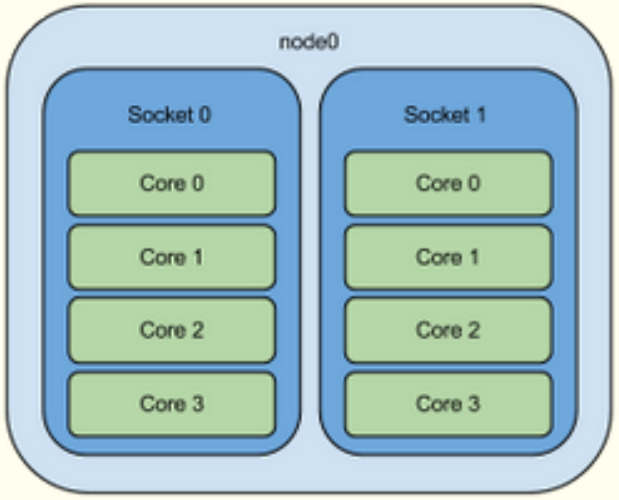


CORE SCHEDULING

8 8

Download page as PDF

An investigation

Introduction
The Windows CPU or CORE schedular divides the running tasks over its available cores. It's a complex internal process on which in the past you had no control. On a multi-core PC every launched program gets access to all available cores.

Go to the task manager, click right on a running program and click **Affinity** and on a typical quad PC with hyper-threading (see upper-left picture) you will notice the CPU schedular is allowed to use all cores as demonstrated in the picture on your right.

For more information on CPU scheduling check the WIKI.

For chess programmers feeding our 4/8/16/32 core PC's with matches playing thousands of games to test if a change will produce an ELO improvement the question arises if we leave the matches into the hands of the MicroSoft Windows CPU schedular or if we take control ourselves because as one can see from the above picture Windows allows the modification of the **Affinity** of each running program. And if we take control, can a chess program benefit from that? This is what the investigation is about.

But first things first, what is **Affinity**? With Affinity one can **pin** a running match to a specific core meaning that the 2 programs playing only run in that particular core and (most important) no other running program in the background can make use of that core. End of theory, let's go to practice.

On a quad PC we start a 4 matches between Andscacs 64-bit vs Andscacs 32-bit leaving Windows in control over the core divisions. Then we start the match again and run a batchfile immediately thereafter that changes the affinity pinning each match to the appropiate core. Then we compare the results. We do the same with Stockfish 4, 64-bit vs the 32-bit edition on an 8 core PC. Below you can follow the results. On a quad with hyperer-threading the affinity settings now will look:

Core one

Core two

Core three

Match [Andscacs 64bit vs Andscacs 32bit] [4 cores]

CPU	Games	Time	Score	ELO	Depth	Depth
Normal	2400	**40/15**	63.9%	+99	16.30	14.80
Affinity	2400	**40/15**	63.5%	+97	16.48	15.03

Match [Stochfish 4 64bit vs Stocfish 4 32bit] [8 cores]

CPU	Games	Time	Score	ELO	Depth	Depth
Normal	4800	**40/15**	57.6%	+52	17.34	16.61
Affinity	4800	**40/15**	58.6%	+60	17.82	17.02

Looks okay but what springs in mind is that both engines seem to profit from the affinity settings resulting in a higher average depth.

One full percent more but with (just) 4800 games that's not conclusive, a sign at the wall at best. But the increase of the average depth of both is remarkable, about half a ply more for the 64-bit version and +0.4 for the 32-bit version.

Match [Andscacs 64bit vs Andscacs 32bit] [4 cores]

CPU	Games	Time	Score	ELO	Depth	Depth
Normal	2400	**40/30**	63.6%	+98	18.36	17.06
Affinity	2400	**40/30**	63.2%	+95	18.50	17.23

Match [Stochfish 4 64bit vs Stocfish 4 32bit] [8 cores]

CPU	Games	Time	Score	ELO	Depth	Depth
Normal	4800	**40/30**	57.6%	+52	19.36	18.49
Affinity	4800	**40/30**	57.0%	+48	19.59	18.84

After this first acquaintance with affinity which shows an increase in search depth but (so far) is non comclusive if affinity produces a more accurate way of testing we increase the number of games to 16,000 and only play 40/5+0.1 games on the 8 core PC. Programs that will be tested: Andscacs, Stockfish 4 Komodo 5 in order to find evidence if testruns with affinity (**table 1**) produce more accurate results.

In the right column (**table 2**) the results are shown if we pitch the 64-bit versions against each other, with and without affinity and see how that will turn out.

TABLE 1

Match [Andscacs 64bit vs Andscacs 32bit] [8 cores]

CPU	Games	Time	Score	ELO	Depth	Depth
Normal	16,000	40/5+0.1	65.3%	+111	13.69	11.26
Affinity	16,000	40/5+0.1	65.5%	+112	14.12	11.71

Match [Komodo 5 64bit vs Komodo 5 32bit] [8 cores]

CPU	Games	Time	Score	ELO	Depth	Depth
Normal	16,000	40/5+0.1	73.8%	+180	13.10	11.57
Affinity	16,000	40/5+0.1	73.2%	+174	13.27	11.77

Match [Stochfish 4 64bit vs Stocfish 4 32bit] [8 cores]

CPU	Games	Time	Score	ELO	Depth	Depth

TABLE 2

Match [Andscacs 64bit] [affinity vs normal] [8 cores]

Games	Time	Score	LOS	ELO	Depth	Depth
16,000	40/5+0.1	53.9%	100%	+27	13.57	12.99

Oh la la .... an astounding result

Match [Komodo 5 64bit] [affinity vs normal] [8 cores]

Games	Time	Score	LOS	ELO	Depth	Depth
16,000	40/5+0.1	51.8%	100%	+10	13.20	13.09

Match [Stockfish 4 64bit] [affinity vs normal] [8 cores]

Games	Time	Score	LOS	ELO	Depth	Depth

Normal	16,000	40/5+0.1	59.3%	+66	15.95	15.13
Affinity	16,000	40/5+0.1	59.1%	+64	16.53	15.75

There is no good reason to question the accuracy of testing with versus without affinity. So far the results are rather inconclusive, +0.2% for Andscacs, -0.6% for Komodo, -0.2% for Stockfish, all within the error margin. What remains is the faster search quite convincingly.

16,000	40/5+0.1	53.6%	100%	+24	17.61	17.23

And another amazing ELO gain

The question is if the setup of these matches are fair in the sense the [affinity] version is perhaps hurting the performance of the version without affinity. This is a crucial point that needs an answer before jumping into final conclusions.

Another observation

Using affinity (for chess matches) only make sense when more than one core is use, that (**say**) on a **quad** PC at least 2 matches are running. This is demonstrated by taking a position that ProDeo solves in 0:43. Then we run ProDeo again with affinity and we get the same solution time. We expand the test by running this position on 2, 3 and 4 cores simultaneously without affinity and thereafter with affinity. This result into the following solution time table.

TABLE 3

Affinity	core-1	core-2	core-3	core-4
off	0:43	0:46	0:51	0:52
on	0:43	0:43	0:49	0:49

CONCLUSIONS

1. From the above paragraph (table 3) we learn there is no gain tuning the affinity when only one core is active.

2. Tuning the cores with affinity starts to make sense running multiple matches spreading over its available cores resulting in a faster search, see table 1 & 2.

3. Although not part of this investigation tuning the cores with affinity it is logical to assume SMP programs will benefit from affinity.

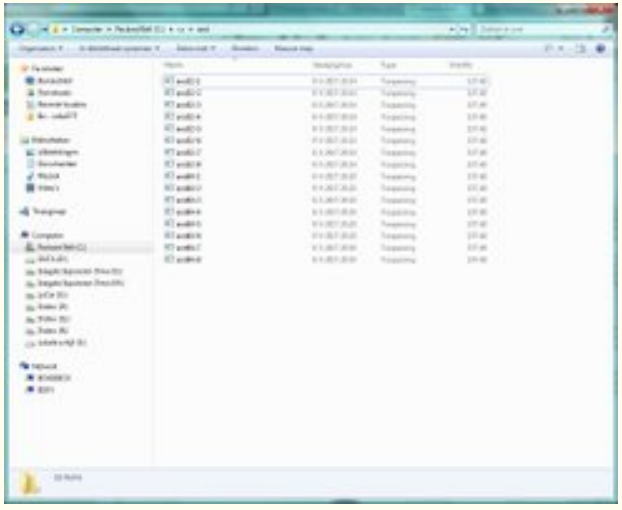
4. While the main purpose of this (2½ weeks) study was to investigate if testing with affinity would possibly a more accurate way this can't be derived from the results in table 1.

SETUP

Matches with Affinity

It's best of course if you add cpu scheduling code to your engine, you profit from a somewhat deeper search and you can keep your normal way of running matches. Open sources who already make use of cpu scheduling are (as far as I know)Texel and Stockfish.

Otherwise you will need to setup your matches somewhat different because I haven't found a way (yet) to modify the affinity settings when you are using **cutechess-cli** in combination with the **-concurrency** option. So here is how it currently runs. As example the **Andscacs** testruns for 4-core and 8 core PC's.



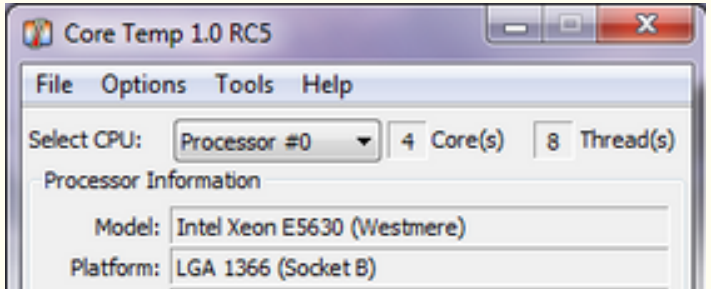
Steps:

1. Create a folder and make 8 copies of 32-bit and 64-bit version, for example as in the picture on your left. Click to enlarge.

2. Then create a batchfile to start the 4 (or 8) matches with cutechess-cli.

3. Then depending on if you are running 4 or 8 matches run the below corresponding batchfile.

For ducumentation check for intance Stackoverflow.




```
PowerShell "$Process = Get-Process and32-1; $Process.ProcessorAffinity=3"
PowerShell "$Process = Get-Process and64-2; $Process.ProcessorAffinity=3"

PowerShell "$Process = Get-Process and32-2; $Process.ProcessorAffinity=12"
PowerShell "$Process = Get-Process and64-2; $Process.ProcessorAffinity=12"
```

Pow erShell "$Process = Get-Process and32-3; $Process.ProcessorAffinity=48"
Pow erShell "$Process = Get-Process and64-3; $Process.ProcessorAffinity=48"

Pow erShell "$Process = Get-Process and32-4; $Process.ProcessorAffinity=192"
Pow erShell "$Process = Get-Process and64-4; $Process.ProcessorAffinity=192"

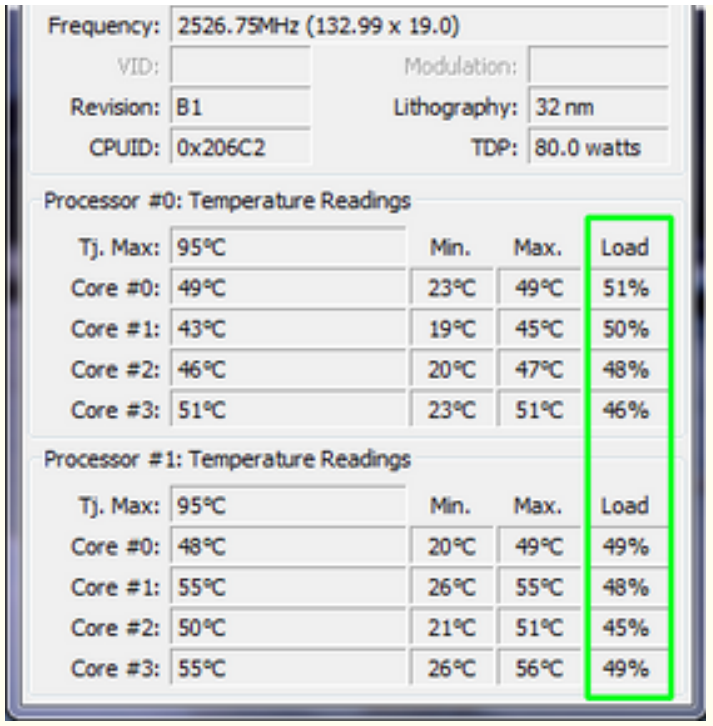
And for 8 cores continue as follows:

Pow erShell "$Process = Get-Process and32-5; $Process.ProcessorAffinity=768"
Pow erShell "$Process = Get-Process and64-5; $Process.ProcessorAffinity=768"

Pow erShell "$Process = Get-Process and32-6; $Process.ProcessorAffinity=3072"
Pow erShell "$Process = Get-Process and64-6; $Process.ProcessorAffinity=3072"

Pow erShell "$Process = Get-Process and32-7; $Process.ProcessorAffinity=12288"
Pow erShell "$Process = Get-Process and64-7; $Process.ProcessorAffinity=12288"

Pow erShell "$Process = Get-Process and32-8; $Process.ProcessorAffinity=49152"
Pow erShell "$Process = Get-Process and64-8; $Process.ProcessorAffinity=49152"



And each match is now pinned to one core, check it out via the control panel.

Another useful tool is CoreTemp to check if everything runs well. Note the CPU loading percentages and how it almost perfectly surrounds the 50%. Compare that with the normal MicroSoft Windows CPU loading percentages constantly interfering in each others cores and it becomes clear why affinity performs better.

A final observation

and warning

As far as I know not many engines (and especially SMP engines) have discovered the benefit of adding affinity code but (unfortunately) the possibility that one can tune the cores behaviour also means a mal intended programmer can misuse the facility as the following 2 matches will clearly demonstrate.

Stockfish 4 is rated **3179** at CCRL and **Komodo 5** is **3144** CRRL rated and we pitch them on a quad in 4 simultaneus mathches of each 250 games so 1000 games in total. The first match we manipulate the affinity in favor of Komodo 5, the second run we manipulate the affinity in favor of Stockfish 4. Watch and shiver.

Match	Affinity	Games	Time	Score	ELO	S4-Depth	K5-Depth

Stochfish 4 vs Komodo 5	favor Komodo	1000	40/15	36.0%	-100	16.50	14.27
Stochfish 4 vs Komodo 5	favor Stockfish	1000	40/15	68.2%	+133	17.86	12.71

As we can see the lower rated Komodo 5 crushes the 35 elo higher rated Stockfish 4 and suddenly is 100 elo stronger, a difference of 135 elo points. And when we favor Stockfish 4 in the second run with the affinity settings the elo gap widens from +35 to +133. Devestating and false results.

As it is not my intention to publish how it is done it's pretty obvious (note the dephts) for the observant reader, point being is, it can be done, and should not be done, ever.
