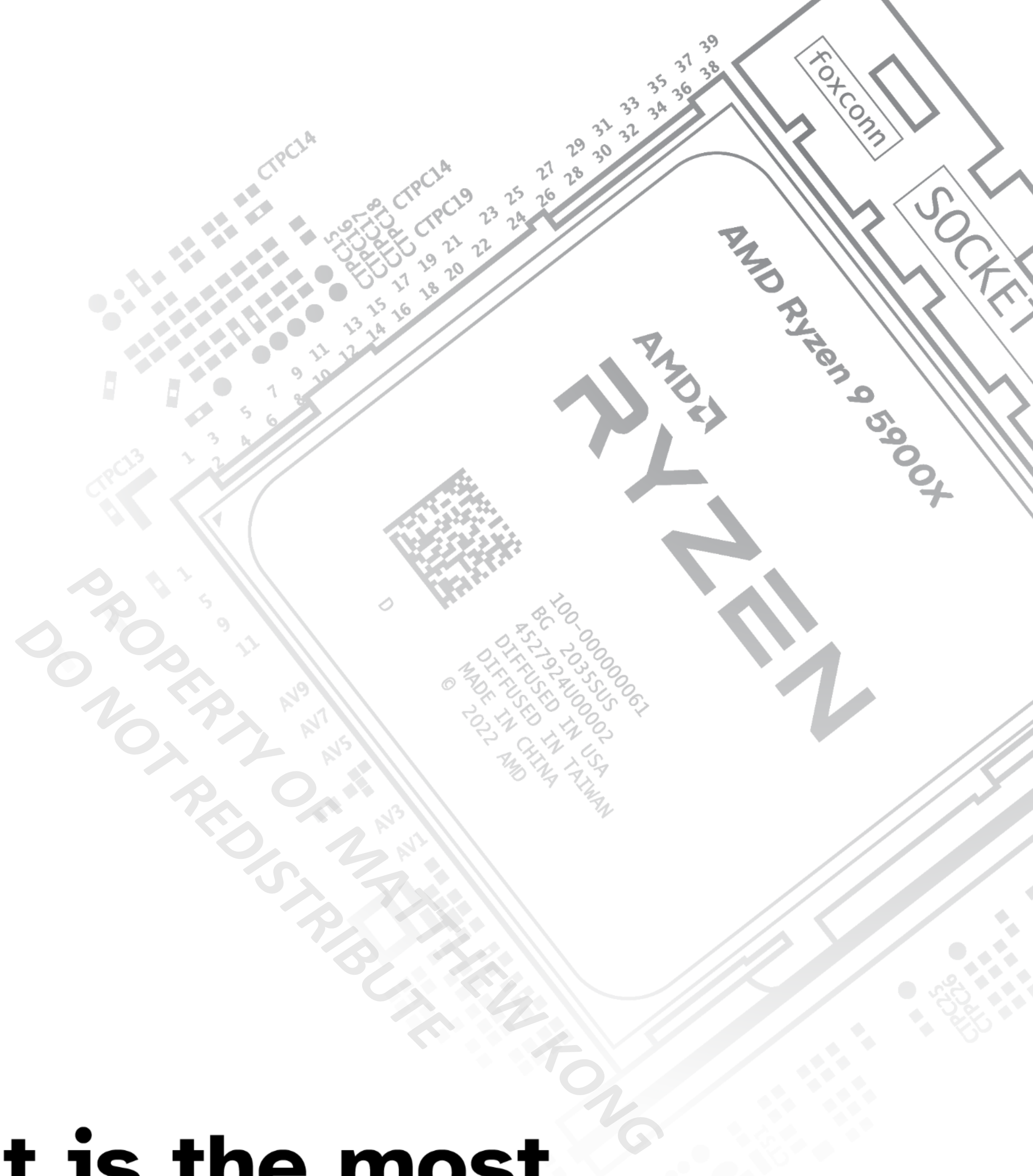


What is the most optimal PPT limit for my CPU?

IB Mathematics Internal Assessment

Introduction

Custom desktop personal computers (PC) have long intrigued me; their performance, stability, and value contrasted greatly with the old and sluggish computers I grew up with. This was made very apparent when I started running computationally intensive tasks like video editing software and video games during the COVID-19 lockdowns. As such, I decided to build my own computer, as can be seen in Fig. 1. However, the costs of a computer are very high – which, as an unemployed student, made the prospect of building and operating my own PC less feasible. These costs generally stem from the Central Processing Unit (CPU), which is the component that is responsible for performing calculations. A faster CPU therefore increases the overall performance of the computer, though the parameters modified to achieve this increase the operation cost. My paper therefore explores my journey to find a balance between the costs and the performance of my CPU. The model of CPU I tested, along with the rest of the components of my PC, can be found in Appendix A.

Fig 1: *An image of my computer.*

Modern CPUs do not consist of a single processor, however; instead, they are composed of several individual processors called cores that perform calculations independently of each other. Some tasks require only one core, and some can utilize many. I run both single-core and multi-core workloads on my computer, and so the speed and costs of both are important to me. This paper therefore explores what CPU parameters are optimal for both scenarios.

Other than upfront purchasing costs, a CPU's costs tend to result from its power consumption and longevity, the latter of which affects the time before it needs to be replaced. These factors are affected by several variables, such as the power limit, core voltages, and average core temperatures. The software I used to monitor and manipulate these parameters can be found in Appendix B.

CPU Package Power Tracking (PPT) was the independent variable in my investigation. In essence, it is the maximum amount of power allowed to be supplied to the entire CPU, measured in watts (W). Increased power consumption is generally not desired because it increases operating costs, as electricity costs money. It generally allows for increased CPU voltage though, which in turn results in greater performance.

Voltage (VID) is another parameter that affects the cost. It represents the electrical pressure applied to a CPU, with higher VIDs pressuring the cores to operate at a faster processing speed. However, elevated voltages can potentially reduce the CPU's lifespan, as the cores may experience malfunctions when subjected to excessive pressure. This is problematic cost-wise because it necessitates additional expenses for replacing the CPU earlier than expected.

CPU temperatures are closely related to both voltage and power consumption in that increases in both directly result in a hotter running processor. Similarly to voltage, high temperatures also lead to malfunctioning cores, which in turn decreases the lifespan of a CPU. This was measured by the die temperature (Tdie), or the highest reported temperature across all cores.

The purpose of this paper is to find the balance between my CPU's PPT, VIDs, Tdie, and its achievable performance, so that I can more easily afford to operate the computer.

Methodology

To find the balance, I will use the point of diminishing returns, i.e., where increases in the PPT limit no longer lead to noticeable increases in performance or longevity. There are two main approaches I will use to accomplish this: a power consumption optimization, and a CPU longevity optimization. The former was conducted via a comparison between PPT limit increases and performance gains in several workloads. Specifically, a preliminary gaming test was conducted within the domain of $\{PPT \in [40\ W, 100\ W] : 10\}$ in a video game called Forza Horizon 5 (FH5) using the in-built benchmark, which is pictured in Fig. 2. I chose the lower boundary of this domain because my CPU would not function at PPT limits less than $40\ W$, and I chose the upper boundary because my CPU would not consume above $200\ W$ of power in any of my tests. Outside of video editing, this video game represents the computationally intensive task that I run the most on my computer. It is particularly useful for the purposes of this paper because performance in this benchmark is determined by how many frames per second (FPS) can be processed and rendered in the benchmark run, which leads to an observably smoother experience. I was therefore able to use this metric to determine where I stopped noticing a difference in FPS increase, which I then converted to a percentage of the regression's asymptote so that I could find the point of diminishing returns in other CPU benchmarks.

I then increased the PPT limit and observed performance increases in the multi-core and single-core tests of Cinebench R23 (Fig. 3) within the domain of $\{PPT \in [40\ W, 200\ W] : 10\}$. I chose this domain for the same reasons as in the FH5 benchmark. The performance in both benchmarks is represented by a score, where a higher score signifies a faster-running CPU. I then performed regressions based on the PPT limit and its associated score, after which I solved for the optimum PPT limits using the previously mentioned percentage.

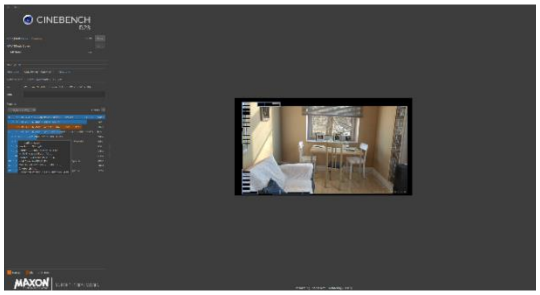


Fig 3: *A Cinebench R23 benchmark run.*

I also optimized CPU longevity, which was largely centered around the VIDs and Tdie values reported during the Cinebench R23 multi-core benchmark runs. Since the trends of both values were similar in the single-core benchmark, I limited my scope to the multi-core tests. Afterwards, I observed the impact of increased PPT limits on the VIDs using a regression analysis. In this analysis, I found the inflection point, and subsequently its corresponding PPT limit. Then, I observed the impact of increases in PPT limit on the Tdie and performed a regression. My primary goal with this was to determine the PPT limit that results in "unsafe" temperatures – that is, temperatures that lead to rapid CPU degradation. However, my CPU never reached the maximum rated temperature of $95\ °C$, and so I resorted to a more qualitative temperature limit. In my computer, fans are installed to cool the CPU, and the speed at which they spin corresponds with its temperatures. When fans spin faster, they produce more noise, which can be distracting when the computer is in use. Therefore, by adjusting their speed until they became distracting, I found an ideal Tdie value and the corresponding PPT limit. The fan curve used to determine this value can be seen in Appendix C.

After these tests, I averaged the PPT limits of the multi-core tests to find the point of diminishing returns for both multi-core and single-core CPU workloads.

Preliminary Forza Horizon 5 (FH5) Benchmark Analysis

Table 1 is a summary of the results of my Forza Horizon 5 runs. Since the rate of FPS increase seems to approach 0 as the PPT increases, I decided to perform a regression of the data in Desmos with a vertically reflected exponential decay function. The following is the formula I regressed, where x_1 represents the PPT limit, and $f_1(x_1)$ represents the average FPS achieved with respect to x_1.

PPT Limit (W)	FH5 FPS Average
40	40.10
50	143.4
60	291.9
70	309.2
80	322.3
90	318.8
100	319.5

Table 1: *PPT limit (W) vs. FH5 FPS Averages. Tables with all raw data collected can be found in Appendix D.*

$$f_1(x_1) = -a^{(x_1-b)} + c$$

This produced the following values, which are kept to 4 significant figures to match the FPS average, and this was kept constant throughout the paper.

1. $a = 0.9335$: Since $0 < a < 1$, the function decreases exponentially, which allows for the FPS to approach the asymptote as the PPT limit approaches ∞, and therefore results in the expected levelling off behaviour.
2. $b = 123.2$: This value horizontally translates the function $123.2\ W$ right, which is necessary since the data does not appear to cross the origin. This makes the regression

less accurate though, since it implies the existence of negative FPS, which is not possible. However, my CPU would not accept PPT limits below $40\ W$, so I believe this is an acceptable limitation and thus it is still sufficiently representative of its behaviour.

3. $c = 336.6$: This value is the horizontal asymptote of the function. It represents the theoretical highest FPS achievable, assuming an infinite power limit. As expected, the measured scores generally appear to approach these values as PPT limit increases. However, datapoints in the domain of $\{PPT \in [80\ W, 100\ W] : 10\}$ do not follow this trend, as they appear to level off about $y = 325$. I therefore manually set it to 325 for $g_1(x_1)$ with newly regressed a and b values, which produced an asymptote that more closely resembled the observed right end behaviour. Below is a graph of both functions, where $y_1 = 336.6$ is the asymptote of $f_1(x_1)$ and $y_2 = 325$ is the asymptote of $g_1(x_1)$.

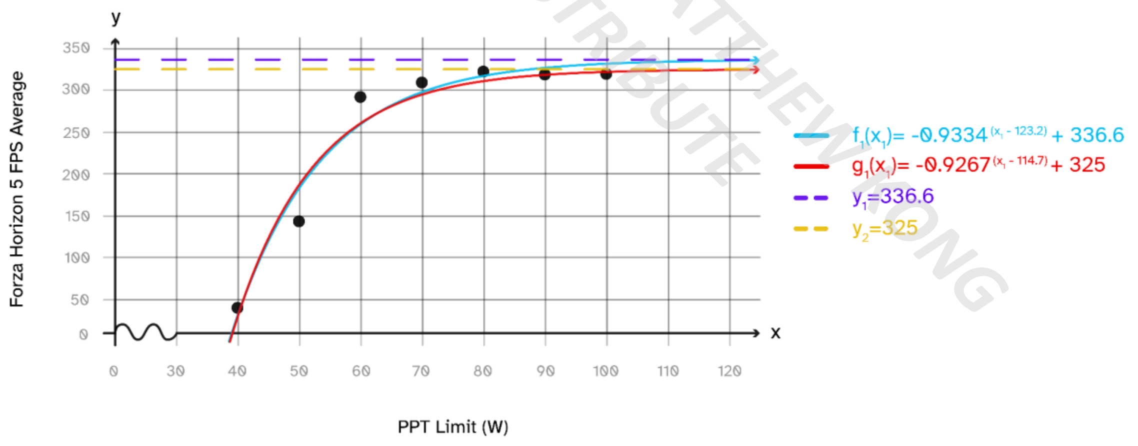


Fig. 4: *PPT Limit (W) vs. average Forza Horizon 5 FPS plotted with two exponential regressions and their respective asymptotes.*

As can be seen in Fig. 4, neither regression fits the data perfectly. However, $g_1(x_1)$ with its asymptote of $y_2 = 325$ more closely follows the right end behaviour of the data (where an FPS value greater than 325 is not achieved), and so I deemed it to be a better-fitting model. This resulted in an r^2 of 0.9558, which signifies that there exists a strong correlation.

I then sought to determine the point of diminishing returns for $g_1(x_1)$. From my own experience, I couldn't tell a difference in visual smoothness once it reached about a $1\ FPS\ \cdot W^{-1}$ increase, and so I found the derivative at this point on the regression to solve for the FPS achieved. By letting $g_1(x_1) = y$ where y represents the FPS, and using Leibniz notation, I found the derivative.

$$y = -0.9267^{(x_1 - 114.7)} + 325$$

$$\frac{dy}{dx_1} = -[\ln(0.9267)]\left[0.9267^{(x_1 - 114.7)}\right]$$

Then, I let the derivative be equal to the point of diminishing returns.

$$-[\ln(0.9267)]\left[0.9267^{(x_1 - 114.7)}\right] = 1$$

$$x_1 = \log_{0.9267}\left[-\frac{1}{\ln(0.9267)}\right] + 114.7$$

$$x_1 \approx 80.93\ W$$

Afterwards, I substituted x_1 into $g_1(x_1)$ to find the FPS at x_1.

$$y = -0.9267^{(80.93 - 114.7)} + 325$$

$$y \approx 311.9\ FPS$$

However, this coordinate isn't useful outside of this specific benchmark because other benchmarks use different performance metrics, namely a relative score instead of FPS. As such, I decided to convert it to a percentage of the asymptote, which could then be used to find the point of diminishing returns for other benchmarks with exponential regressions. In the following formula, CF represents the conversion factor, or the percentage of the asymptote at the point of diminishing returns. Additionally, y is the optimal FPS, and y_2 is the asymptote.

$$CF = \frac{y}{y_2}$$

$$\approx \frac{311.9}{325}$$

$$CF \approx 96.0\%$$

Cinebench R23 Single-Core Analysis

Table 2 is a summary of the results of my Cinebench R23 single-core runs. This data set seemed to follow a pattern like that of the Forza Horizon 5 benchmark, where the performance is exponential and approaches a horizontal asymptote from $-\infty$ as the PPT limit increases. I thus decided to utilize the same regression formula. In Desmos, my regression produced the following equation.

PPT Limit (W)	Cinebench R23 Single-Core Score
40	943
50	1466
60	1627
70	1618
80	1626
90	1636
100	1634

Table 2: *PPT limit (W) vs. Cinebench R23 Single-Core Score. Tables with all raw data collected can be found in Appendix D.*

$$f_2(x_2) = -0.8629^{(x_2 - 84.38)} + 1636$$

I then plotted the PPT Limit against the Cinebench R23 single-core score.

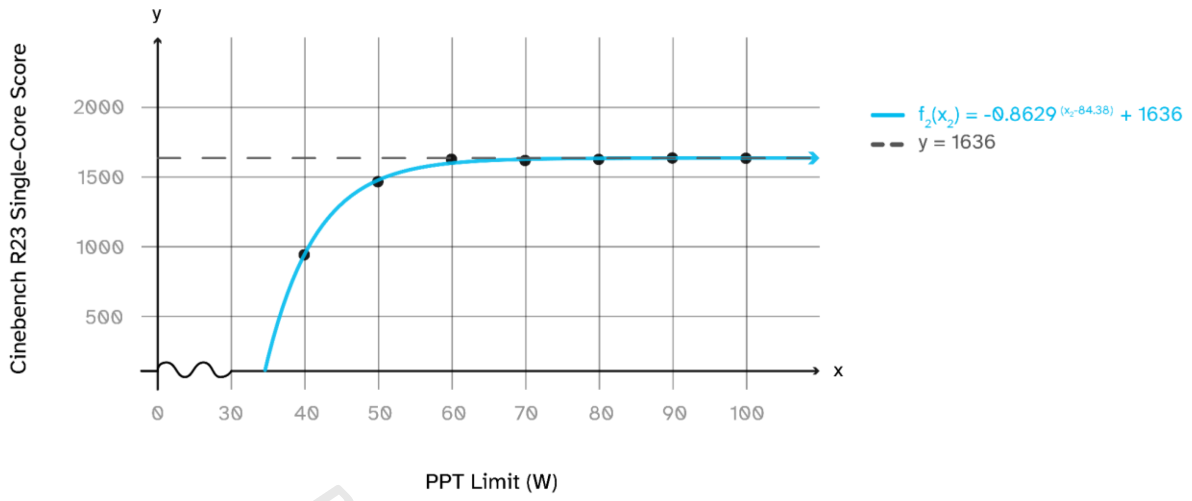


Fig. 5: *PPT Limit (W) vs. Cinebench R23 Single-Core Score plotted with an exponential regression and its asymptote.*

As can be seen in Fig. 5, the regression fits the data well; the regression produced an r^2 of 0.9974, which is close to a perfect r^2 of 1 and signifies a strong correlation. Although the data point for $x_2 = 90\ W$ intersects the asymptote, I deemed it to be acceptable because of run-to-run variance; Cinebench R23 isn't perfectly consistent because my CPU also needs to run other background tasks. Small deviations from the expected pattern are thus expected.

Then, to find the point of diminishing returns, I multiplied the asymptote (y) with the previously obtained conversion factor (CF) to calculate the point of diminishing returns (PDR).

$$\begin{aligned} PDR &= \mathrm{y}(CF) \\ &\approx 1636(0.96) \\ &\approx 1571 \end{aligned}$$

Afterwards, I solved $f(x) = 1571$ to find the optimal PPT limit for this benchmark.

$$-0.8629^{(x_2-84.38)} + 1636 = 1571$$

$$\log_{0.8629}(1636 - 1571) = x_2 - 84.38$$

$$x_2 \approx 56.0 \text{ W}$$

Therefore, my CPU's most optimal PPT limit for single-core workloads is $56.0\ W$.

Cinebench R23 Multi-Core Analysis

The following is a summarized table of the results of my Cinebench R23 multi-core runs.

PPT limit (W)	Cinebench R23 Multi-Core Score	Average Core VIDs (V)	CPU Tdie (°C)
40	2419	0.8941	31.5
50	5328	0.8931	33.4
60	9412	0.9033	35.5
70	12356	0.9114	39.6
80	15878	0.9235	43.0
90	18142	0.9301	46.6
100	19165	0.9731	50.6
110	19953	1.010	54.3
120	20598	1.051	57.5
130	21231	1.088	60.9
140	21788	1.122	61.3
150	21991	1.138	67.7
160	22291	1.179	71.4
170	22401	1.203	75.2
180	22581	1.232	79.3
190	22542	1.242	81.7
200	22697	1.237	81.7

Table 3: A table displaying the raw data collected during the Cinebench R23 multi-core tests. Tables with all raw data collected can be found in Appendix D.

I started my analysis of this data with a comparison of the PPT limits and the corresponding Cinebench R23 multi-core scores. This seemed to follow a similar exponential relationship to the Forza Horizon 5 analysis, so I used the same regression formula on this data set using Desmos, where x_3 represents the Cinebench R23 multi-core score, and $f_3(x_3)$ represents the PPT limit with respect to x_3. This produced the following equation.

$$f_3(x_3) = -0.9747^{(x_3 - 429.5)} + 23325$$

Using this regression, I plotted the PPT limit against the multi-core score.

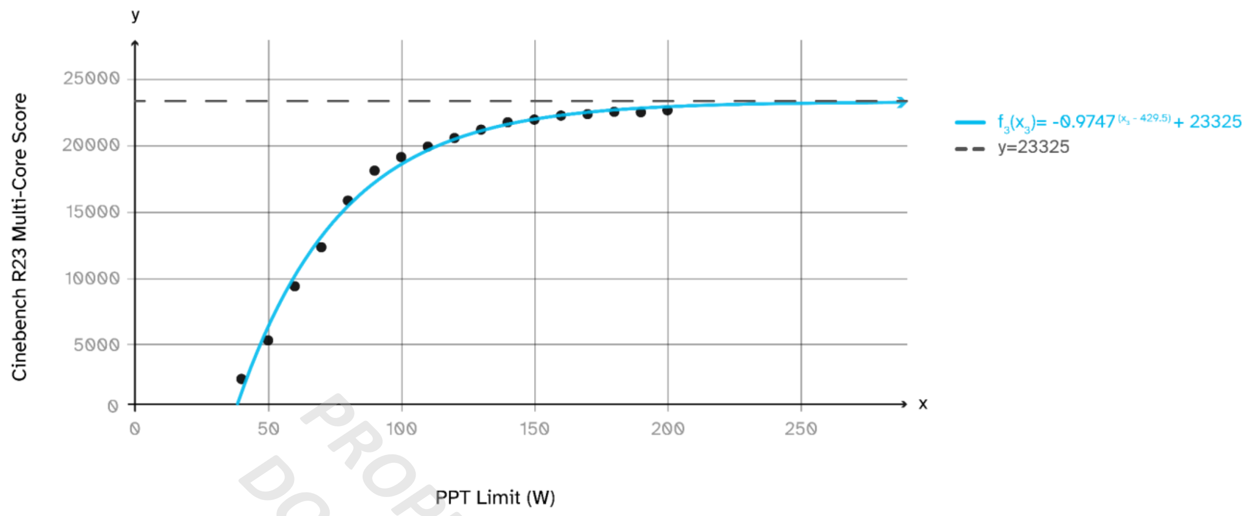


Fig. 6: *PPT limit (W) vs. Cinebench R23 Multi-Core Score plotted with an exponential regression and its asymptote.*

The data in Fig. 6 seems to fit the regression quite well, with an $r^2$of 0.9925 indicating a strong correlation. Then, to find the score at the point of diminishing returns, I multiplied the asymptote (y) by the conversion factor obtained in the Forza Horizon 5 benchmark (CF) to obtain the point of diminishing returns (PDR). See the full calculation in Appendix E.

$$PDR \approx 22392$$

Then, I solved for the PPT limit by letting $f(x) = PDR$. See the full calculation in Appendix E.

$$x_3 \approx 163.0\ W$$

Therefore, my CPU's most optimal PPT limit for multi-core workloads is $163.0\ W$.

Voltage Analysis

Outside of just power consumption optimizations, I also wanted to optimize my CPUs longevity. Specifically, since the voltage supplied to a CPU directly influences its lifespan, I wanted to minimize this value as much as possible. To do this, I compared the PPT Limit with the Average Core VIDs measured in the Cinebench R23 Multi-Core benchmark, the values of which can be found in Table 3. However, as seen in the following graph, the relationship between these two variables was not the same as in the power consumption optimizations.

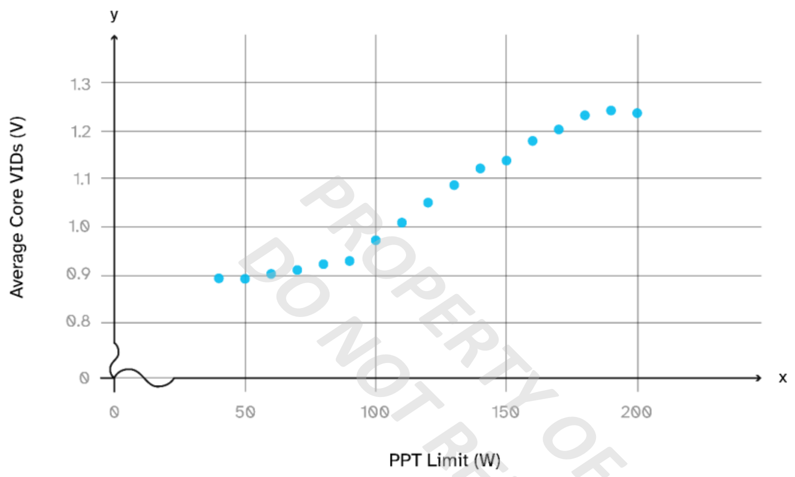


Fig. 7: PPT limit (W) vs. Cinebench R23 Multi-Core VIDs.

As can be seen in seen in Fig. 7, the data does not seem to follow the same concave-down exponential increase relationship as in the power consumption optimizations. In fact, it appears almost logistic, with both the right and left end behaviours levelling off. However, a logistic regression would imply the existence of horizontal asymptotes, which does not coincide with the behaviour that should occur; generally, increases in power consumption are the direct result of increases in voltage, and so the voltage should never approach a set value without ever intersecting. A sine regression would also model the right and left end behaviours for PPT values within the domain of $\{PPT \in [40\ W, 200\ W]: 10\}$, but is bound by a similar flaw; sine functions are periodic, meaning they are not always increasing. This would therefore imply that certain domains of PPT values greater than those tested would result in decreasing voltages, which is incorrect because increases in power consumption should always result in increases in voltage.

I settled on an inverse tangent function, which I deemed to be more fitting because it effectively models the right and left end behaviours of the data, and never approaches an asymptote. Below is the function I regressed, where x_4 represents the PPT, and $f_4(x_4)$ represents the voltage at x_4.

$$f_4(x_4) = a\arctan(bx_4 - c) + d, \qquad \left\{f_4(x_4) \in \left[-\frac{a\pi}{2} + d, \frac{a\pi}{2} + d\right]\right\}$$

After performing the regression using Desmos, I obtained the following values.

1. $a = 0.1620$: Since the scale that VIDs operate on is very small, a vertical compression allows the function to model this scale more effectively.
2. $b = 0.02540$: Similarly, a horizontal stretch better models the scale of the PPT limit.
3. $c = 3.226$: A phase shift is needed because the inflection point of the base inverse tangent function is on the y-axis, which does not occur in this data set.
4. $d = 1.070$: Similarly, a vertical translation is needed because the inflection point of the base inverse tangent function is on the x-axis, which does not occur in this data set.

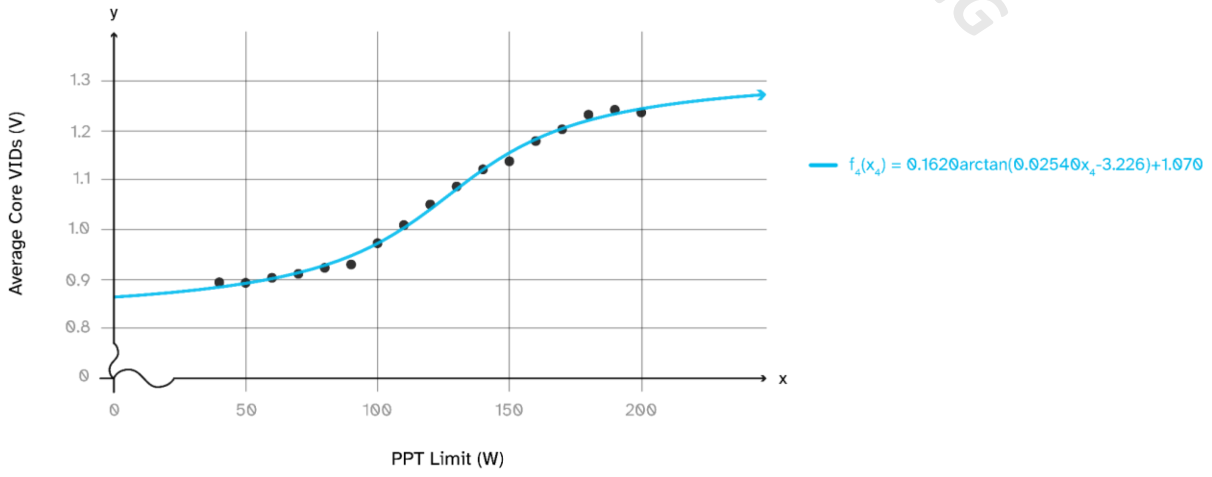


Fig. 8: PPT limit (W) vs. Cinebench R23 Multi-Core VIDs plotted with an inverse tangent regression.

As seen in the Fig. 8, this regression models the relationship quite well, with an r^2 of 0.996 indicating a strong correlation between the variables. However, like with the previous optimizations, this model does not extrapolate well for PPT limits below those tested. For instance, a PPT limit of $0\ W$ should theoretically result in a VID of $0\ V$, since the CPU would not be able to run without power. This is not represented in the model. In practice, this flaw does not matter since my CPU is can't use a PPT limit less than $40\ W$, and so I believe this is acceptable.

Since the regression is an inverse tangent function, it is concave down after an inflection point. This is likely because increases to the PPT limit also lead to increases in temperature, which inadvertently hinders the voltages achievable by the CPU. This process is known as thermal throttling and is not desirable because it can lead to rapid CPU degradation. As such, the greatest PPT limit where this does not occur is optimal, so I let the inflection point be the point of diminishing returns. By letting $f_4(x_4) = y$ where y represents the FPS, and using Leibniz notation, I found the derivative.

$$y = 0.1620 \arctan(0.02540x_4 - 3.226) + 1.070$$

$$\frac{dy}{dx_4} = \frac{(0.1620)(0.02540)}{1 + (0.02540x_4 - 3.226)^2}$$

Then, I found the second derivative.

$$\frac{d^2y}{dx_4{}^2} = \frac{-(0.1620)(0.02540)\frac{d}{dx_4}[1 + (0.02540x_4 - 3.226)^2]}{[1 + (0.02540x_4 - 3.226)^2]^2}$$

$$\frac{d^2y}{dx_4{}^2} = \frac{-2(0.1620)(0.02540)(0.02540x_4 - 3.226)(0.02540)}{[1 + (0.02540x_4 - 3.226)^2]^2}$$

To find the inflection point, I set the second derivative equal to zero.

$$\frac{-2(0.1620)(0.02540)(0.02540x_4 - 3.226)(0.02540)}{[1 + (0.02540x_4 - 3.226)^2]^2} = 0$$

$$-2(0.1620)(0.02540)(0.02540x_4 - 3.226)(0.02540) = 0$$

$$x_4 \approx 127.0\,W$$

Therefore, my CPU's most optimal PPT limit for multi-core workloads, when considering voltage, is $x_4 \approx 127.0\,W$.

Temperature Analysis

Similarly to voltage, operating temperature is also very closely tied to a CPU's lifespan, and so I decided to observe the effect of increases to the PPT limit on the CPU Tdie in the Cinebench R23 multi-core test. The temperature data presented in Table 3 appears to follow a linear increase, and so I decided to perform a linear regression using the following expression, where x represents the PPT limit, and $f_5(x_5)$ represents the Tdie temperature (°C) at x_5.

$$f_5(x_5) = ax_5 + b$$

After performing the regression in Desmos, I obtained the following values.

1. $a = 0.3396$: This value represents the gradient of the increase in Tdie temperature.
2. $b = 16.38$: A vertical translation is needed since the temperature does not appear to intersect the origin.

I then plotted the PPT limit against the Tdie temperature.

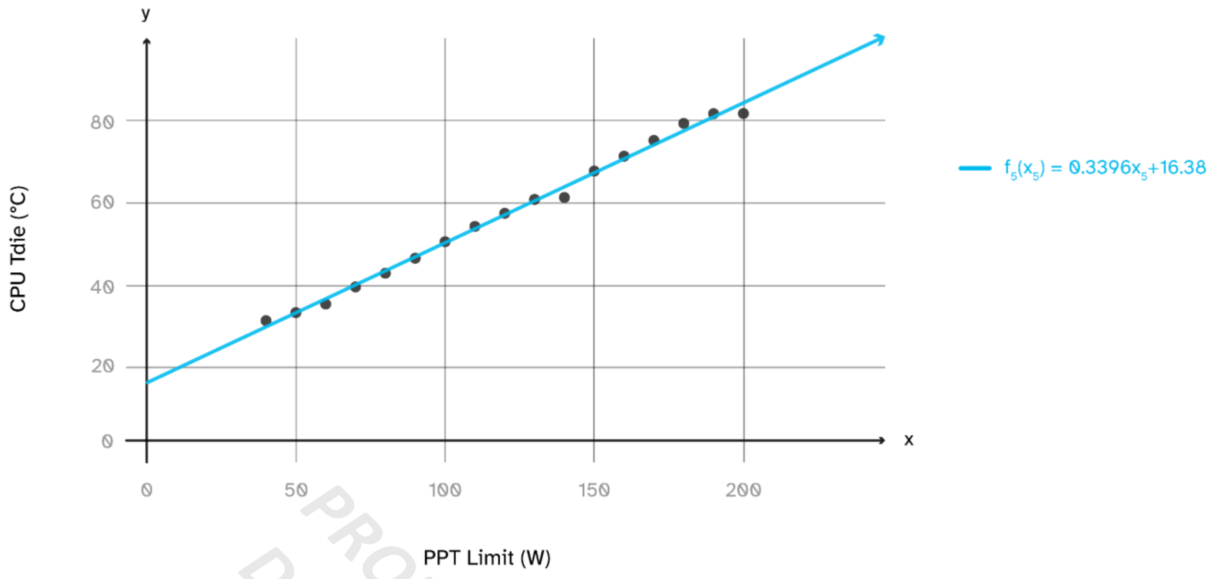


Fig. 9: *PPT limit (W) vs. Cinebench R23 Multi-Core VIDs plotted with an inverse tangent regression.*

As seen in Fig. 9, the linear regression fits this data very well, with an r^2 of 0.9949 indicating a very strong relationship. However, this regression does not extrapolate accurately. For instance, it implies that a PPT limit of $0\ W$ results in a Tdie of $16.38\ °C$, which is lower than room temperature (See appendix A for my test conditions). My CPU would not accept PPT limits below $40\ W$ though, so I believe that this is an acceptable limitation. It also implies that the Tdie approaches ∞ as the PPT limit increases, which is not possible because it should thermally throttle after it reaches the maximum allowed temperature of $95\ °C$, where the CPU lifespan is drastically decreased. These values are thus not ideal cost-wise, and so I believe that this is also an acceptable limitation.

To find the point of diminishing returns, I observed the noise produced by the fans in my computer. Specifically, my fans spin faster in response to higher CPU temperatures, which in turn produces more noise and can be distracting if too loud. I thus manually altered the fan speed and found that it became unpleasant to listen to at about 80% speed, which is set to correspond with a $70\ °C$ Tdie temperature (See Appendix C for the default fan curve). So, by letting $f_5(x_5) = 70$, I found the point of diminishing returns in terms of PPT limit.

$$0.3396x_5 + 16.38 = 70$$

$$x_5 \approx 157.9\ W$$

Therefore, my CPU's most optimal PPT limit for multi-core workloads, when considering temperature, is $x_5 = 157.9\ W$.

How well does my data fit?

Qualitatively, my regressions model my data sufficiently well. A quantitative analysis of the regressions was able to substantiate this using the coefficient of determination (r^2), which is a measure of how strong the correlation is; values closer to 1 signify a greater correlation, and values closer to 0 signify a lower correlation. Using Desmos, I was able to obtain the r^2 for each regression, which can be seen in Table 4.

Regression	r^2
$g_1(x_1)$	0.9558
$f_2(x_2)$	0.9974
$f_3(x_3)$	0.9925
$f_4(x_4)$	0.9960
$f_5(x_5)$	0.9949

Table 3: *Regression functions and their r^2.*

The r^2 values for each of the regressions are all very high, which signifies a very strong relationship. However, the r^2 value for $g_1(x_1)$ is noticeably lower than the other regressions (albeit still very high) though, which means that it is less precise. As such, I decided to omit its point of diminishing returns (x_1) from my calculation of the optimal PPT limit.

Conclusion

By calculating the average point of diminishing returns, I determined the optimal PPT limit for both single-core and multi-core scenarios (Fig. 10). The optimal PPT limit for my CPU in multi-core workloads is therefore $149.3\ W$, and $56.0\ W$ for single-core workloads. These values are the result of optimizations that consider its longevity, performance and power consumption. Furthermore, the regressions performed on the data all have high r^2 values, meaning that there are strong relationships.

$$x_{multi} = \frac{x_3 + x_4 + x_5}{3}$$

$$x_{multi} = \frac{163.0 + 127.0 + 157.9}{3}$$

$$x_{multi} = 149.3\ W$$

And...

$$x_{single} = x_2$$

$$x_{single} = 56.0\ W$$

Fig. 10: Finding the mean optimal PPT

While my regression models may have technically yielded precise results, they do possess certain limitations. Notably, the averaging of PPT limits from various multi-core optimizations fails to accurately represent each optimization individually. The voltage optimization (x_4), for instance, is $12.3\ W$ less than the average (x_{multi}), thus surpassing the inflection point and falling within voltage ranges I deemed unsafe. To address this, I could introduce weighting by using the following formula in a future analysis.

$$x_{multi} = 0.6x_3 + 0.2x_4 + 0.2x_5$$

Determining the weighting is very subjective though. Do I value CPU performance more or less than its lifespan, for instance? An extension could therefore be to qualitatively determine my preferred weighting, which could produce a more desirable optimized PPT limit.

Additionally, several of my models do not realistically extrapolate to PPT limits that are less than those tested. For example, my Cinebench R23 multi-core regression model implies that a PPT limit of $0\ W$ would result in a negative score, which is not possible. However, my CPU cannot operate at PPT limits below $40\ W$, so this extrapolation is not applicable. Regardless, I could try to account for this in future analyses by using a function that exhibits a horizontal asymptote in both the right and left end behaviours, such as a logistic curve.

Furthermore, I only analyzed my specific CPU, and so the results are not applicable elsewhere. To address this limitation, I could find the point of diminishing returns of other CPU models in a future analysis.

Overall, I was very pleased with my analysis and its outcomes. The investigation has enabled me to fine-tune the PPT limit in relation to my CPUs performance, VIDs, and Tdie, making it more feasible for me to run CPU-intensive tasks on my computer. It has also allowed me to explore several mathematical concepts and their use in the real world. Calculating the derivative of exponential and inverse tangent functions, for instance, has broadened my understanding of differential calculus, and has fostered a greater appreciation of the intricacies of mathematical modeling and analysis.

References

Alcorn, Paul. (21 July 2022). *How to Safely Overclock Your CPU: Get the Most MHz from Your Processor*. Tom's Hardware. www.tomshardware.com/how-to/how-to-overclock-a-cpu

(26 May 2023). *Desmos | Let's Learn Together*. Desmos. www.desmos.com/

Lathan, Patrick. (15 July 2019). *Explaining AMD Ryzen Precision Boost Overdrive (PBO), AutoOC, & Benchmarks*. GamersNexus. www.gamersnexus.net/guides/3491-explaining-precision-boost-overdrive-benchmarks-auto-oc

Fisher, R. (2018, June 11). *What's Thermal Throttling and How to Prevent It*. TechSpot; TechSpot. https://www.techspot.com/article/1638-what-is-thermal-throttling/

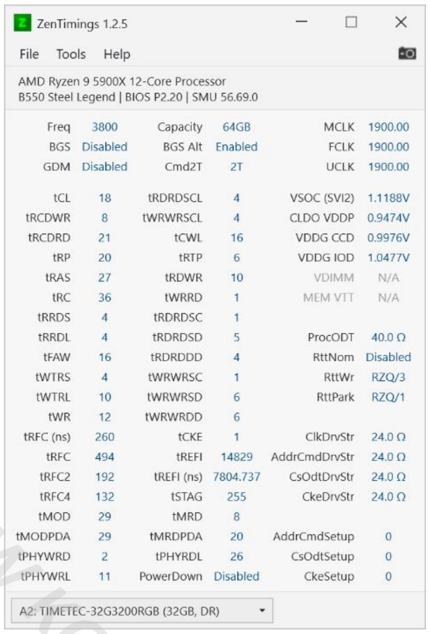
Appendices

Appendix A

Test conditions and the components of my PC.

Test Conditions

- Room temperature set to a constant 24 ℃ using an air conditioner.
- My RAM was overclocked (Fig. 11).
- My GPU was not overclocked.
- I conducted the tests using the Windows 10 IoT Enterprise LTSC 21H2 19044.3086 operating system.
- My motherboard BIOS was on version P2.20 (AGESA 1.2.0.6b).



AMD Ryzen 9 5900X 12-Core Processor
B550 Steel Legend | BIOS P2.20 | SMU 56.69.0

Freq	3800	Capacity	64GB	MCLK	1900.00
BGS	Disabled	BGS Alt	Enabled	FCLK	1900.00
GDM	Disabled	Cmd2T	2T	UCLK	1900.00
tCL	18	tRDRDSCL	4	VSOC (SVI2)	1.1188V
tRCDWR	8	tWRWRSCL	4	CLDO VDDP	0.9474V
tRCDRD	21	tCWL	16	VDDG CCD	0.9976V
tRP	20	tRTP	6	VDDG IOD	1.0477V
tRAS	27	tRDWR	10	VDIMM	N/A
tRC	36	tWRRD	1	MEM VTT	N/A
tRRDS	4	tRDRDSC	1		
tRRDL	4	tRDRDSD	5	ProcODT	40.0 Ω
tFAW	16	tRDRDDD	4	RttNom	Disabled
tWTRS	4	tWRWRSC	1	RttWr	RZQ/3
tWTRL	10	tWRWRSD	6	RttPark	RZQ/1
tWR	12	tWRWRDD	6		
tRFC (ns)	260	tCKE	1	ClkDrvStr	24.0 Ω
tRFC	494	tREFI	14829	AddrCmdDrvStr	24.0 Ω
tRFC2	192	tREFI (ns)	7804.737	CsOdtDrvStr	24.0 Ω
tRFC4	132	tSTAG	255	CkeDrvStr	24.0 Ω
tMOD	29	tMRD	8		
tMODPDA	29	tMRDPDA	20	AddrCmdSetup	0
tPHYWRD	2	tPHYRDL	26	CsOdtSetup	0
tPHYWRL	11	PowerDown	Disabled	CkeSetup	0

A2: TIMETEC-32G3200RGB (32GB, DR)

Fig. 11: The overclock settings of my RAM.

PC Components

- CPU: AMD Ryzen 9 5900X
- CPU Cooler: Deepcool CASTLE 360EX
- Thermal Compound: ARCTIC MX-4
- Motherboard: ASRock B550 Steel Legend
- RAM: Timetec Pinnacle Konduit RGB 2x32GB DDR4 3200MHz CL16-18-18-38
- Storage
  - Kingston A2000 1TB
  - Teamgroup MP33 1TB
- GPU: ASUS Dual RTX 3060 Ti
- Case: Corsair 4000D Airflow

- PSU: EVGA SuperNOVA 750 B2
- Fans
    - ARCTIC P12 PWM PST A-RGB (6 installed)
    - Deepcool TF-120S (2 installed)
- Monitor: Samsung S70A 27.0” 3840x2160 60 Hz

Appendix B

Manipulating PPT and monitoring CPU performance, VIDs and Tdie.

The PPT limit was manipulated via my motherboard’s BIOS using PBO. Besides PPT, the following are the PBO settings that I modified.

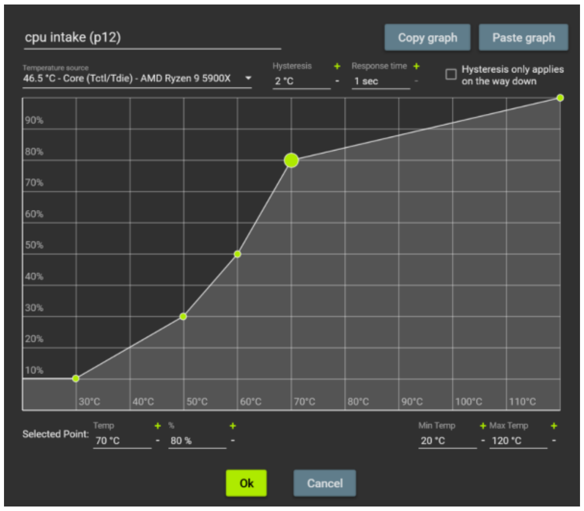
- EDC: $155\ A$
- TDC: $125\ A$
- Boost override: $50\ MHz$

To monitor the VIDs and Tdie, I used HWInfo, which outputted the values to a .csv file with a polling period of $500\ ms$. This was then aggregated in Excel and graphed with Desmos. The graphs were exported as a .svg file, and then edited in Adobe Illustrator.

Appendix C

The default fan curve.

Fig. 12 shows the default fan curve of my PC, which was found in a program called Fan Control. Using the same program, I adjusted the fan speed until I became annoyed with its sound.



cpu intake (p12)

Temperature source: 46.5 °C - Core (Tctl/Tdie) - AMD Ryzen 9 5900X

Hysteresis: 2 °C

Response time: 1 sec

☐ Hysteresis only applies on the way down

Selected Point: Temp 70 °C, % 80 %

Min Temp: 20 °C

Max Temp: 120 °C

Fig. 12: The default fan curve.

Appendix D

All the aggregated raw data collected during my tests.

https://docs.google.com/spreadsheets/d/e/2PACX-1vSP-LibnBN91BJLOoXrvWcOv3dq1jTbNC7blbbNpJ-04MCo6gHLBpCzdWx6UEL7LA/pubhtml

Appendix E

Repeated calculations that were removed for the sake of concision.

Calculating PDR for $f_3(x_3)$:

$$\begin{aligned} PDR &= \mathrm{y}(CF) \\ &\approx 23325(0.96) \\ &\approx 22392 \end{aligned}$$

Calculating x_3 using PDR:

$$-0.9747^{(x_3-429.5)} + 23325 = 22392$$

$$0.9747^{(x_3-429.5)} = 23325 - 22392$$

$$\log_{0.9747}(23325 - 22392) = x_3 - 429.5$$

$$x_3 \approx 163.0\,W$$