Parallelism Analysis of Prominent Desktop Applications: An 18-Year Perspective

Siying Feng, Subhankar Pal, Yichen Yang, Ronald G. Dreslinski
University of Michigan, Ann Arbor, MI, USA
{fengsy, subh, yangych, rdreslin}@umich.edu

Abstract—Improvements in clock speed and exploitation of Instruction-Level Parallelism (ILP) hit a roadblock during mid-2000s. This, coupled with the demise of Dennard scaling, led to the rise of multi-core machines. Today, multi-core processors are ubiquitous and architects have moved to specialization to work around the walls hit by single-core performance and chip Thermal Design Power (TDP). The pressure of innovation in the aftermath of Dennard scaling is shifting to software developers, who are required to write programs that make the most effective use of underlying hardware. This work presents quantitative and qualitative analyses of how software has evolved to reap the benefits of multi-core and heterogeneous computers, compared to state-of-the-art systems in 2000 and 2010. We study a wide spectrum of commonly-used applications on a state-of-the-art desktop machine and analyze two important metrics, Thread-Level Parallelism (TLP) and GPU utilization.

We compare the results to prior work over the last two decades, which state that 2-3 CPU cores are sufficient for most applications and that the GPU is usually under-utilized. Our analyses show that the harnessed parallelism has improved and emerging workloads show good utilization of hardware resources. The average TLP across the applications we study is 3.1, with most applications attaining the maximum instantaneous TLP of 12 during execution. The GPU is over-provisioned for most applications, but workloads such as cryptocurrency mining utilize it to the fullest. Overall, we conclude that the effectiveness of software in utilizing the underlying hardware has improved, but still has scope for optimizations.

Index Terms—Benchmarking, Multi-Core, Desktop Applications, Thread-Level Parallelism, GPU utilization, Virtual Reality, Cryptocurrency Mining, Characterization

I. INTRODUCTION

Innovation in the domain of improving single-threaded performance hit a plateau in the early 21st century. Anticipating the end of Dennard scaling, which states that power density of a chip remains almost constant across technology nodes [10], the hardware industry swiftly pivoted towards multi-core processors. The post-Dennard scaling era is plagued by the problem of dark silicon, caused because improvements in cooling technology failed to keep up with the Thermal Design Power (TDP) requirements of newer technology node chips [11]. Today, the hardware world is experiencing a move to specialization, trading away silicon area for gains in energy efficiency [32] [37]. Modern systems are almost ubiquitously heterogeneous, involving a combination of the CPU with a GPU and/or fixed-function accelerators. Common desktop systems at homes and offices have 4-8 logical CPUs with a discrete GPU connected via PCI-Express.

User requirements are significantly diverse. Gamers, for example, require a high-end machine with an advanced GPU and efficient cooling. On the other hand, for a user who uses their system primarily to browse the web and watch videos, it is cost-inefficient to own a system with a high-end GPU. Thus, it is worthwhile to analyze the characteristics of commonly used applications tailored toward different users.

We repeat some of the experiments of an eight-year old work by Blake et al. [3], characterizing the parallelism exploited by software in desktop workstations. To analyze how a wide spectrum of commonly used desktop applications have evolved to utilize available hardware, we use the metrics of Thread Level Parallelism (TLP) and GPU utilization. Our application suite consists of a diverse choice of traditional desktop applications, such as web browsers, video authoring utilities, media players, as well as emerging applications, such as personal assistants, cryptocurrency miners and virtual reality (VR) games. In addition to the metrics mentioned earlier, we analyze the effect of core scaling and simultaneous multi-threading (SMT) on these applications.

This work attempts to answer the following questions:

- Have modern versions of legacy software, and newer software that have replaced them, been keeping up with advances in hardware technology?
- How well do contemporary and emerging applications utilize the parallelism in the underlying hardware?
- What is the impact of core scaling, SMT and the GPU on the performance of applications?

The rest of the paper is organized as follows. Section II introduces the parallelism analyses done 18 and 10 years ago [3] [13] [14], and emerging applications that have evolved from advancements in hardware since then. Section III describes the system used for benchmarking, the metrics we study, trace collection methodology and the automation technique used to obtain consistent measurements. Section IV details each testbench and Section V contains evaluation of trends in parallelism. Section VI presents work related to characterization of different workloads. We discuss key takeaways and suggestions for software developers to better harness the hardware in Section VII and conclude in Section VIII.

II. BACKGROUND AND MOTIVATION

This study provides an 18-year perspective on the evolution of parallelism in desktop workloads. In early 2000, when un-processors were prevalent, Flautner et al. [13] [14] evaluated...
the TLP of existing desktop applications on a symmetric multiprocessor (SMP) with 2-4 cores. The average TLP observed across all benchmarks was lower than 2 and only specific workloads, such as video encoding, benefited from more processing cores. However, a second processor improved the responsiveness of interactive applications. 10 years later, Blake et al. [3] presented a study of TLP for commercial desktop applications on an 8-core processor with SMT. They concluded that 2-3 processor cores were still more than sufficient for most applications and that the GPU was mostly underutilized.

Another 8 years have passed since then, and desktop machines have evolved into a combination of CPU, GPU, and fixed-function hardware. The prevalence of multiprocessors begs the question of how software developers have been catching up with the advancements in hardware. We analyze the TLP and GPU utilization of a wide variety of commonly used applications on a state-of-the-art desktop with 6 SMT cores. GPU utilization measures the average amount of GPU usage over time, and TLP characterizes the amount of concurrency with idle time factored out.

We also evaluate emerging workloads that have gained popularity in recent years, including virtual reality (VR) games, cryptocurrency miners, and personal assistants. The first commercial VR headset was not released until 2016, and currently there are more than 150 million active VR users worldwide [23]. Cryptocurrency mining has experienced tremendous growth over the past decade, reaching a total market capitalization of over 200 billion USD [1]. However, both the immersive gaming experience provided by VR and the computational complexity of cryptocurrency mining have non-trivial hardware requirements. For personal assistant applications, the user demands have been scaling since Apple introduced Siri in 2011 [19]. Although personal assistant applications rely heavily on datacenters to offload the complex part of the workload, it is worthwhile to explore how much parallelism is exploited by the work performed locally.

### III. Methodology

#### A. System Setup

Moore’s law has continued, albeit at a slower pace, led by incremental advances in manufacturing technology and hardware architecture over the past decade. The system used by Blake et al. [3] employed a dual-socket CPU with four 2.26 GHz 4-way out-of-order cores per socket, along with an 8 MB last-level cache and 6 GB of RAM. We built a desktop machine with state-of-the-art hardware components, representative of a high-end gaming rig by current standards. A dual-socket (SMP) with 2-4 cores. The processor, operating at 3.70 GHz with Turbo boost to 4.70 GHz, consists of six superscalar cores with a last-level cache of size 12 MB and 64 GB of RAM. Each core supports 2-way hyper-threading, providing us with a maximum of 12 logical cores. The processor is also equipped with an integrated graphics processor and specialized hardware blocks, such as Quick Sync Video (QSV) [21].

Despite the processor having an integrated GPU, we evaluate a discrete GPU, which is a more practical setup in desktop systems. The GTX 285, used by Blake et al., consists of 240 CUDA cores operating at 648 MHz [30]. This work uses the GTX 1080 Ti, which has 3584 CUDA cores (∼15× more), running at 1481 MHz (∼2× more) [29]. Some benchmarks are also tested with the GTX 680, which operates at 1006 MHz with 1536 cores, to evaluate the differences in performance and utilization between a high-end and a mid-end GPU [31].

Windows 10 is used in this work, as it supports a wide range of commercial applications that are commonly used by desktop consumers. More than 80% of the global operating systems market share for desktops comprises of Windows [36]. Besides, some applications in the benchmark suite, such as VR games, are only supported by Windows.

#### B. Metrics

The formula for TLP is shown in Equation [1] where \( c_i \) denotes the percentage of execution time when \( i \) threads are running simultaneously, in a system with \( n \) logical CPUs. \( c_0 \) represents the idle time in the application.

\[
TLP = \frac{\sum_{i=1}^{n} c_i t}{1 - c_0}
\]

(1)

The measurements are reflective of the application-level TLP, which measures the TLP of processes that pertain only to the application under consideration, in contrast to the system-wide TLP measured by Blake et al. [3] and Flautner et al. [13] [14]. This is because unlike system TLP, application TLP exposes the application behavior directly.

For GPU utilization, we consider the amount of time spent by work packets actually running over a period of time, where a packet is defined as a large collection of Application Programming Interface (API) calls packaged into a command stream. GPU utilization is measured by aggregating for all packets the ratio of packet running time to total time.

#### C. Trace Collection

Event Tracing for Windows (ETW) is a kernel-level tracing feature that allows logging of application-defined events. We use WlforETW [33], a wrapper around ETW, to collect Event Trace Log (ETL) traces, after ending unrelated background processes. The traces are then analyzed using the built-in Windows Performance Analyzer (WPA). Within WPA, we extract the relevant data columns from the CPU Usage (Precise) Timeline by CPU analysis for TLP and from the GPU Utilization (FM) analysis for GPU utilization. wpaexporter is used to automate the extraction of relevant data from WPA. Lastly, we use custom scripts to

<table>
<thead>
<tr>
<th>CPU</th>
<th>Intel Core i7-8700K, 3.70-4.70 GHz, 6 cores / 12 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics</td>
<td>NVIDIA GTX 1080 Ti, 1481 MHz, 3584 CUDA cores</td>
</tr>
<tr>
<td>RAM</td>
<td>64 GB (16 GB × 4) DDR4 @ 3200 MHz</td>
</tr>
<tr>
<td>Storage</td>
<td>2 TB (1 TB × 2) PCIe NVMe SSD</td>
</tr>
<tr>
<td>OS</td>
<td>Windows 10 Education Version 1803</td>
</tr>
</tbody>
</table>

**TABLE I**: Specifications of the benchmarking desktop system.
process the outputs of wpaexporter. Figure 1 summarizes the measurement workflow for collecting TLP and GPU data. We cross-validate the GPU data with those reported by WPA.

D. Testing Automation

While conducting multiple iterations of experiments with the same applications, it is crucial to maintain consistency across events such as mouse clicks and keystrokes. AutoIt is an automation language designed for Windows that can simulate keystrokes and mouse activities at a user-specified time. Testing automation mitigates the variations created by user interactions among different test iterations. For each application that can be automated by AutoIt, we construct an automation script that initiates the application and performs a carefully designed sequence of mouse and keyboard activities.

We inspect the effect of AutoIt on TLP and GPU utilization by comparing experiments on an application with a high amount of user interactions (PowerDirector) and one with non-trivial GPU utilization (VLC Media Player). The TLP for manual testing was 3.3% smaller than with automatic testing. The GPU utilization is 2.4% lower with AutoIt than when performed manually. This demonstrates that the use of AutoIt does not significantly distort the results in this work.

E. Manual Testing

Some applications accept user inputs that cannot be precisely reproduced by automation tools. Personal assistants, for instance, require audio inputs. So, we apply a fixed sequence of requests and questions with strict timing constraints and use the same person’s voice for all test iterations.

VR games have a diverse set of inputs to track the position and action of the player, such as signals from motion sensors and controllers. Besides, VR game scenes vary in a non-deterministic manner, requiring players to take different actions to survive in the game. Repeating the same action can cause larger variation in each testing iteration. Therefore, it is both infeasible and undesirable to provide the exact same inputs to VR games. To maximize the similarity between different test iterations, we choose the campaign mode for each game to create similar scenarios and survive for a predefined amount of time. For games without a campaign mode, we start from a fixed checkpoint and perform similar actions each time.

IV. Benchmarks

We construct a suite of common desktop applications based on their popularity among users and create as much overlap as possible with the benchmarks used in [3, 13, 14] to understand how software has adapted to advancements in hardware. The version number of each application is specified in Table III.

A. Image Authoring

Image authoring exhibits a high degree of parallelism during rendering. We experiment on a 2D image editor, a 2D/3D CAD application, and a 3D animation modeling application.

Adobe Photoshop: Photoshop is an advanced graphics editing and design tool. 5 custom filters are applied serially on a 100 mega-pixel photograph.

Autodesk AutoCAD: AutoCAD is a CAD tool used in architecture, engineering, etc. We import a floorplan, pan, zoom, draw, fillet the edges, mirror and enter text.

Autodesk Maya 3D: Maya is a 3D graphics software used for making animated movies, 3D modeling, etc. We open up a complex model, smooth, perform a software render with raytracing followed by a hardware render with fog, motion blur and anti-aliasing, rotate, pan and zoom the camera.

B. Office

Office productivity applications are ubiquitous. A suite of applications that aim to help office tasks are included.

Adobe Acrobat Pro: Acrobat Pro is designed for PDF editing. We scan documents, combine different files into one PDF, manipulate the pages, insert links, watermarks and signatures, and export the PDF into slides.

Microsoft Excel: Excel is a popular spreadsheet editor. We open a spreadsheet containing 1 million rows of text and numbers, copy multiple columns, zoom, pan, change layout, compute means, sort and filter rows, and plot a histogram.

Microsoft Outlook: Outlook is a desktop email client by Microsoft. We compose a new email, save and delete the draft, search and reply to a specific email, delete and recover an email from the inbox, move an email in and out of the junk folder, and finally categorize emails and do a filter operation.

Microsoft PowerPoint: PowerPoint is a tool that lets users create slides and deliver presentations. We open a complex template, add bullet points, format them, add shapes and animate them, add a picture, scale and rotate it and finally create a table and populate it with text and numbers.

C. Multimedia

Despite the booming popularity of online video streaming/sharing, multimedia players remain widely in use. This work analyzes QuickTime Player, Windows Media Player and VLC Media Player. For each application, a 480p and a 1080p version of the same video are played in succession.
D. Video Authoring and Transcoding

Video authoring and transcoding utilities enable users to edit videos and convert high definition videos into various formats. GPUs are usually used in such workloads to assist in performing highly parallel, compute-intensive tasks.

**CyberLink PowerDirector**: PowerDirector is a video editing software that allows composing and editing video clips. We import three clips in PowerDirector, add transitions, titles, color correction and render it with and without CUDA support.

**Adobe Premiere Pro**: Premiere Pro is a video editor geared for professionals. The same operations for PowerDirector are repeated with slight differences in filters and transitions.

**HandBrake**: HandBrake is an open-source video transcoding. We use it to transcode part of a 3840×2160 resolution high-quality video at 50 frames per second (FPS) to a 1920×1080 resolution MP4 video at 30 FPS.

**WinX HD Video Converter**: WinX is a video transcoding utility supporting GPU acceleration. We repeat the same test sequences that were used for HandBrake.

E. Web Browsing

Popular web browsers, **Chrome**, **Firefox**, and **Edge** are chosen for the benchmark suite. Chrome and Firefox take up almost 80% of the desktop web browser market and Edge is a built-in browser for Windows 10 that succeeded Internet Explorer. We perform similar tests as in the prior work by Blake et al. [3] to study the impact of improved software.

For the first two tests, we watch a random video on YouTube, then browse ESPN, CNN and BestBuy, and finally play a flash game. We use a different tab for each website in the first test and a single tab to browse all websites in the second test. In the third test, we browse ESPN, which has plenty of active content (e.g. ads, videos, etc.) and in the final test, we browse Wikipedia, which has little active content.

F. Virtual Reality (VR) Gaming

VR headsets provide players with immersive gaming experience that traditional 3D games cannot offer. The diverse set of sensors and advanced rendering techniques necessary for VR gaming impose non-trivial pressure on hardware. We select a collection of VR games that have a large player community, high user ratings and intensive graphics, and use the highest settings for all games to stress the GPU to the maximum.

**Arizona Sunshine**: We play the game in single-player Horde mode, surviving multiple waves of zombies.

**Fallout 4**: We continue from a saved checkpoint where the character has escaped from the nuclear fallout shelter.

**Serious Sam 3**: We play the game in survival mode, and due to the difficulty in surviving continuously for 3 minutes, we play through after getting killed and respawned.

**Space Pirate Trainer**: We play the game in “old school” mode that involves surviving multiple waves of pirate bots.

**Project CARS 2**: We start a quick race with the default car and track and race 1-2 laps with multiple other drivers.

**RAW Data**: We play the game in campaign mode, surviving waves of attacking humanoid robots and protecting an object.

G. Cryptocurrency Mining

Cryptocurrency miners validate transactions by computing on blockchains. The benchmark suite encompasses 4 miners – **Bitcoin Miner** and **EasyMiner** for Bitcoin and **PhoenixMiner** and **Windows Ethereum Miner** for Ethereum. Each one is run for a predefined amount of time.

H. Personal Assistant

The prevalence of machine learning and natural language processing has given rise to personal assistant applications. Apart from **Cortana**, which is built-in for Windows, we also analyze **Braina**, a multi-functional interactive AI software. The tested queries cover requests for daily news, weather forecast, alarm/reminder management and questions about general knowledge, word definitions and simple math problems.

V. Evaluation

Table III summarizes the TLP of the 6-core processor with SMT enabled and the GPU utilization of the GTX 1080 Ti for all the applications. The “execution time” column illustrates the amount of time when 0, 1,..., 12 logical cores are active simultaneously. The color of the heat map region corresponding to $c_i$ illustrates the percentage of execution time when $i$ threads are executed simultaneously. The “TLP” column shows the average and standard deviation of the TLP derived from 3 test iterations (with same duration) for each application. Similarly, the “GPU utilization” column contains the average and standard deviation of the measured GPU utilization. Based on the low standard deviations, we conclude that our experimental results are consistent. The last two columns in Table III present the average TLP and GPU utilization for each category, respectively. In summary, every application exploits parallelism to some extent, with a few applications showing more concurrency than others. For categories like office, multimedia playback, personal assistant and web browsing, the degree of parallelism exploited is quite low, as concluded from the average TLP of around 2. VR gaming displays moderate concurrency, with an average TLP ranging from 2 to 4. The TLP is expected to be similar within a category, but some categories are exceptions, including image authoring, video authoring, and cryptocurrency mining. There also exist applications that effectively utilize most of the available cores, e.g. applications for video transcoding exhibit an average TLP over 9. Overall, the average TLP across all benchmarks is 3.1, where 6 out of 30 applications have an average TLP higher than 4.

---

In general, achieving an average GPU utilization over 90%. VR games and cryptocurrency applications exhibit moderate GPU usage. Video authoring and transcoding applications show lower GPU utilization. This can be attributed to the GPU being underutilized under most circumstances, except for graphic-intensive and cryptocurrency mining applications.

### B. Evolution of Concurrency

The values of GPU utilization are lower than 10% for most applications. Video authoring and transcoding applications exhibit moderate GPU usage. VR games and cryptocurrency miners, however, show significant utilization of the GPU, achieving an average GPU utilization over 90%. In general, the GPU is underutilized under most circumstances, except for graphic-intensive and cryptocurrency mining applications.

#### 1) Core Scaling

Experiments are performed on the processor with 4, 8 and 12 active logical cores to analyze the TLP behavior when more resources are available. Figure 4 shows the TLP characteristics of the application with the highest average TLP in each category. For applications exhibiting a low degree of parallelism, including Chrome, VLC, Excel and Cortana, the TLP is tied to 2, since there is not much parallelism to exploit. On the contrary, EasyMiner assigns independent threads to each of the logical cores, leading to the TLP scaling linearly with the number of active cores. The TLP of other applications scale sub-linearly, based on the amount of parallel work. The measured TLP, clearly showing these trends, is shown in Figures 5-7.

HandBrake presents a highly parallel workload. The TLP is mostly at its maximum, but drops periodically due to serialization. Increasing the core count results in more fluctuations in instantaneous TLP. This is consistent with the HandBrake documentation [13], which states that it can scale up to 6 cores and presents diminishing returns beyond that. The frame rate also scales up with the number of cores and the time spent on transcoding the same length of video decreases in proportion.

Photoshop involves a significant amount of user interaction, which leads to a non-trivial amount of idle time waiting for user inputs. However, as mentioned in Section II idle time...
is not considered while calculating average TLP. User input processing exhibits a low TLP, whereas the TLP of filter rendering scales linearly with the number of active cores and can reach a maximum of 12 when all cores are enabled. The runtime is bottlenecked by user response time, so it gets smaller with increasing number of cores, but is still far from linear scaling, in compliance with Amdahl’s law [2].

Project CARS 2, similar to Photoshop, involves significant user interaction and is also continuously processing sensor data and rendering graphics. The instantaneous TLP reaches the maximum in bursts, but mostly remains between 2 and 6. The average TLP saturates around 5, due to serialized work.

Overall, the performance gains provided by increasing the amount of hardware resources heavily depends on the volume of parallel tasks. Performance of parallel workloads scales up with growing number of cores, leading to shorter execution times for the same tasks. Interactive benchmarks, with a significant amount of parallelism, can also benefit from more processor cores, though the processing of user interactions is usually serialized. However, non-bursty workloads limited by low TLP do not benefit much from extra processor cores. Therefore, further exploitation of parallelism is necessary to take advantage of the available hardware resources.
2) Simultaneous Multi-Threading (SMT): SMT is aimed at exploiting more parallelism and improving functional unit (FU) utilization in CPUs by allowing a physical core to run multiple threads simultaneously \[38\]. Prior work states that SMT boosts performance as threads bring useful data on-chip for each other \[3\]. However, Figure 8 shows that the transcode rates of both HandBrake and WinX decrease when SMT is enabled. This is because SMT helps when data is reused among threads within a physical core, but it also limits the hardware resources (e.g. functional units) available for each thread. Statistics from Intel\'s VTune Amplifier show that for HandBrake, enabling SMT causes a decrease in Last-Level Cache (LLC) misses and the time spent waiting on main memory, as threads fetch data for one another \[22\]. However, the percentage of time spent by a core stalled on the L1 cache, without missing in it, increases from 5.3% to 10.7%. This is explained by thread contention for computation resources within a physical core. For example, an old store may be waiting for available functional units to resolve its address and blocking a newer load. The performance degradation due to resource contention overcomes the benefits from lower pressure on the LLC/off-chip memory. It also hurts the utilization of GPU, leading to a non-negligible reduction in transcode rate. This implies that as software exploits parallelism by distributing computation among threads, SMT may have no or even detrimental impact on performance.

D. GPU Analysis

The dramatic breakthrough in GPU hardware over the past couple of decades has made it crucial to understand how GPUs are used to effectively assist compute-intensive tasks and whether GPUs are exploited to their full potential.

1) GPU Offloading: The performance and GPU utilization of HandBrake and WinX with the high-end GTX 1080 Ti and the mid-end GTX 680 are shown in Figure 8. HandBrake does not offload tasks to the GPU, so the utilization stays below 1%, regardless of the number of active cores and GPU settings. WinX, on the other hand, supports hardware acceleration with CUDA/NVENC. The transcode rates for different GPUs are almost the same (the plots for GTX 680 are omitted as they overlap with those for GTX 1080 Ti). In order to achieve similar performance, the GTX 680, which is inferior to the GTX 1080 Ti, harnesses a much higher utilization. If we use an even lower-end GPU, we expect the GPU utilization to increase, and the performance will start to degrade after the GPU utilization saturates at the maximum.

2) GPU Utilization: As shown in Table II, the GPU is under-utilized for most of the applications, which is possibly because the computational power of the GPU greatly exceeds what is demanded from it. The GPU indeed executes substantial tasks in various applications, such as hardware rendering in Maya video export in PowerDirector, yet both exhibit GPU utilizations lower than 10%. Even for WinX Video Converter, which uses CUDA/NVENC in the GPU for transcoding, the average GPU utilization is 13.6%. On the other hand, there are applications that utilize the GPU much more efficiently, such as VR games and cryptocurrency miners. We measure the GPU utilization of the mid-end GPU for video-related applications and cryptocurrency miners, as these use the GPU more than the others, and compare them to the utilization of the high-end GPU. Most applications see a notable improvement in utilization, except for cryptocurrency mining. Both GPUs show utilizations of up to 100% for

<table>
<thead>
<tr>
<th>Logical Cores</th>
<th>Transcode Rate (FPS)</th>
<th>GPU Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No GPU</td>
<td>GPU</td>
<td>No GPU</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>28</td>
<td>37</td>
</tr>
</tbody>
</table>

TABLE III: Transcode rate, TLP, and GPU utilization of WinX with and without NVIDIA CUDA/NVENC. Enabling the GPU improves the transcode rate and lowers the TLP.
GPU Utilization

Fig. 10: GPU utilization of GTX 680 and 1080 Ti for applications that show substantial use of GPU. VR is excluded as it requires a GPU better than GTX 970. PhoenixMiner does not support GTX 680.

Bitcoin Miner and EasyMiner, but as expected, the hash rate of GTX 680 is at least 2× lower despite the assistance of the CPU. Windows Ethereum Miner, however, has a higher GPU utilization with the superior GPU, since NVIDIA’s Kepler architecture in GTX 680, released before the prevalence of cryptocurrency, is not optimized to run mining workloads.

In summary, a mid-end GPU is sufficient for most applications, including video-editors/transcoders. However, for applications, such as VR gaming and mining, that perform intensive computations on the GPU, a high-end GPU is indispensable, as the mid-end GPU causes a significant performance loss.

E. Web Browsing Workload Analysis

Among the web browsers, Chrome and Firefox share common features. While the number of processes created by Chrome is 10× larger than that by Firefox, Firefox uses much more resources in GPU to match the performance. Edge claims to have the best power efficiency, with Chrome and Firefox consuming 36% and 53% more power respectively [40], which is consistent with its low TLP and GPU utilization.

The TLP and GPU utilization of the web browsing test-bench are illustrated in Figure 11. The tests using multiple tabs have similar or higher TLP compared to those using a single tab, which is in contradiction to the results from Blake et al. [3]. In the past, web browsers used to run the entire application in a single process, and the overhead of garbage collection when a user navigates to another website resulted in higher TLP for single-tab tests. Current web browsers use multi-process models to separate websites from each other and the browser itself, so that web contents are loaded in parallel to improve responsiveness. This is also to prevent failures in one webpage’s content from crashing the entire browser [7][27]. Inactive tabs run as background processes in the system. Web browsing using multiple tabs spawns more processes and threads than when using a single tab, resulting in higher TLP. However, the increase is not significant, because browsers constantly throttle inactive tabs after a certain amount of time [8]. Chrome generates the most number of processes and shows the least difference in the number of processes as well as TLP between the two tests. The overhead of garbage collection is also reduced, as it is scheduled to take place during idle time to avoid degradation of user experience [9].

In terms of the ESPN tests, Chrome attains the highest TLP, while Firefox and Edge do not exhibit much difference in TLP. Chrome generally creates a rendering process for each instance of the website, and the large amount of active content in ESPN makes Chrome spawn more processes for webpage rendering, leading to a higher TLP. Firefox and Edge, on the other hand, do not show any apparent increase in the number of processes. All web browsers use more GPU while rendering ESPN, suggesting that graphic-intensive work is offloaded to the GPU when possible, as expected.

Overall, web browsers have shown improvements in exploiting parallelism over the last two decades. Although TLP is bottlenecked by waiting for and processing user interaction, the improved parallelism enhances user experience in terms of both responsiveness and stability of web browsers.

F. Virtual Reality Workload Analysis

Oculus Rift and HTC Vive were released in 2016, and HTC Vive Pro launched in 2018. As shown in Figure 12, Rift achieves the highest TLP, especially for graphic-intensive games like Project CARS and Fallout 4. Vive and Vive Pro have almost the same TLP. Furthermore, Rift achieves the most stable frame rate of the three headsets (Figure 13). The specified frame rate for all the headsets is 90 FPS, but if only 4 logical cores are available, the actual frame rate of Rift is clamped to 45 FPS due to asynchronous space warp (ASW). ASW compromises the frame rate when the system cannot handle rendering at the full rate, to lower the minimum system requirements for the headset. This is consistent with the reduction in both TLP and GPU utilization (Figure 7). Vive and Vive Pro, instead, apply asynchronous reprojection to improve user experience. This technique pushes the GPU to render at 90 FPS, and inserts an adjusted frame when the GPU fails to render the next frame in time. So, with 4 logical cores, the frame rate oscillates between 90 and 45 FPS, and only slight variations appear in the TLP and GPU utilization.

The GPU utilization correlates with the resolution of the headset. For all games except Fallout 4, Vive Pro, which has the best resolution, achieves the highest GPU utilization. Rift and Vive have the same resolution and show comparable GPU utilizations. Fallout 4 exhibits a different trend in hardware utilization than the other games. The GPU utilization for Vive Pro is the lowest, and a lower frame rate for Vive Pro is observed in the game.

VI. RELATED WORK

Prior work has explored characterizing simulated as well as real systems. Eyerman and Eeckhout [12] evaluated a variety of multi-core systems to find the optimal design with limited hardware resources. Lorenzon et al. [24] investigated the TLP and energy consumption of Application Programming Interfaces (APIs) on embedded systems and general purpose
systems. Our work analyzes commercial applications on a real desktop machine, allowing us to obtain realistic data.

Plenty of hardware reviews on VR headsets, cryptocurrency mining and desktops are available through tech channels. They measure gaming performance through frame rate and mining performance through hash rate. The study done by Magaki et al. illustrated that ASICs have better energy and cost efficiency over GPU for mining. This work does an analysis from a parallelism perspective and evaluates TLP and GPU utilization.

Extensive characterization work has also been done on mobile devices. Gao et al. studied how multi-core mobile devices are utilized by common mobile apps. Chen et al. characterized mobile augmented reality applications from the system and architecture perspectives.

**VII. DISCUSSION**

TLP and GPU utilization can act as useful guidelines for end-users on the amount of hardware resources to invest. A key takeaway from this work is that employing many processing cores and a high-end GPU does not always bring benefits. For users who primarily spend time online or on office applications, 2-3 cores are sufficient to achieve maximal performance. For professional users who use their desktops for video transcoding or image editing, performance scales roughly linearly with increasing number of cores. For gaming and cryptocurrency mining, a better GPU leads to much better performance than a CPU with a large core count.

Multi-core scaling has hit a plateau, and software developers are now, more than ever, expected to write hardware-aware programs. This is because TLP and GPU utilization are not fundamental to an application, but highly dependent on how they are implemented in software. Based on our 18-year perspective analysis, we discuss potential areas where software can be improved to further exploit parallelism.

- Applications exhibiting complementary TLP characteristics can be scheduled to execute concurrently to achieve best utilization of the processor. For example, HandBrake exhibits high TLP with short periods of TLP drop. The OS could schedule another task during troughs in TLP, thus trading off fairness for better overall utilization.
- The GPU can be further exploited to assist compute-intensive tasks. Although the GPU suffers from poor single-threaded/latency-sensitive performance, tasks that have lower priority or latency requirements could be offloaded to the GPU. For example, when a Photoshop user selects a blur filter, the system can speculate the next task to be blur filter rendering and the core can start fetching off-chip data locally, while the user is specifying filter configurations.
- As the trend of innovation shifts further away from parallelism into heterogeneity, future software could offload kernels within an application to dedicated hardware/accelerators that execute the kernel most efficiently.

**VIII. CONCLUSION**

Major advancements have taken place in desktop hardware over the past decade. In this work, we analyzed the TLP, GPU utilization, and effects of core scaling and SMT, on traditional and emerging applications commonly used in contemporary desktop systems. Our results showed that software has improved to take advantage of the parallelism available in the hardware compared to the work in 2010. Noticeable increases were seen in many applications, including those reputed for effective utilization of processor cores like HandBrake and Photoshop. For applications with slight changes in TLP over the past 18 years, efforts for exploiting available parallelism were exhibited by them achieving high instantaneous TLP during execution. For example, Excel spent 3.7% of time using the maximum number of available logical cores concurrently, and web browsers have shifted from single-process models to multi-process models, resulting in better responsiveness and reliability. Emerging applications also demonstrated good utilization of hardware resources. The average TLP of VR gaming is twice that of traditional 3D gaming, and cryptocurrency miners involving CPU mining have a TLP higher than that of over 80% of the benchmarks. In addition, SMT is beneficial when threads running on the same physical core work on the same data with sufficient computation resources, else it becomes detrimental for performance.
On the other hand, overall GPU utilization was lower than that observed in 2010. This showed that the improvements in the amount of available resources in the GPU has been growing at a faster pace than improvements in the parallelism harnessed by software. However, emerging workloads, e.g., VR games and cryptocurrency miners, exhibited great potential, as they fully exploited the computation power of the GPU.

In conclusion, appreciable progress has been made by software in exploiting parallelism. However, there is still sufficient scope for software to further improve hardware utilization.

REFERENCES


