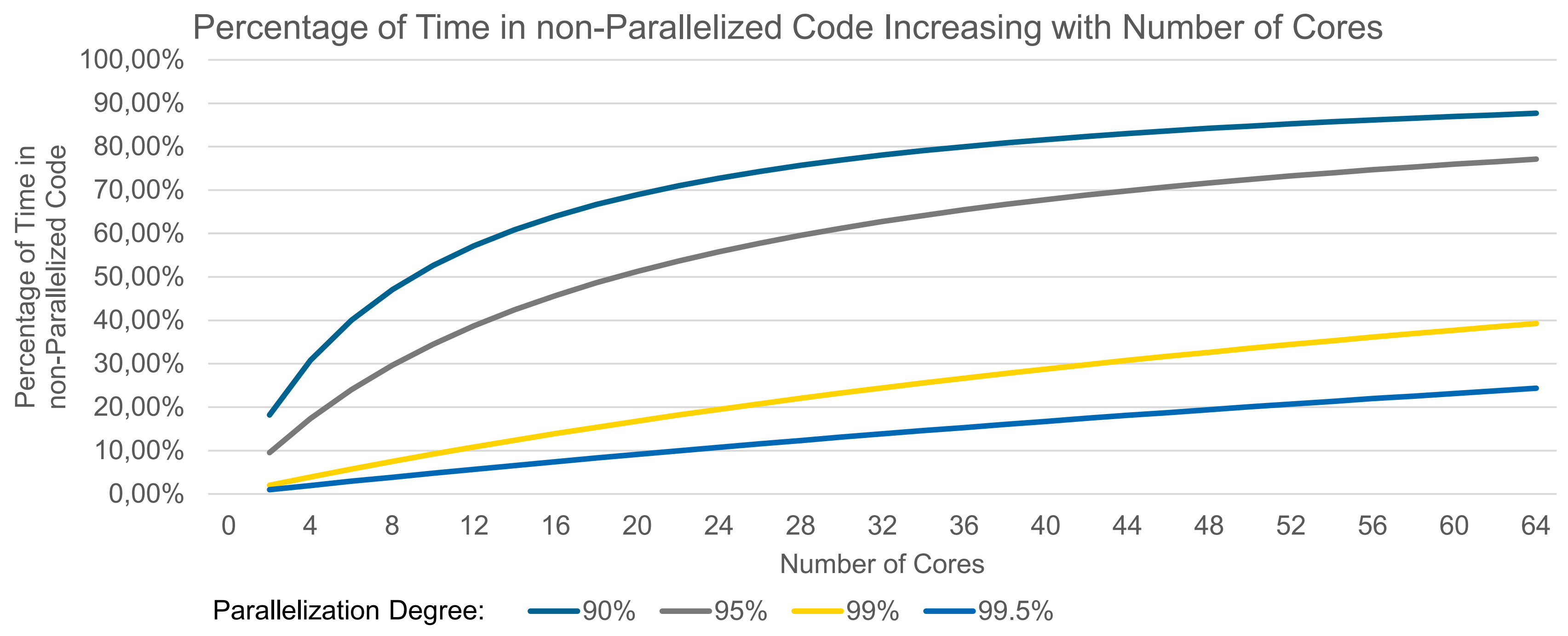


- Example for the (average) CPU usage of a Classical Field Tracing calculation in VirtualLab Fusion. Each horizontal line marks one logical core.

Reason

- Assume you have an algorithm which takes 100-time units on a CPU with only one core. The parallelization degree is 90 %, i. e. 90 % of the time the currently executed code could run in parallel when there are multiple cores.
- Now you switch to a machine with 10 cores. Then the parallelized code runs in 9-time units instead of 90-time units as on a single-core machine. The non-parallelized code still takes 10-time units. Thus the overall algorithm runs 10/19 = 53 % of the time non-parallel despite 90 % of the code are parallelized.
- This phenomenon becomes worse with more cores being available. The next slides show more examples.

Graphical Example



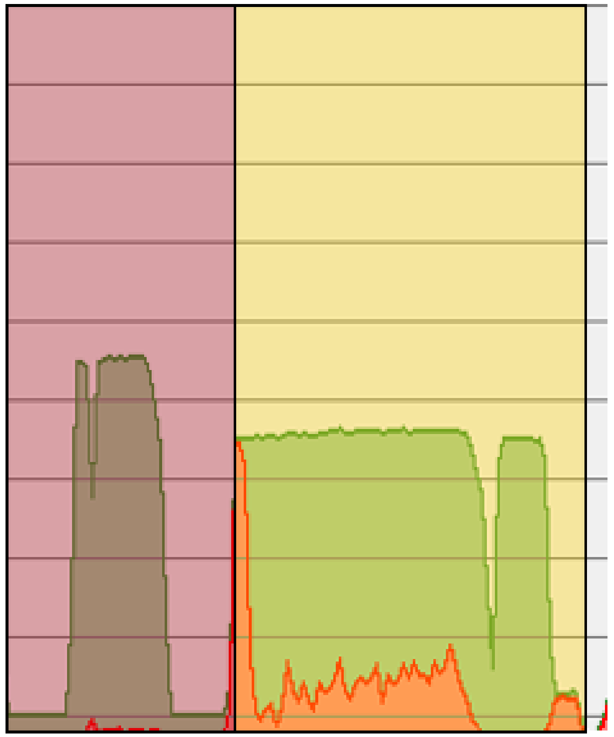
- The diagram shows that e.g. even for a parallelization degree of 99% an algorithm runs almost 40% of the time non-parallel on a machine with 64 (physical) cores.

How Can VirtualLab Fusion use the whole CPU?

Discussion

- As discussed in the previous FAQ, the reasons why VirtualLab Fusion doesn’t use 100 % of the CPU are manifold, often quite complex and cannot be resolved easily.
- For example it is not constructive to develop an own thread control just to use all logical cores. And there will always be code which cannot be parallelized, for example file operations.
- We understand that it can be irritating when VirtualLab Fusion doesn’t use the whole CPU, but in the end, it is not decisive how many CPU cores or RAM VirtualLab Fusion uses. **The only thing that counts is the time** it needs for calculations.
- “What [you] should really be concerned about is performance - how fast did [the algorithm] do the operation - not how much CPU did it use. If it's fast and doesn't use full CPU then it is an efficient algorithm. We strive for numerical operations that are fast and don't use much CPU as CPU is needed by the rest of the application.”
  (Trevor Misfeldt, Co-founder & CEO of CenterSpace Software, which develops the NMath library used by VirtualLab Fusion)
  …

Example



- Comparison of two FFT algorithms applied on the same data.
- The “yellow” LightTrans FFT uses in average a higher share of the CPU.
- But in the end the “red” MKL FFT takes much less time.

“Time to Result”

- = How long does a single simulation take?
- Depends on:
- Hardware
- How fast is the algorithm?
- Does it avoid unnecessary / too much calculations (like too high sampling)?
- How much RAM requires the algorithm? (RAM swapped on hard disc is slow)
- How good does the algorithm run in parallel?
- How fast converges the algorithm?
- …

“Time to Solution”

- = How long does it take till we have the final results?
- A program which does a simulation in 1 minute but needs 1000 simulations for a good result is worse than a program which needs 10 simulations which each take 10 minutes.
- “Time to solution” depends on:
- “Time to Result”
- How easy is the software to use?
- How often does it crash?
- How fast do you find a way to do what you want?
- How responsive is the GUI?
- Are the results correct and stable (i.e. the same with each simulation)?

What We Do to Improve Overall Performance

- As mentioned before, we cannot achieve that VirtualLab Fusion always uses the CPU completely.
- But with every new version we work hard to reduce the “time to result” of the various algorithms implemented in VirtualLab Fusion and to reduce the time till you have a solution for your problem.
- Examples:
  - Caching to avoid redundant calculations
  - Parallelization of more and more algorithms
  - Using up-to-date mathematics libraries which take advantage of the newest processor features
  - Completely new algorithms yielding physical correct results faster (2nd Generation Field Tracing)

What can I do to improve the Performance of VirtualLab Fusion?

General

- There are a few things you can do to improve the performance of VirtualLab Fusion:
  - When buying a new computer, do not only look for the number of cores but also for the turbo levels to get the maximum single core performance.
  - The Path for Temporary Files set in the Global Options dialog should point to a folder on an SSD.
  - Create a Windows swap file and make it sufficiently large. Details how to do this can be found in the VirtualLab Fusion Administrator’s Manual.
    Otherwise Windows might need to increase the swap file during a simulation.
  - Optimize the Multi-Core settings in the Global Options dialog (see next slide).

Global Options > Performance > Multi-Core

- Recommendations for the settings on this tab:
  - *Use Multiple Cores*: Check
  - *Number of Cores To Use*:
    - Set it to the number of logical cores *N* to use your CPU as much as possible.
  - *Use Multiple Core for Parameter Run Loop*: Check
    - As explained on a previous slide, the CPU usage can vary during a simulation. Thus if several simulations run parallel which each use only one thread, the CPU is used more evenly than if only one simulation runs at a specific time.
    - Only if you are usually using Parameter Runs with very fast iterations, we recommend to switch off this feature (which can also be done per Parameter Run).
    - Note that this setting does not have an effect for Grating Optical Setups. Such setups spend most of their simulation time to execute the FMM which itself is always running in parallel. Thus creating new threads to run the iterations of the Parameter Run in parallel is not constructive.