Fair-share policy on IAP compute servers

Dennis Arslan
Daniel Repp
Katsuya Tanaka
Maximilian Weißflog

16th August, 2022

Contents

1 General rules of conduct

Our IAP compute servers, such as Heron and Thales, are used by a large number of group members for computation-intensive tasks. However, since there is no automatic scheduling of resources, it is the responsibility of each user to ensure that their processes are not overusing resources or negatively interfering with the processes of other users. Misconduct can not only slow down the entire server, but may even lead to severe crashes. In this document, we want to make you aware of the most important issues and advise you on how to deal with them.

For each type of resource, there are a few critical aspects to consider:

CPU If there are more concurrent long-running threads than there are physical processors, the entire server may freeze. Each physical processor can work on only one thread at a time. Make sure that your programs spawn only a limited number of threads and leave enough processors for other users. On Thales, we also bind processes to specific processors, so that users can use them exclusively. This usually results in a small performance increase and processes of different users interfere less with each other.

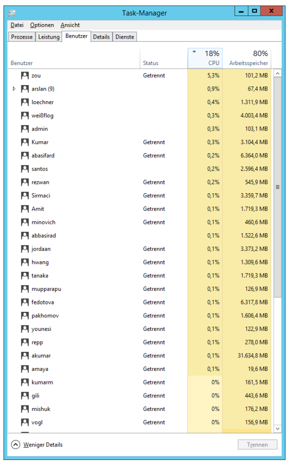
RAM Currently, it seems that we hardly ever come close to the capacity limit. For good practice, however, you should always release the RAM once you are done with your computations.

Local drives The operating system and most programs are installed on the C-drive. If this drive runs out of memory, various programs of every user on the server may crash and certain files can get corrupted. Simply do not store any private data on the C-drive, except when it is inevitable for the operation of your program. Unfortunately, most programs put their temporary folders on the C-drive by default. If your program produces large amounts of temporary data, you must move its temporary folder to another local drive. Every user must regularly remove their private data from all local drives and delete any residual temporary files which were not automatically deleted by the respective program.

Network bandwidth If your program constantly reads or writes large amounts of data, make sure that these files are on a local drive and not the network drive.

1.1 Logging out versus disconnecting from a remote desktop session

Problem Disconnecting from a remote desktop session does not log you out, so that you can continue your work from where you left. However, even if your programs are in an idle state, every logged-in user consumes a small amount of RAM and CPU-time. This quickly sums up with our large number of users. On Heron, we even observed up to 30% CPU utilization when all users were idle. Figure 1.1a, for example, shows all the disconnected users (Status: Getrennt) and their occupied resources.



Task-Manager

Datei Optionen Ansicht

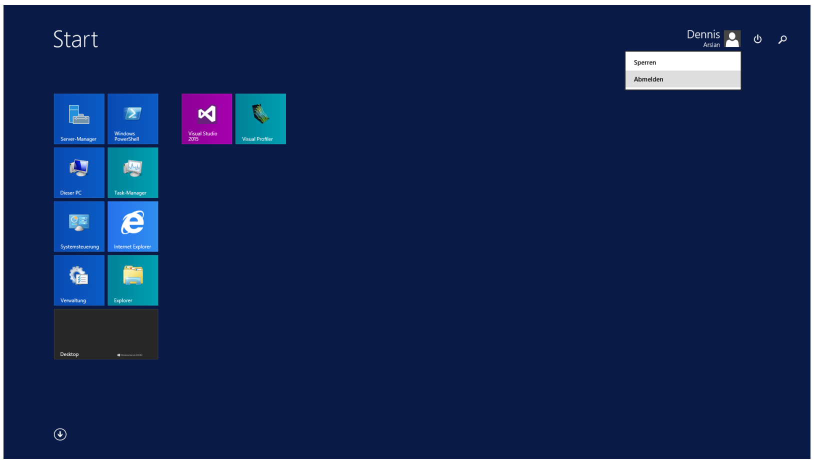
Prozesse Leistung Benutzer Details Dienste

Benutzer	Status	18% CPU	80% Arbeitsspeicher
zou	Getrennt	5,3%	101,2 MB
arslan (9)		0,9%	67,4 MB
loechner		0,4%	1.311,9 MB
weißflog		0,3%	4.003,4 MB
admin		0,3%	103,1 MB
Kumar	Getrennt	0,3%	3.104,4 MB
abasifard	Getrennt	0,2%	6.364,0 MB
santos		0,2%	2.596,4 MB
rezwan	Getrennt	0,2%	545,9 MB
Sirmaci	Getrennt	0,1%	3.359,7 MB
Amit	Getrennt	0,1%	1.719,3 MB
minovich	Getrennt	0,1%	460,6 MB
abbasirad		0,1%	1.522,6 MB
jordaan	Getrennt	0,1%	3.373,2 MB
hwang	Getrennt	0,1%	1.309,6 MB
tanaka	Getrennt	0,1%	1.719,3 MB
mupparapu	Getrennt	0,1%	126,9 MB
fedotova	Getrennt	0,1%	6.317,8 MB
pakhomov	Getrennt	0,1%	1.606,4 MB
younesi	Getrennt	0,1%	122,9 MB
repp	Getrennt	0,1%	278,0 MB
akumar	Getrennt	0,1%	31.634,8 MB
amaya	Getrennt	0,1%	19,6 MB
kumarm	Getrennt	0%	161,5 MB
gili	Getrennt	0%	443,6 MB
mishuk	Getrennt	0%	176,2 MB
vogl	Getrennt	0%	156,9 MB

Weniger Details

Trennen

(a) Task Manager, showing the state and resources of each user.



(b) Start screen, showing how to log out from the remote desktop session.

Figure 1.1: How to log out from a remote desktop session on Heron and Thales.

Solution

1. **Whenever you are done with your work on the server, completely log out from your user account in order to free up resources.**
   If you think about the server as if it was your own desktop PC, then logging out is effectively the same as shutting down and disconnecting is the same as turning off the screen.

Figure 1.1b shows how to log out from Heron and Thales. Go to "Windows Start", click on your user name, then click on "Abmelden".

1.2 Keeping the local drives clean

Problem When the C-drive runs out of memory and any program of any user tries to write to that drive, it will most likely crash. This usually results in aborted simulations, loss of data, or even corrupted system files. We encountered cases where Matlab would not start up again because of corrupted configuration files. The entire server can become unusable until the person who caused the error takes some action. Therefore, it is critical to keep enough free space on the C-drive. When the other local drives run out of memory, we may still encounter aborted simulations and loss of data, but at least no corrupted system files. Consequently, it is important to keep enough free space on all local drives.

Solution

1. **Do not store any big files on the C-drive.**
   Let's estimate the maximum space on Heron's C-drive you are theoretically allowed to occupy: There are ca. 70 user profiles, the C-drive has a total capacity of 446 GB, but the operating system and other programs already occupy around 136 GB. This leaves us with around 4.5 GB for every user. However, this is really the best-case scenario which would still lead to a full drive. In this situation, it may be acceptable if you occupied up to about 2 or 3 GB. However, we also believe that there is usually no reason for any user to occupy more than 1 GB. Note that these limits include both private and temporary data.

2. **Ideally, move all your temporary folders to another local drive.**
   On Heron and Thales, the other local drives are SSDs with more capacity than the C-drive. If you run programs which generate huge amounts of temporary data, you must make sure that the temporary folders are on the other local drives. Since all our drives are currently pretty full, you may keep certain temporary folders on the C-drive if you can guarantee that they will not take up much space.
3. **Regularly delete your private data or move it off the server to a permanent storage location.**
4. **Regularly delete any temporary files which were not automatically deleted by your programs.**

Depending on the programs you use, there can be several locations where your temporary files are hiding. In the following, we give some hints on where to look and what to do.

Start by checking your Windows user folder (`%USERPROFILE%`) for large files and folders, then dig down the folder tree. Also make sure that you look inside the app data folder (`%USERPROFILE%\AppData`), which may be hidden by default. It is likely that all your data on the C-drive is contained within these folders. You can quickly access these folders via the Windows Explorer (see fig. 1.2).

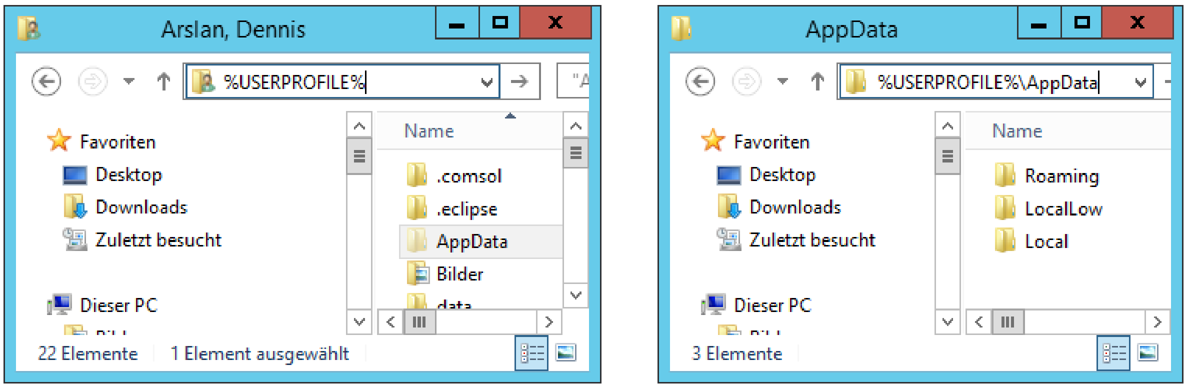
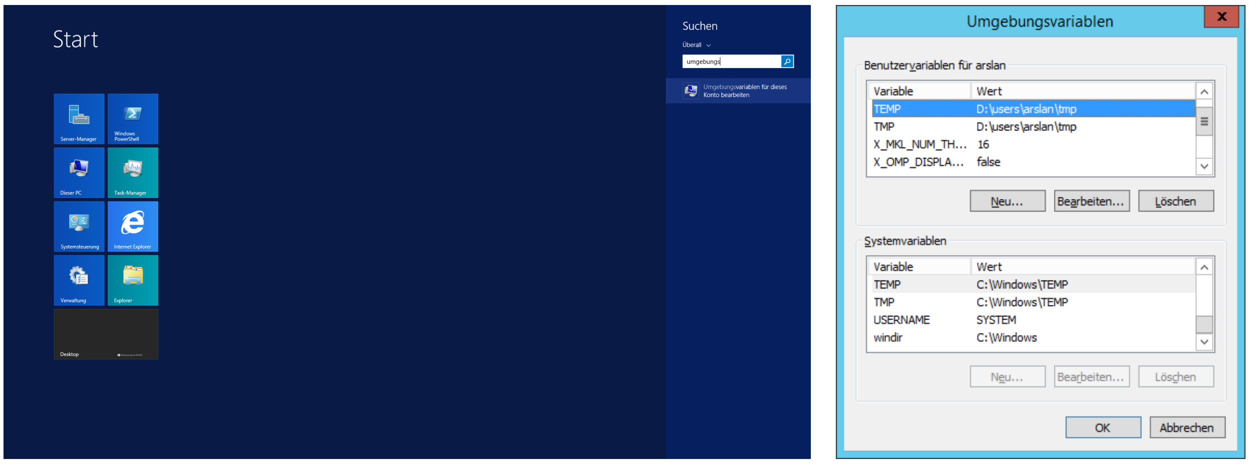


Figure 1.2: Folders on the C-drive which usually contain all user-specific data.

You can create a new temporary folder on another drive and change the environment variables `TEMP` and `TMP` such that they point to this new folder. Note that this is the default temporary folder which Windows offers to all programs. Be aware that some programs may not use this folder, or even create their own temporary folder somewhere else. Figure 1.3 shows how I moved the Windows temporary folder to `D:\users\arslan\tmp`.



Benutzervariablen für arslan

Variable	Wert
TEMP	D:\users\arslan\tmp
TMP	D:\users\arslan\tmp
X_MKL_NUM_TH...	16
X_OMP_DISPLA...	false

Systemvariablen

Variable	Wert
TEMP	C:\Windows\TEMP
TMP	C:\Windows\TEMP
USERNAME	SYSTEM
windir	C:\Windows

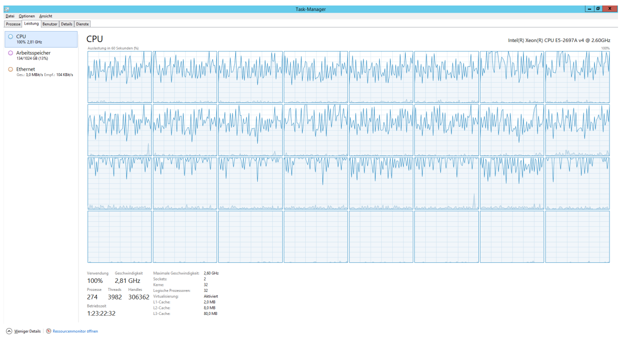
Figure 1.3: How to move the Windows temporary folder location by editing the TEMP and TMP environment variables.

1.3 Parallel computing

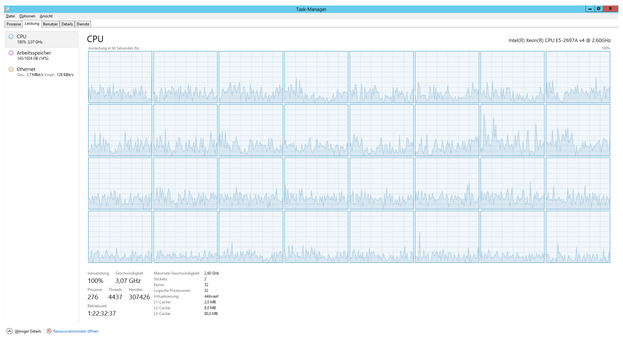
Problem Depending on how well your program can parallelize its work, there is usually an optimal number of processors for which your program will run the fastest (Amdahl's law[1]). In most cases, you will even observe a slow-down if you use too many processors. This means that these additional processors are not only useless or even detrimental for your program, but they are also not available for other users.

Solution Usually, you will have to determine the optimal number of processors via trial and error. Sometimes you can run a small part of your workload and compare the execution time for different numbers of processors. At other times, it can be sufficient to look at the Task Manager.

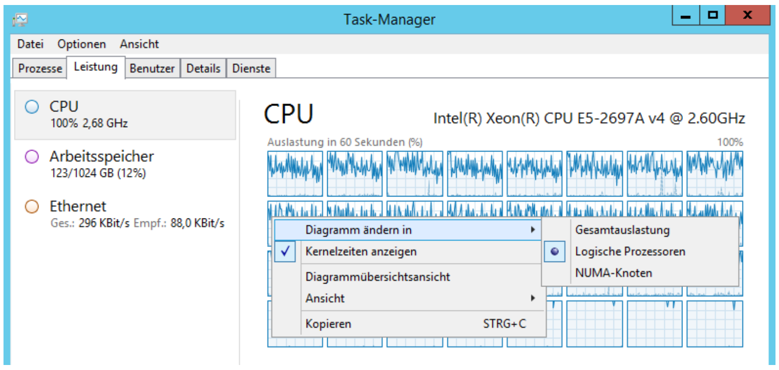
Figure 1.4c shows how you can configure the Windows Task manager to show the utilization of each processor and the kernel time. The kernel time tells you how much time the processor spends on low-level tasks of the operating system, as for example communicating with or waiting for other processors. If the kernel time is constantly high, it usually means that there is something wrong with your settings.



(a) Almost full utilization of all processors with almost no kernel time (dark blue).



(b) Full utilization of all processors, but roughly 20% of the time is wasted in the kernel (dark blue).



(c) How to configure the display.

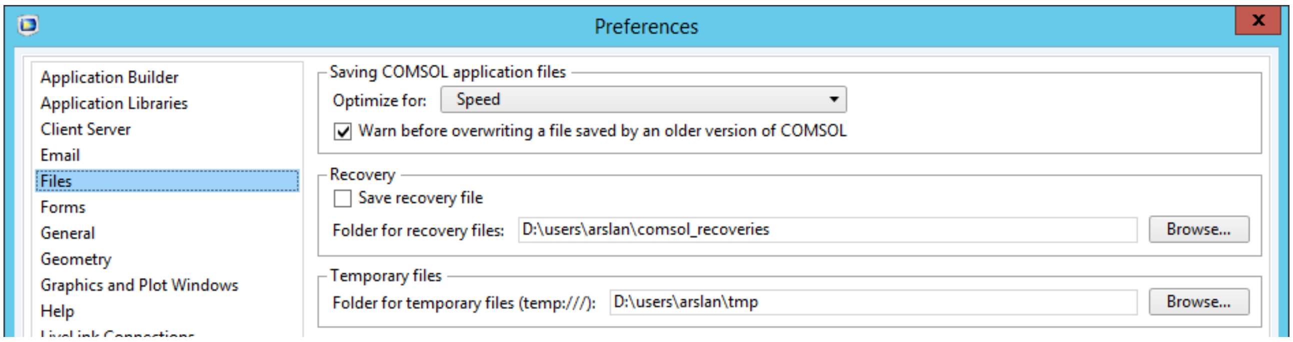
Figure 1.4: How to interpret the CPU performance tab in the Windows Task Manager.

[1]We suggest that you make yourself familiar with the basic concepts of parallel computing to better understand the implications of our suggestions. For example, see: `https://en.wikipedia.org/wiki/Parallel_computing#Background`

2 COMSOL

2.1 Folder settings

Figure 2.1 summarizes some optimal settings regarding the location of temporary files.



Preferences

Application Builder
Application Libraries
Client Server
Email
Files
Forms
General
Geometry
Graphics and Plot Windows
Help

Saving COMSOL application files
Optimize for: Speed
✓ Warn before overwriting a file saved by an older version of COMSOL

Recovery
☐ Save recovery file
Folder for recovery files: D:\users\arslan\comsol_recoveries Browse...

Temporary files
Folder for temporary files (temp:///): D:\users\arslan\tmp Browse...

Figure 2.1: COMSOl preferences "Files"

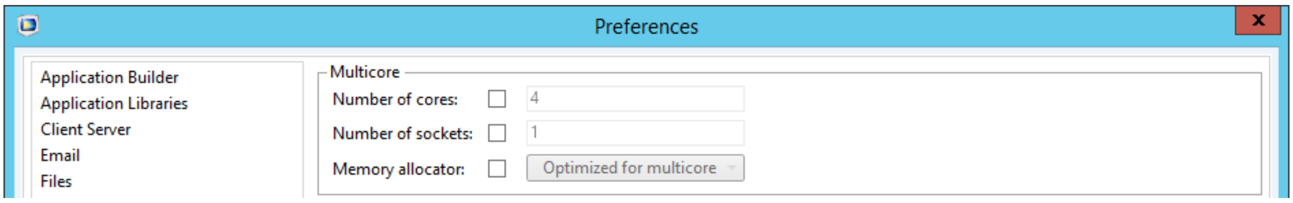
Save recovery file Disable recovery files to make your simulations complete faster and produce less junk data! These recovery files are hardly ever useful. Especially if your model is big, COMSOL will waste a lot of extra time and memory on writing these recovery files.

Folder for recovery files Create a new folder on a local drive (not on C:) and reference it here. Due to some reason, COMSOL temporarily writes simulation results into the recovery folder instead of the temporary folder.

Folder for temporary files Create a new folder on a local drive (not on C:) and reference it here. This folder usually contains files which are required by the graphic user interface or COMSOL server.

2.2 Limiting the number of processors

Figure 2.2 shows how to set the default number of processors used by each COMSOL process that you start. Unchecked boxes use the grayed-out default values. Check a box to provide your own settings. Note that these settings usually only apply after a restart of COMSOL.



Preferences

Application Builder
Application Libraries
Client Server
Email
Files

Multicore
Number of cores: ☐ 4
Number of sockets: ☐ 1
Memory allocator: ☐ Optimized for multicore

Figure 2.2: COMSOL preferences for "Multicore and Cluster Computing"

Number of cores This is the number of processors which will be used by each COMSOL process that you start. WARNING: While this is the normal way to set the number of processors, it will have no or even negative effects on Heron and Thales if you set the number to anything other than 4. See below for an alternative.

Number of sockets Keep it unchecked unless you know what you are doing.

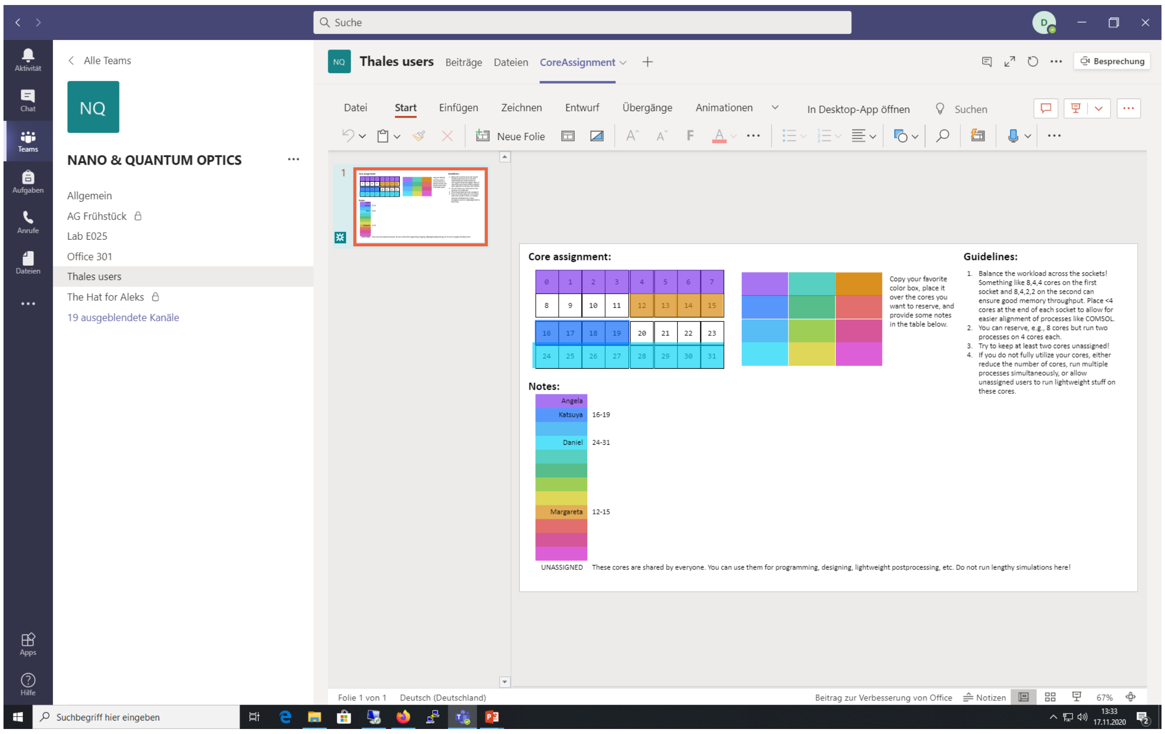
Memory allocator Keep it unchecked unless you know what you are doing.

The best way to control how many and which processors shall be used, is to start COMSOL with a few command line arguments. We prepared Windows batch scripts for Thales which not only limit the number of processors, but also set some critical environment variables and bind COMSOL to specific processors (see chapter 5). Please do not use the scripts on Heron, since the processor binding can have negative effects there. Furthermore, due to the large number of users on Heron, we also do not see the need to provide modified scripts for Heron.

3 Lumerical

3.1 Limiting the number of processors

If you are on Thales, first communicate with the other users which processors you want to use (see fig. 3.1 and chapter 5). You will have to bind Lumerical to these processors. On Heron, you should not use processor binding.



Core assignment:

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

Copy your favorite color box, place it over the cores you want to reserve, and provide some notes in the table below.

Guidelines:

1. Balance the workload across the sockets! Something like 8,4,4 cores on the first socket and 8,4,2,2 on the second can ensure good memory throughput. Place <4 cores at the end of each socket to allow for easier alignment of processes like COMSOL.
2. You can reserve, e.g., 8 cores but run two processes on 4 cores each.
3. Try to keep at least two cores unassigned!
4. If you do not fully utilize your cores, either reduce the number of cores, run multiple processes simultaneously, or allow unassigned users to run lightweight stuff on these cores.

Notes:

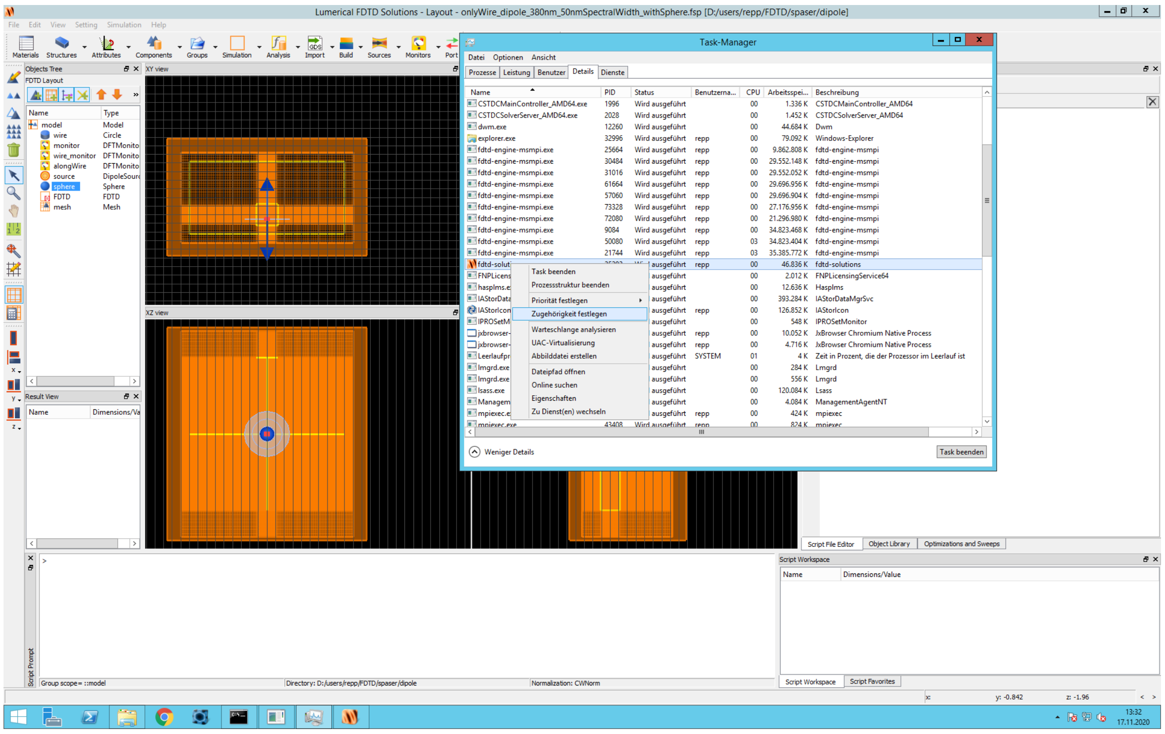
Angela	
Katsuya	16-19
Daniel	24-31
Margareta	12-15
UNASSIGNED	These cores are shared by everyone. You can use them for programming, designing, lightweight postprocessing, etc. Do not run lengthy simulations here!

Figure 3.1: MS Teams – Thales users. How to communicate exclusive processor reservation.

Processor binding via the Windows Task Manager For this method to work, you first have to start the graphic user interface of Lumerical, then open the Windows Task manager, go to

```
Windows Task Manager > Details > fdtd-solutions.exe > Zugehörigkeit festlegen
```

and select the cores you reserved (see fig. 3.2). This sets the so-called affinity mask which is managed by Windows. Every child process started by `fdtd-solutions.exe` will inherit the affinity mask and run on the same selected processors. The graphic user interface usually starts several `fdtd-engine.exe` processes which work in parallel on the simulation. Please verify that these processes are indeed bound to the selected cores.



Name	PID	Status	Benutzerna...	CPU	Arbeitsspei...	Beschreibung
CSTDCMainController_AMD64.exe	1996	Wird ausgeführt		00	1.336 K	CSTDCMainController_AMD64
CSTDCSolverServer_AMD64.exe	2028	Wird ausgeführt		00	1.452 K	CSTDCSolverServer_AMD64
dwm.exe	12260	Wird ausgeführt		00	44.684 K	Dwm
explorer.exe	32996	Wird ausgeführt	repp	00	79.092 K	Windows-Explorer
fdtd-engine-msmpi.exe	25664	Wird ausgeführt	repp	00	9.862.808 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	30484	Wird ausgeführt	repp	00	29.552.148 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	31016	Wird ausgeführt	repp	00	29.552.052 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	61664	Wird ausgeführt	repp	00	29.696.956 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	57060	Wird ausgeführt	repp	00	29.696.904 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	73328	Wird ausgeführt	repp	00	27.176.956 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	72080	Wird ausgeführt	repp	00	21.296.980 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	9084	Wird ausgeführt	repp	00	34.823.468 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	50080	Wird ausgeführt	repp	03	34.823.404 K	fdtd-engine-msmpi
fdtd-engine-msmpi.exe	21744	Wird ausgeführt	repp	03	35.385.772 K	fdtd-engine-msmpi
fdtd-soluti	[illegible]	ausgeführt	repp	00	46.836 K	fdtd-solutions
FNPLicens		ausgeführt		00	2.012 K	FNPLicensingService64
hasplms.e		ausgeführt		00	12.636 K	Hasplms
IAStorData		ausgeführt		00	393.284 K	IAStorDataMgrSvc
IAStorIcon		ausgeführt	repp	00	126.852 K	IAStorIcon
IPROSetM		ausgeführt		00	548 K	IPROSetMonitor
jxbrowser-		ausgeführt	repp	00	10.052 K	JxBrowser Chromium Native Process
jxbrowser-		ausgeführt	repp	00	4.716 K	JxBrowser Chromium Native Process
Leerlaufpr		ausgeführt	SYSTEM	01	4 K	Zeit in Prozent, die der Prozessor im Leerlauf ist
lmgrd.exe		ausgeführt		00	284 K	Lmgrd
lmgrd.exe		ausgeführt		00	556 K	Lmgrd
lsass.exe		ausgeführt		00	120.084 K	Lsass
Managem		ausgeführt		00	4.084 K	ManagementAgentNT
mpiexec.e		ausgeführt	repp	00	424 K	mpiexec
mpiexec.exe	43408	Wird ausgeführt	repp	00	824 K	mpiexec

Figure 3.2: How the set the affinity mask for Lumerical.

Processor binding via Windows batch scripts This method allows you to start any Lumerical process with a given affinity mask from the command line or a Windows batch script. The command takes the following form

```
start "" /affinity 0x30000000
    "C:\Program Files\Microsoft MPI\Bin\mpiexec.exe" -n 4
    "C:\Program Files\Lumerical\FDTD\bin\fdtd-engine-msmpi.exe" -t 1
    "yourModelFile.fsp"
```

where the line breaks are only for readability and each element has the following meaning:

start <title> /affinity <hex> <program> <parameter> The string `<title>` will be displayed in the title of the command prompt. Although the documentation says it is an optional argument, you usually have to provide an empty string `""` for the command to work correctly. The `/affinity <hex>` key provides the affinity mask as a hexadecimal number. A detailed description of this number is given below. The sequence `<program> <parameter>` specifies the program which shall be started with the given affinity mask and the parameters which shall be passed to the program during startup. In the example above, `<program>` represents the string containing `mpiexec.exe` and `<parameter>` the remainder.

mpiexec <mpiOptions> <solver> <solverOptions> The `mpiexec.exe` process manages the parallel execution of a particular Lumerical `solver`. In the example above, `<mpiOptions>` represents `-n 4`, `<solver>` the string containing `fdtd-engine-msmpi.exe` and `<solverOptions>` the remainder. The option `-n 4` means that up to 4 solver processes are allowed to run in parallel.

fdtd-engine <options> <filename> The `fdtd-engine-msmpi.exe` performs the simulation. In the example above, `<options>` represents `-t 1` and `<filename>` the model file `"yourModelFile.fsp"`. The option `-t 1` implies that each solver process runs on one processor.

Note that you may simply omit the `/affinity <hex>` key to avoid the processor binding. For a full list of all available options, please refer to the respective documentations:

- start – Starts a separate Command Prompt window to run a specified program or command.
- mpiexec – Starts one or more Message Passing Interface (MPI) applications on an HPC cluster.
- fdtd-engine – Running simulations via the MPI.

Determining the hexadecimal value of the affinity mask The affinity mask is based on a binary representation of the selected processors. In the following example, we assume that the CPU has 32 processors with identification numbers (IDs) ranging from 0 to 31. We want to select the processors with ID 28 and 29:

```
Processor ID    31      24   23      16   15       8   7        0
                |        |   |        |   |        |   |        |
Binary mask     0011  0000   0000  0000   0000  0000   0000  0000
                ----  ----   ----  ----   ----  ----   ----  ----
Hexadecimal        3     0      0     0      0     0      0     0  =  0x30000000
```

You may omit the leading zeros. For example, selecting processors 8 to 11 yields:

```
Processor ID    31      24   23      16   15       8   7        0
                |        |   |        |   |        |   |        |
Binary mask     0000  0000   0000  0000   0000  1111   0000  0000
                ----  ----   ----  ----   ----  ----   ----  ----
Hexadecimal        0     0      0     0      0     f      0     0  =  0xf00
```

Figure 3.3 shows how you can use the Windows calculator tool to quickly find the hexadecimal value of the affinity mask by hand. Simply switch to the programmer mode and click on the bits in the binary representation which you want to set (directly below the output field).

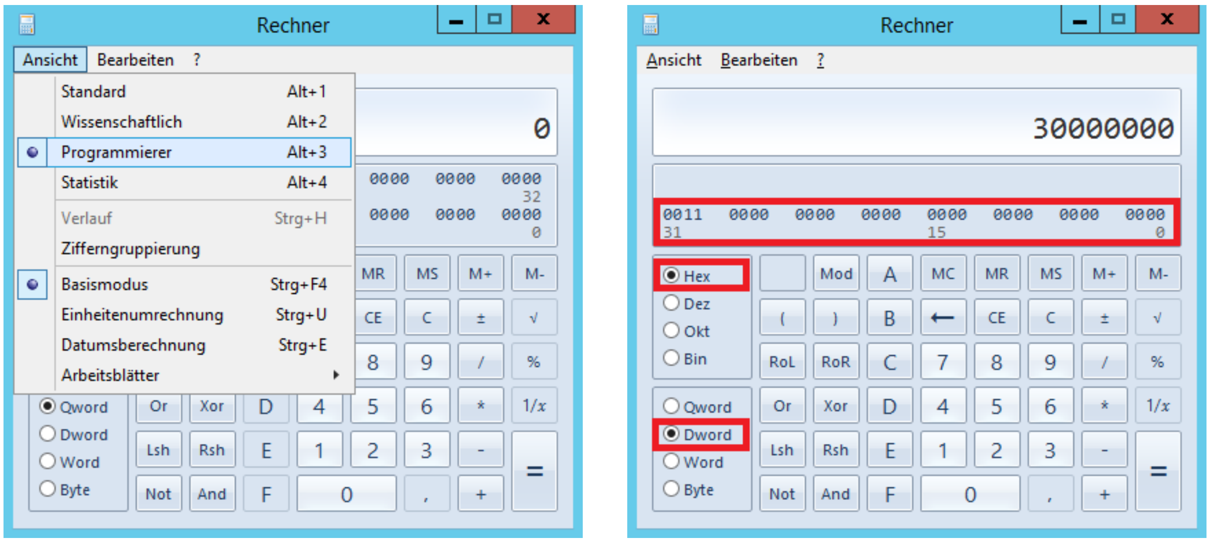


Figure 3.3: Using the Windows calculator to quickly find the affinity mask by hand.

4 MATLAB

4.1 Limiting the number of processors

In MATLAB, you can set the number of computation threads at runtime via the function:

```
maxNumCompThreads(N)
```

See `https://de.mathworks.com/help/matlab/ref/maxnumcompthreads.html` for more information.

It is also possible to bind all MATLAB computation threads to specific processors at runtime by setting the affinity mask via the .NET interface. See chapter 5 for more information.

5 Thales – Advanced topics

On Thales, we agreed on binding all processes to specific processors. This often not only speeds up simulations, but also prevents users from interfering with each other. If you are working on Thales or want learn more about some technical details not covered in this document, join the Thales users channel on Microsoft Teams. There, you can also find some further information and utility scripts in the “Files” section.

List of Figures