ChessBase Engine Performance Tests

Mark Kaprielian

2025-03-30

I. Introduction

This document contains some testing I have performed to try to answer some questions I've often had about the CB Engine such as:

- Does the number of lines Displayed impact the analysis
- How much does using more CPUs appear to improve the results or speed of reaching the same results.
- How repeatable is the analysis for a given position

My hope is that some insight will be gained but not necessarily firm conclusions without much more rigorous testing

The tests and results are presented in the order to help best understand them and some of the tests reference previous tests.

Probably the most interesting and useful results of this document is the last section, **"Best Lines"**.

II. The general test conditions

- CB 18
- Game Analyzed: One move played: 1 d4
- Hash table not cleared unless explicitly mentioned it has:
  - According the CB Help: *Kibitzers use "hash tables" to store positions that have already been examined, in case they turn up again in the search tree. This speed things up tremendously, especially in the endgame.*
- Task Manager: CPU performance window running, No browser windows open or other user activity going on.
- NOTE:
  - Normally we observe the depth grow from 10 to the high 40s over time when analyzing a middle game position
  - During these tests the Depth started out at about 35 and stayed in that range. Apparently, the test conditions being used are not appropriate for testing depth over time. I suspect the complexity of the position would have a large impact and thus make it difficult to profile the behavior.

III. kN/s vs CPUs and Lines viewed

Three test runs were made:

- Pass 1: To establish a base line of measurements, 8 lines being displayed
- Pass 2: Same as Pass 1 to check repeatability
- Pass 3: Change of lines being viewed from 8 to 4 to see what impact it may have

The results appear to be repeatable

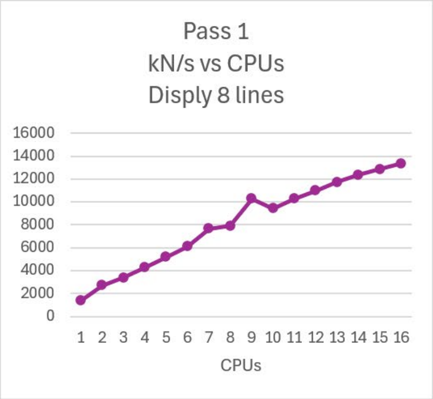
There is an anomaly in Pass 1 where a jump in performance occurred then returned to the trend when more CPUs were added. Pass 2 did not show this anomaly.

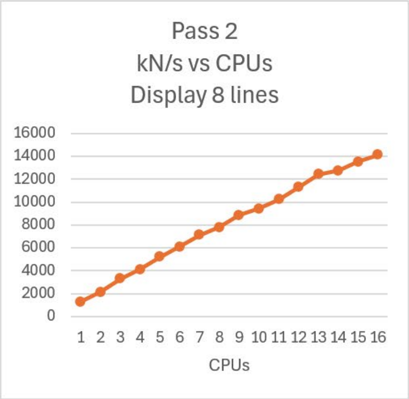
Pass 3 reduced the number of lines displayed to see if there was any general impact. The anomaly was seen again but with a higher number of CPUs.

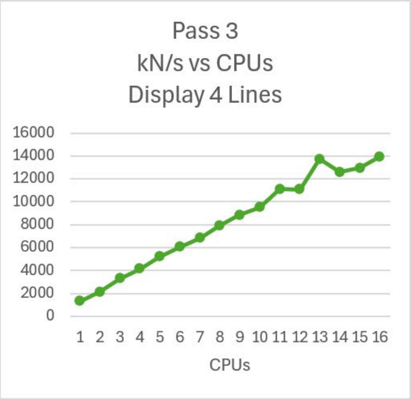
I the anomaly was due to some background activity of the OS I would expect that performance would go down when such activity occurred, not up! There are a lot of things in the mix:

- The OS
- CB handling of UCI (Universal Chess Interface) resource management
- Stockfish resource management

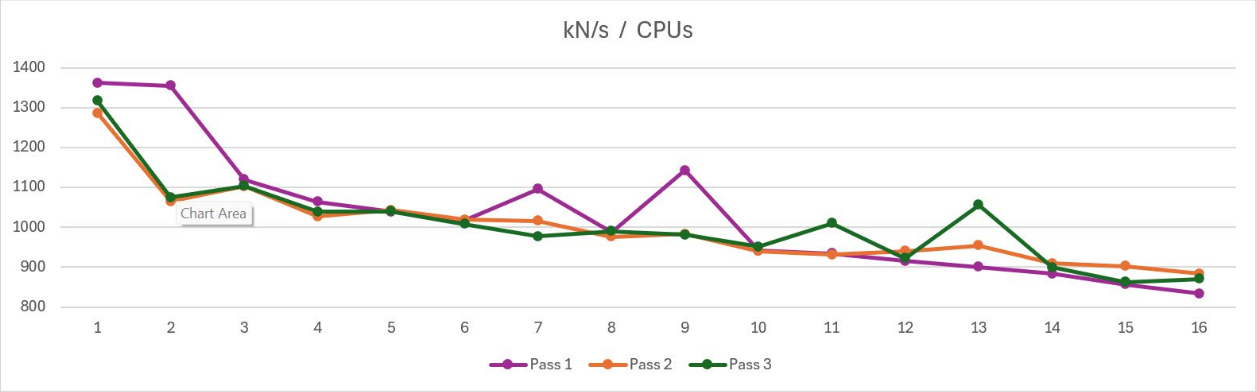
CB support had told me in one of my cases that CPU and memory management depends on how the engine creators manage the resources from within the UCI.

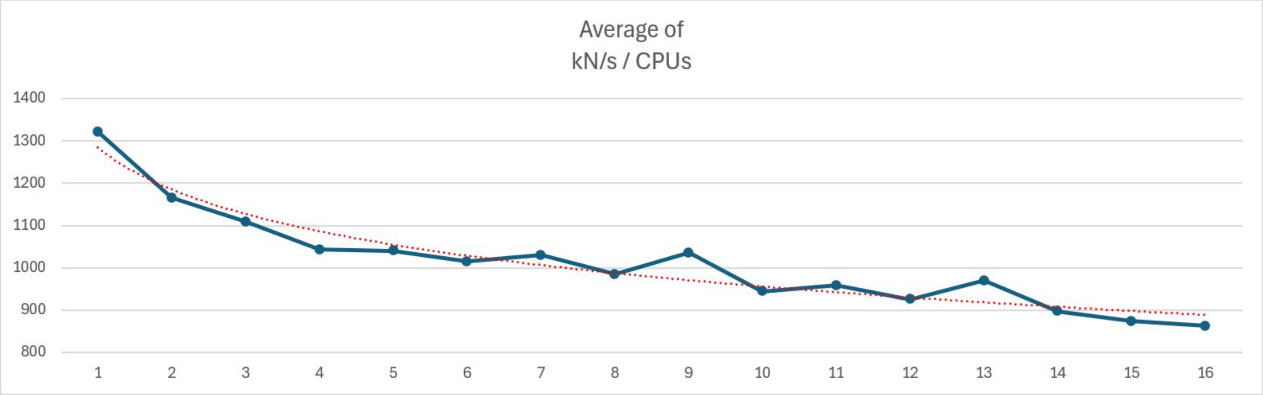






Since the kN/s trends in a straight line as you add CPUs, the following graphs show the relationship.



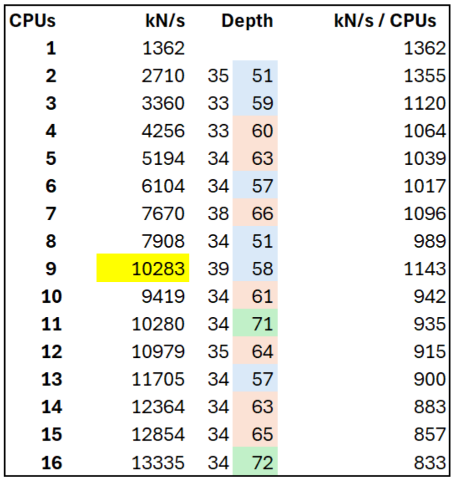


The top graph shows how closely the performance tracked across the three Passes. The spikes at 9 and 13 CPUs are the anomalies noted previously.

The bottom graph is an average of the three Passes

What I find interesting is that the more CPUs you add, the performance drops off. Not by a lot but there seems to be a clear trend. One theory could be that it's going out so much deeper that it has more work to do and thus it slows down. Here is the data from Pass 1 showing Depth.

Notice that the depth does not appear to increase in relation to the number of CPUs going up while performance is clearly going down. Interesting also I think is that the current ply being analyzed for the depth remains about 34 after 1 minute despite the number of CPUs



CPUs	kN/s	Depth		kN/s / CPUs
1	1362			1362
2	2710	35	51	1355
3	3360	33	59	1120
4	4256	33	60	1064
5	5194	34	63	1039
6	6104	34	57	1017
7	7670	38	66	1096
8	7908	34	51	989
9	10283	39	58	1143
10	9419	34	61	942
11	10280	34	71	935
12	10979	35	64	915
13	11705	34	57	900
14	12364	34	63	883
15	12854	34	65	857
16	13335	34	72	833

IV. How you change CPU count appears to impact results

12 CPUs is my preferred sweet spot for my PC where I can still use my PC to do a lot of other things at the same time, I'm letting the engine sit on a position. I based this on fiddling with the number of CPUs after I determined that my optimal Hash table size is 4096. With this size Hash table, the engine loads and unloads in under a second.

To get the measurements I made a video of the lines displayed then played it back in paused mode to get the performance at ten second intervals.

Having set the cores to 12 the performance at one minute should match the three Passes

The previous test passes show at 1 minute the following:

- Pass 1: 10979
- Pass 2: 11280
- Pass 3: 11072

In a different set of 4 test runs, at 12 cores at 1 minute we see:

- New Pass 1: 13994
- New Pass 2: 14003
- New Pass 3: 12452
- New Pass 4: 13882

There was a difference in how the first three Passes were run compared to these later runs.

For the first three Passes:

- The engine was running from the previous CPU test
- The CPU button was pressed which stopped the engine
- The number of CPUs to run was increased by one
- The OK button was pressed and as soon as the display window opened the timer for a minute started.

For the later test runs, the number of CPUs never changed.

- The engine was running then paused
- The video recording was started
- The engine was un-paused, which resets the calculation
- The video is stopped when 5 minutes have been recorded.

So, better performance seems to be due to not having changed the CPUs in the later tests as opposed to the early tests when changing the CPU was required.

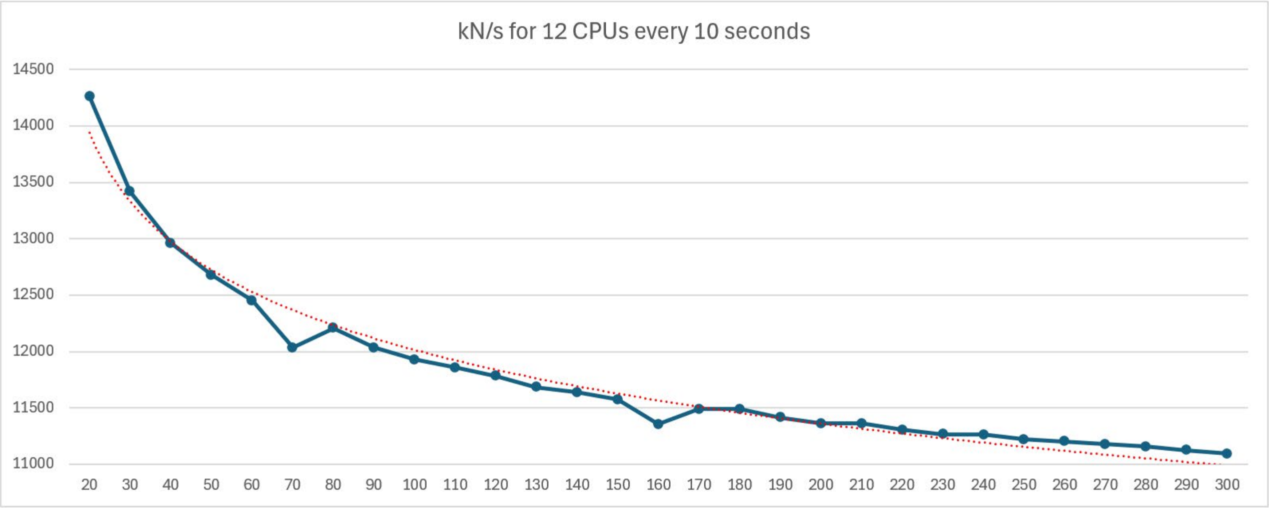
Further tests could be done to verify if this was the cause of not making use of the running engines ability to change CPUs but actually changing the default.

V. Performance as you let the engine keep running

Here’s a visual of what happens to performance as the longer you let the engine run.

In the graph below, performance settles out to about 11000 at five minutes.

As mentioned earlier, due to the test condition of having the position of 1 d4 being analyzed, no insight on how depth was of significance as the depth starts and stays near 34 right from the start of every test.



VI. Best Lines – Are we getting the same results every time?

Four Passes were made, in the following order:

- Pass 1: 12 CPUs, 5 minutes, Hash untouched, 8 lines displayed
- Pass 2: 12 CPUs, 5 minutes, Hash untouched, 8 lines displayed
- Pass 3: 12 CPUs, 5 minutes, Hash cleared, 8 lines displayed
- Pass 4: 12 CPUs, 5 minutes, Hash untouched, 4 lines displayed

Pass 1 and Pass 2 were the same scenario and we would hope, provide the same results

Pass 3 had the Hash table reset. As mentioned earlier in the test conditions section, the hash should speed up calculations as it stores previous results. Since we've just tested the same move twice, in theory then, we should get faster / deeper results.

Pass 4 changes the lines displayed to see if it appears to have any impact

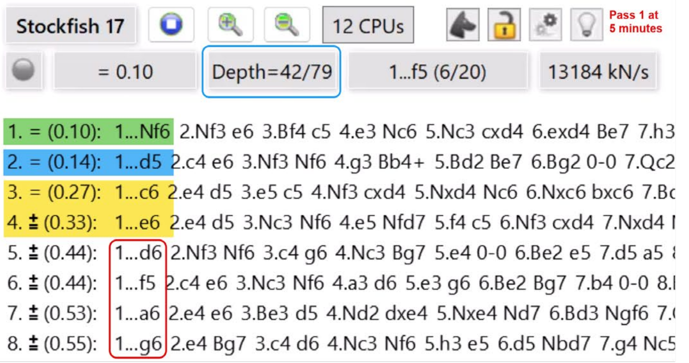
The answer appears to be:

- You get about the same results and the lines in the display don't matter if you wait 5 minutes.

Notice the Depth changes. Remember Pass 2 had the benefit of Pass 1 due to the Hash table. Pass 3 had the Hash table cleared so a clean run, actually the cleanest run here, but Pass 4 had the benefit of Pass 3.

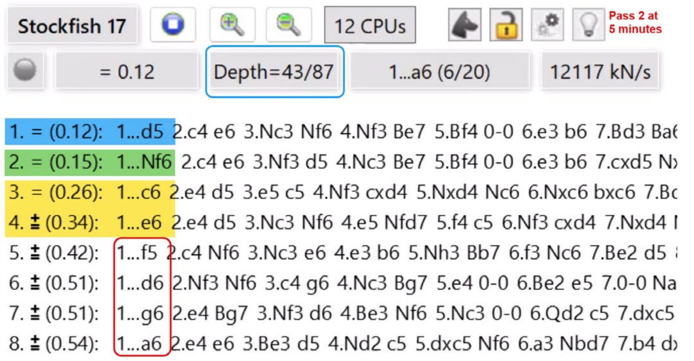
And we thought computers should always give the same results.

Here are the results for each pass at the end of the test.


Pass 1 at 5 minutes

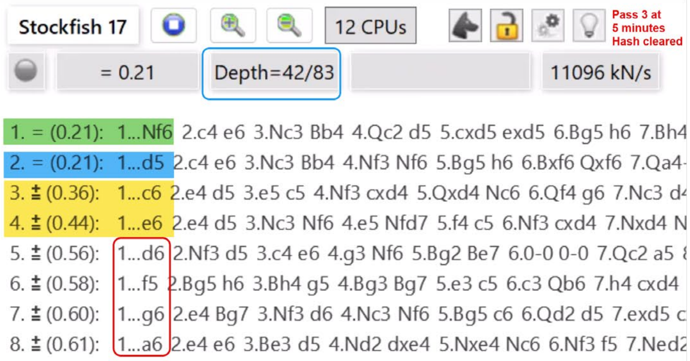
```
Stockfish 17   12 CPUs
= 0.10   Depth=42/79   1...f5 (6/20)   13184 kN/s

1. = (0.10):  1...Nf6 2.Nf3 e6 3.Bf4 c5 4.e3 Nc6 5.Nc3 cxd4 6.exd4 Be7 7.h3
2. = (0.14):  1...d5 2.c4 e6 3.Nf3 Nf6 4.g3 Bb4+ 5.Bd2 Be7 6.Bg2 0-0 7.Qc2
3. = (0.27):  1...c6 2.e4 d5 3.e5 c5 4.Nf3 cxd4 5.Nxd4 Nc6 6.Nxc6 bxc6 7.Bc
4. ± (0.33):  1...e6 2.e4 d5 3.Nc3 Nf6 4.e5 Nfd7 5.f4 c5 6.Nf3 cxd4 7.Nxd4
5. ± (0.44):  1...d6 2.Nf3 Nf6 3.c4 g6 4.Nc3 Bg7 5.e4 0-0 6.Be2 e5 7.d5 a5
6. ± (0.44):  1...f5 2.c4 e6 3.Nc3 Nf6 4.a3 d6 5.e3 g6 6.Be2 Bg7 7.b4 0-0 8.
7. ± (0.53):  1...a6 2.e4 e6 3.Be3 d5 4.Nd2 dxe4 5.Nxe4 Nd7 6.Bd3 Ngf6 7.
8. ± (0.55):  1...g6 2.e4 Bg7 3.c4 d6 4.Nc3 Nf6 5.h3 e5 6.d5 Nbd7 7.g4 Nc5
```


Pass 2 at 5 minutes

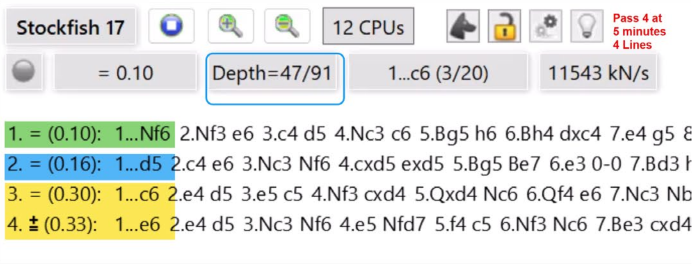
```
Stockfish 17   12 CPUs
= 0.12   Depth=43/87   1...a6 (6/20)   12117 kN/s

1. = (0.12):  1...d5 2.c4 e6 3.Nc3 Nf6 4.Nf3 Be7 5.Bf4 0-0 6.e3 b6 7.Bd3 Ba
2. = (0.15):  1...Nf6 2.c4 e6 3.Nf3 d5 4.Nc3 Be7 5.Bf4 0-0 6.e3 b6 7.cxd5 N
3. = (0.26):  1...c6 2.e4 d5 3.e5 c5 4.Nf3 cxd4 5.Nxd4 Nc6 6.Nxc6 bxc6 7.Bc
4. ± (0.34):  1...e6 2.e4 d5 3.Nc3 Nf6 4.e5 Nfd7 5.f4 c5 6.Nf3 cxd4 7.Nxd4
5. ± (0.42):  1...f5 2.c4 Nf6 3.Nc3 e6 4.e3 b6 5.Nh3 Bb7 6.f3 Nc6 7.Be2 d5
6. ± (0.51):  1...d6 2.Nf3 Nf6 3.c4 g6 4.Nc3 Bg7 5.e4 0-0 6.Be2 e5 7.0-0 Na
7. ± (0.51):  1...g6 2.e4 Bg7 3.Nf3 d6 4.Be3 Nf6 5.Nc3 0-0 6.Qd2 c5 7.dxc5
8. ± (0.54):  1...a6 2.e4 e6 3.Be3 d5 4.Nd2 c5 5.dxc5 Nf6 6.a3 Nbd7 7.b4 d
```


Pass 3 at 5 minutes Hash cleared

```
Stockfish 17   12 CPUs
= 0.21   Depth=42/83   11096 kN/s

1. = (0.21):  1...Nf6 2.c4 e6 3.Nc3 Bb4 4.Qc2 d5 5.cxd5 exd5 6.Bg5 h6 7.Bh4
2. = (0.21):  1...d5 2.c4 e6 3.Nc3 Bb4 4.Nf3 Nf6 5.Bg5 h6 6.Bxf6 Qxf6 7.Qa4
3. ± (0.36):  1...c6 2.e4 d5 3.e5 c5 4.Nf3 cxd4 5.Qxd4 Nc6 6.Qf4 g6 7.Nc3 d
4. ± (0.44):  1...e6 2.e4 d5 3.Nc3 Nf6 4.e5 Nfd7 5.f4 c5 6.Nf3 cxd4 7.Nxd4 N
5. ± (0.56):  1...d6 2.Nf3 d5 3.c4 e6 4.g3 Nf6 5.Bg2 Be7 6.0-0 0-0 7.Qc2 a5
6. ± (0.58):  1...f5 2.Bg5 h6 3.Bh4 g5 4.Bg3 Bg7 5.e3 c5 6.c3 Qb6 7.h4 cxd4
7. ± (0.60):  1...g6 2.e4 Bg7 3.Nf3 d6 4.Nc3 Nf6 5.Bg5 c6 6.Qd2 d5 7.exd5 c
8. ± (0.61):  1...a6 2.e4 e6 3.Be3 d5 4.Nd2 dxe4 5.Nxe4 Nc6 6.Nf3 f5 7.Ned2
```


Pass 4 at 5 minutes 4 Lines

```
Stockfish 17   12 CPUs
= 0.10   Depth=47/91   1...c6 (3/20)   11543 kN/s

1. = (0.10):  1...Nf6 2.Nf3 e6 3.c4 d5 4.Nc3 c6 5.Bg5 h6 6.Bh4 dxc4 7.e4 g5
2. = (0.16):  1...d5 2.c4 e6 3.Nc3 Nf6 4.cxd5 exd5 5.Bg5 Be7 6.e3 0-0 7.Bd3
3. = (0.30):  1...c6 2.e4 d5 3.e5 c5 4.Nf3 cxd4 5.Qxd4 Nc6 6.Qf4 e6 7.Nc3 Nb
4. ± (0.33):  1...e6 2.e4 d5 3.Nc3 Nf6 4.e5 Nfd7 5.f4 c5 6.Nf3 Nc6 7.Be3 cxd4
```