March 2023

Mechanism for Balancing of GPU Usage Between AI Queue Depth and Graphical Fidelity

Chris Schneider

Nic Watson

Mechanism for Balancing of GPU Usage Between AI Queue Depth and Graphical Fidelity

ABSTRACT

Concurrent application demands that exceed available GPU resources can degrade performance and worsen the user experience. However, there are no easy mechanisms for lay users to tailor application performance and control GPU resource allocation. This disclosure describes techniques that enable users to specify preferences for prioritizing computational tasks performed via a GPU via conventional user interface mechanisms. Scheduling and switching workloads and batches of the requested computations on the GPU is performed according to user-specified preferences. Users can leverage the preference settings for managing performance tradeoffs between application functionality, balancing resource allocation when multitasking, troubleshooting potential GPU-related problems, etc. The techniques can enable users to save time and effort in troubleshooting GPU-related problems and can save manufacturers the cost of handling return and replacement of defective hardware.

KEYWORDS

- Graphics Processing Unit (GPU)
- Computationally intensive tasks
- Artificial Intelligence (AI)
- Graphical fidelity
- Queue depth
- Task scheduler
- Resource allocation
- Task hierarchy

BACKGROUND

Many devices include specialized chips known as graphics processing units (GPUs) or similar components or cards. These chips are used for achieving high performance by supporting the central processing unit (CPU) in performing tasks that involve specialized complex calculations, such as those needed for rendering high quality graphics. Apart from graphical tasks, GPUs can additionally be employed for other tasks with heavy computational requirements, such as applications with artificial intelligence (AI) capabilities, proof-of-work calculations in technologies such as blockchain, etc. Computations requested to be performed via the GPU are typically scheduled and completed as workloads or batches based on a hierarchy. The hierarchy can be accessed through the use of flags and other technical settings to reserve specific GPU components for specific workflows. Most GPUs additionally include specific types of hardware components for specific functions, such as ray tracing. Such components are typically employed for narrow sets of operations.

GPUs draw large amounts of power to achieve their high computational performances. Yet, users who wish to favor power savings over computational performance cannot currently specify that the GPU and/or certain components of the graphics card not be used for specific types of computations or specific components of specific applications.

Some applications include multiple computationally heavy aspects that make use of the GPU. In some cases, multiple applications that use the GPU may be running concurrently. In such cases, the concurrent application demands for the GPU can exceed available GPU resources, thus degrading performance and worsening the user experience (UX). For instance, an immersive virtual reality (VR) application that uses the GPU for AI calculations and for rendering the environment at high fidelity may exceed available GPU resources and require

choosing between prioritizing AI or graphics calculations, worsening the aspect that receives lower priority. Alternatively, both aspects can be slightly degraded instead of having a larger negative impact on only a single aspect.

In the above example of an immersive VR application that exceeds available GPU resources, some users may prefer to sacrifice graphical fidelity or framerate in favor of maintaining the quality of AI-based immersive interactions that can benefit from the GPU allocating greater queue depths and memory capacity for these purposes. In contrast, other users may desire photorealistic graphics at the expense of somewhat degraded AI-based immersive interactions. Yet, when the demand for GPU resources exceeds capacity, the manner in which the underlying application components are queued and served is typically determined automatically via relevant GPU task switching mechanisms. Parameters of the GPU task switching operations can be adjusted via a few mechanisms, such as configuration files, command line arguments, and other highly technical approaches. However, the technical nature of such approaches makes them non-comprehensive and unusable for lay users who wish to tailor their experience by specifying which aspects of the UX they value more.

DESCRIPTION

This disclosure describes techniques that provide easy to use mechanisms for users to specify preferences for prioritizing computational tasks performed via a GPU when concurrent computational demands from applications exceed available GPU resources. For example, users can indicate that they favor GPU use for AI over graphics in situations that require a tradeoff between these two aspects of an application or can specify the maximum proportion of GPU resources that can be allocated to a given task when demand for GPU resources exceeds availability. Users can readily specify such preferences without requiring specialized technical

approaches via any suitable, commonly known user interface (UI) mechanisms, such as toggles, checkboxes, sliders, dropdowns, etc.

Whenever application demands for GPU resources exceed available capacity, scheduling and switching workloads and batches of the requested computations can be performed according to the user-specified preferences. As appropriate, the task scheduling based on user preferences can be achieved by reserving specific GPU areas or components for specific workflows. Apart from specifying preferences for desired UX within applications, users can leverage the settings for diagnostic purposes to identify potentially defective GPU components.

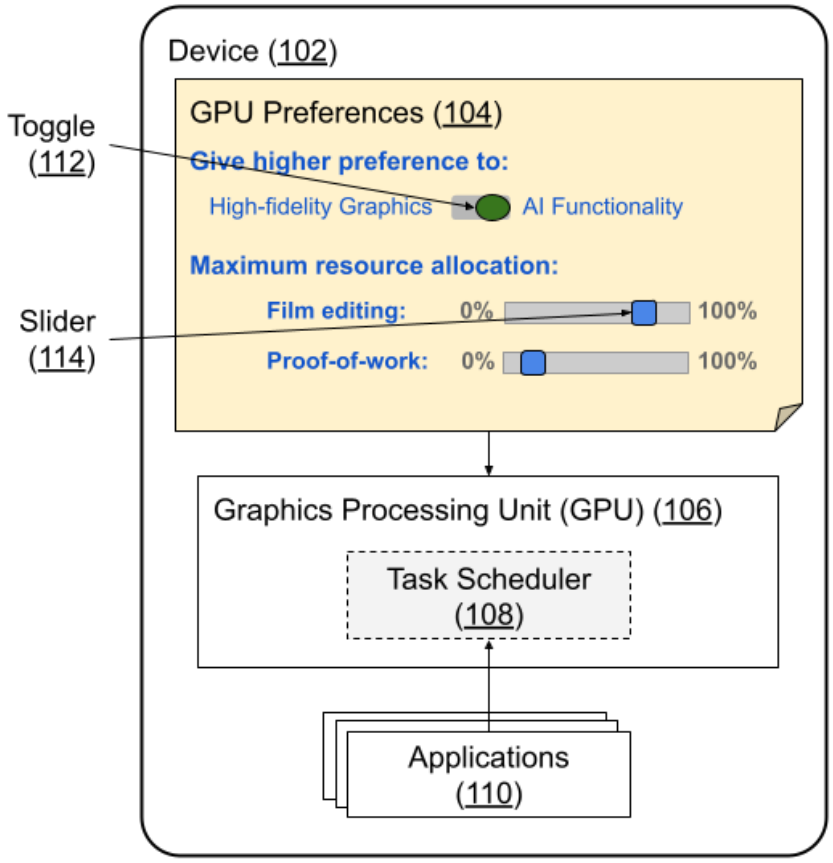

Fig. 1: UI mechanisms for specifying preferred GPU allocation for specific tasks

Fig. 1 shows an example of operational implementation of the techniques described in this disclosure. A user can specify preferences for GPU use (104) on a device (102) via various UI mechanisms. As shown in Fig. 1, the user can indicate the preferred tradeoff between graphics and AI functionality via a toggle (112), selectively disable individual GPU components, or indicate the maximum proportion of the GPU allocation when sharing the GPU for specific tasks via corresponding sliders (114). The user preferences are relayed to the task scheduler (108) within the GPU to determine optimal task allocation and scheduling for various applications (110) for which GPU resources are needed concurrently.

Users can employ the preference settings as illustrated above for a variety of purposes such as managing performance tradeoffs between application functionality, balancing resource allocation when multitasking, troubleshooting potential GPU-related problems, etc. For example, a user who wishes to see non-playing characters (NPC) with real-time reflections can set the toggle to favor AI over graphics rendering. The GPU can allocate more memory and compute time to the AI computations by increasing the queue depth available of the decisions of the NPCs in the game.

As another example, a user who is working on film editing in the foreground while also performing an AI compute workload in the background can set corresponding sliders to specify that at most 20% of the available GPU computational resources are to be provided to the background task whenever the available GPU resources exceed the combined demand from the applications. As yet another example, a user experiencing fractured graphics when playing a game can record their interactions with a game until the issue surfaces. By replaying the issue and automating the slider to selectively disable each stream manager or processor through a

series of tests, and by lowering and increasing the voltage, automated testing of hardware issues can be done reducing return merchandise authorization (RMA) costs and diagnostic times.

The niche function of the more specialized parts of the GPU dedicated to a narrow set of operations can make user control unnecessary or irrelevant. Therefore, such GPU components can be excluded from the user preference specification UI. The maximum proportion of GPU resources for a given task as specified by the users can be used to prioritize rather than reserve GPU resources. As a result, when there are no current tasks of the type for which a maximum proportion is specified, GPU resources that would have been used by that type of task can instead be allocated to other tasks until the specific type of task appears in the scheduling requests. Alternatively, the proportion can be reserved to avoid jagged performance experiences.

The techniques described herein can be implemented within any devices that contain a GPU. Such devices include computers, smartphones, game consoles, smart TVs, VR headsets, etc. Moreover, the implementation can support any type of GPU chip or card. The techniques can be applied to enable users to specify preferences for any GPU-reliant tasks, such as VR, games, AI, blockchain applications, scientific visualization, cloud computing workloads, etc. Implementation of the techniques provides greater control over the user experience and performance of applications that use the capabilities of the device GPU, thus helping users optimize their experience without requiring in-depth technical knowledge. In addition, the techniques can enable users to save time and effort in troubleshooting GPU-related problems, thus helping manufacturers avoid the cost of handling return and replacement of defective hardware.

CONCLUSION

This disclosure describes techniques that enable users to specify preferences for prioritizing computational tasks performed via a GPU via conventional user interface mechanisms. Scheduling and switching workloads and batches of the requested computations on the GPU is performed according to user-specified preferences. Users can leverage the preference settings for managing performance tradeoffs between application functionality, balancing resource allocation when multitasking, troubleshooting potential GPU-related problems, etc. The techniques can enable users to save time and effort in troubleshooting GPU-related problems and can save manufacturers the cost of handling return and replacement of defective hardware.

REFERENCES

1. Business Wire. “Run:AI Eliminates GPU Resource Allocation Issues For AI Workloads” available online at https://ai-techpark.com/runai-eliminates-gpu-resource-allocation-issues-for-ai-workloads/ accessed February 27, 2023.
2. IBM. “Adding Disk, Memory, and CPU” available online at https://cloud.ibm.com/docs/databases-for-elasticsearch?topic=databases-for-elasticsearch-resources-scaling&interface=ui accessed February 27, 2023.