Performance Analysis of GPU Accelerators with Realizable Utilization of Computational Density

Justin W. Richardson, Alan D. George, Herman Lam
NSF Center for High-Performance Reconfigurable Computing (CHREC)
ECE Department, University of Florida, Gainesville, FL, USA
{richardson,george,hlam}@chrec.org

***Abstract*—With the rising number of application accelerators, developers are looking for ways to evaluate new and competing platforms quickly, fairly, and early in the development cycle. As high-performance computing (HPC) applications increase their demands on application acceleration platforms, graphics processing units (GPUs) provide a potential solution for many developers looking for increased performance. Device performance metrics, such as Computational Density (CD), provide a useful but limited starting point for device comparison. The authors developed the Realizable Utilization (RU) metric and methodology to quantify the discrepancy between theoretical device performance shown by CD and the performance developers can achieve. As the RU score increases, the application is achieving a larger percentage of the computational power the device can provide. The authors survey technical publications about GPUs and use this data to analyze the RU scores for several arithmetic application kernels that are frequently accelerated in GPUs. The RU concepts presented in this paper are a first step towards a formalized comparison framework for diverse devices such as CPUs, FPGAs, GPUs and other novel architectures. GPU kernels for matrix multiplication, matrix decomposition, and N-body simulations show RU scores ranging from almost 0% to approaching 99% depending on the application, but all kernel areas show a significant decrease in RU as the computational capacities increase. Additionally, the RU scores show the higher realized performance of the GeForce 8 Series GPUs versus newer GPU architectures. This paper shows that applications running on GPUs with higher computational density report significantly lower RU scores than more mature GPUs with lower computational density. This trend implies that while the raw performance available is still increasing with newer GPUs, the achieved performance is not keeping pace with the theoretical capacities of the devices.**

I. Introduction

As the computational demands of modern applications increase, many developers are looking for ways to accelerate their applications beyond the speed that conventional central processing units (CPUs) can provide. Commodity graphics processing units (GPUs) provide a possible option for accelerating application kernels in high-performance computing (HPC) environments. This growth of GPU technology has driven continual development in GPUs both in terms of hardware and software. As the feature size used for GPUs decreases and the device architectures become more complex, the computational power of GPUs is rapidly expanding, but even with this expansion of technology developers are not achieving the theoretical performance that GPUs are providing.

For applications on all types of hardware, including GPUs, many different factors limit the achieved application performance such as developer experience, available tools, application characteristics, architecture bottlenecks, etc. These varied factors make exhaustive device benchmarking for every application unrealistic. Even though the computational power and popularity of GPUs is increasing, arithmetic applications are not keeping pace with the performance theoretically available from the GPU-based accelerators. Device performance metrics provide insight into device capabilities, but do not account for any application-specific factors effecting performance.

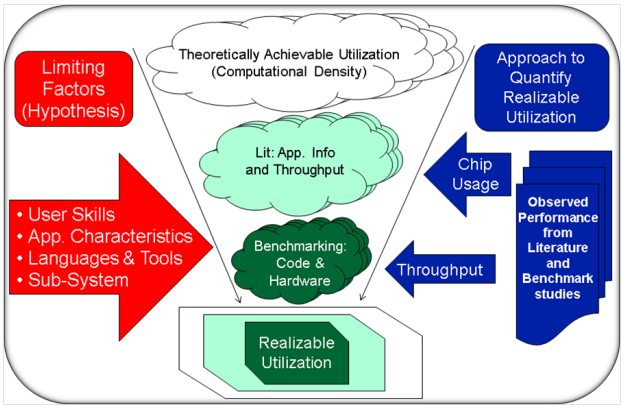

Fig. 1. Concept Diagram of Realizable Utilization

This paper outlines a methodology to use the Computational Density (CD) metric from [1] and data from technical publications to analyze and compare application accelerators. Figure 1 outlines the conceptual approach, by showing that the theoretical computational capacity, represented by CD, is reduced by various application and implementation dependant factors. This Realizable Utilization (RU) metric, which we introduce in this paper, formally quantifies the percentage of the available performance an application is achieving. Using the CD values, we analyze a set of matrix and n-body application kernels from published technical papers and observe an important trend in RU as the CD of GPUs increase.

II. Background

One of the most important aspects of accelerator comparison is having a fair and balanced set of metrics for the devices being compared. By necessity, these will be calculated in different ways for each device type. In [2], the authors outline

a computational-based metric, CD, for both fixed-logic devices (FLDs), such as GPUs and reconfigurable-logic devices (RLDs), such as field-programmable gate arrays (FPGAs). CD is a measure of a device's theoretical computational capacity. Further publications, [1], and [3], expand on this concept showing how this metric can be used across a wide range of devices and architectures. The developed CD metric forms a strong basis upon which the RU metric is built.

III. Realizable Utilization

This section defines and illustrates the RU concept, outlines how to use the CD metric as the basis for the RU metric, and demonstrates the insights that can be gained from using the RU methodology. The following sections show the RU concept applied to GPUs and CPUs and the results of the analysis.

A. RU Concepts

There are many factors that can reduce a device's performance including application characteristics, tools, and user experience. The concept of realizable utilization is a method to quantify the approximate difference between a device's theoretical performance and the actual performance a user can expect to achieve. Since benchmarking every device with every application is not practical, RU allows developers to estimate their application's projected performance on a particular device.

In Figure 1, illustrating the RU concept, the theoretical computational capacity, represented by CD, is reduced by various factors such as developer experience, tools used, application characteristics, etc. The application throughput from the technical data and benchmarks shows, for the specific platform, application, and implementation, the performance achieved.

RU starts with the CD metric representing the theoretical computational capacity of a device. Performance data is then collected from either scholarly publications or benchmarking experience (Figure 1). Data from technical papers is used to compare observed throughput with CD yielding the RU score. Benchmarking information requires more development time and effort, but because benchmarking can use more hardware and is closer to the desired application, it provides more accurate data. Using data acquired during benchmarking provides even more revealing RU scores when the developer tunes the application to the hardware or conversely tunes the hardware to the application, as in FPGAs.

B. Calculating CD

The first step in computing the RU for a given device is to find out what the CD value is for that device at the precision level of the application. To determine the single-precision floating-point (SPFP) CD for FLD and coarse-grained RLD devices, Equation 1 is used, where N_i is the number of floating point execution units or the number of floating point instructions that can be issued simultaneously of element type i, CPI_i is the average number of clock cycles per integer instruction for element type i (such as DSPs {digital signal processors}, ALUs {arithmetic logic units}, or LUT {lookup table} resources), and f is the operating frequency of the device. The subscript i represents the type of computational element within the device that is under analysis. The summation over i, in this equation, takes into account architectures that support single-instruction multiple-data (SIMD) instructions by including different types of computational components. We assume that only addition and multiplication operations are considered, and the number of parallel operations is maximized while keeping the number of additions and multiplications equal. When calculating the number of parallel operations supported by a device, we consider a hardware-supported multiply-accumulate operation as only one operation.

$$CD_{SPFP} = f \times \sum_i \frac{N_i}{CPI_i} \tag{1}$$

C. Calculating RU

Once CD is determined, the RU metric is calculated by dividing the observed throughput ($R_{throughput}$) in OPS (operations per second) by the relative CD of the device (Equation 2). The relative CD is the value of a device's CD multiplied by a scaling factor representing the fraction of the device used. The factor α is necessary because some applications have not been parallelized, and without adjusting the available CD the comparison between applications would not be as insightful.

$$U_{realized} = \frac{R_{throughput}}{\alpha \cdot CD_{device}} \tag{2}$$

The developer's knowledge of their application and its implementation allows α to be calculated easiest during benchmarking. When the information found in publication sources does not provide enough data to reliably determine α, then a ratio of 1 is assumed. This assumption is based on the hypothesis that most developers who are publishing their work and having it peer-reviewed will be trying to maximize performance of their application. If the application is not using all of the main resources of the device, the developer generally includes enough information to calculate α.

Since the CD value represents the theoretical maximum throughput, Equation 2 shows that the RU metric ($U_{realized}$) is bounded below by zero and above by one. While RU is a ratio, it is expressed as a percentage. From this alternate perspective, RU is the percentage of the theoretical performance of a device an application is achieving. This information provides insight for developers, not only before coding their application, but also during the development cycle.

D. Using RU

Once the RU metric has been calculated, it provides useful insight into various applications. RU provides developers a method of device comparison before choosing their platform. During the development cycle, it provides feedback on kernel development and optimization.

The applications of RU start during the device design process before a device is manufactured. Novel device architectures can be compared to similarly structured existing devices. The RU score then shows what application areas are most likely to fit well on a future device and that information can be incorporated into the device development process. By targeting an application area that will score well on a novel device, further resources can be freed for optimization and debugging.

Secondly, from a developer's standpoint RU can be used to help select an appropriate acceleration platform before significant costs are expended on cutting edge hardware. Applications with similar structure or kernels could be analyzed to see what platforms are making the most of the available resources. This insight helps to mitigate some of the risk with developing applications on new platforms and helps developers narrow the possibilities among many application accelerators.

Finally, developers can use RU to gain feedback while developing their applications. During the optimization cycle of development, it can be difficult to judge when the maximized performance from the optimizations has been reached, and how much more optimization performance you can expect. RU allows developers to compare their kernels or applications that are undergoing optimization to similar applications and kernels. The developer can then decide if additional performance is worth the time and cost.

IV. Results

Section III defined the concepts and outlined the methodology for computing the RU metric. This section reports the CD results of the survey of GPU and CPU devices, then applies the RU methodology to matrix multiplication, matrix decomposition, and n-body simulation kernels from scholarly publications and benchmarks and finally analyzes the RU metric results.

A. RU Results

For this paper, the authors searched for scholarly publications on GPU-based arithmetic benchmarks. The first step in reviewing the published data was to calculate the CD metric for each of the GPUs and CPUs found during our survey. This was done using the methodology presented in [1], [2], and [3].

Next, the publications and benchmarks were analyzed to discover which sources provided enough detail to calculate a RU score. Papers and books tended to report their results in one of two ways. The first type reported their application's achieved performance in terms of operations performed over a unit of time. Dividing the observed, or actual, operations by the units of time yielded the throughput needed to calculate RU. The second type reported their performance in terms of speedup over some baseline. When the performance is reported in terms of speedup, analysis of the algorithm is necessary. For well known algorithms, such as matrix multiply, the throughput value can be calculated if a finite processing time is given. From the algorithm, the necessary operations can be determined and dividing by the time shows the throughput needed for RU. If no definite time is given, or if the algorithm was too complex, then the paper was not suitable for the RU analysis without contacting the author for additional information.

This paper separates the RU results into three key kernel types, matrix multiplication, matrix decomposition, and n-body simulations. The results from each type of kernel are discussed separately and then collectively. Figure 2 shows the results for publications and benchmarks focused on matrix multiplication.

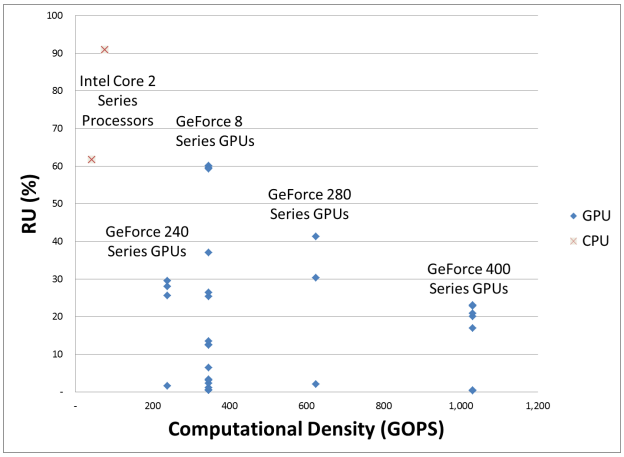

Fig. 2. Realized Utilization for Matrix Multiplication

For the matrix-multiplication kernels, Figure 2 shows the best RU scores for GPU devices are found in the GeForce 8 Series GPUs. Matrix-multiplication kernels were the most frequently found and due to the well known features of the matrix-multiplication operations they provide significant RU results. The peak matrix-multiplication RU scores highlight an obvious trend, significantly decreasing as the CD range increases. This trend implies that while on average the raw performance is still increasing with more powerful chips, the performance is not keeping pace with the theoretical capacities of the devices. A decreasing RU trend, with increasing CD, points to an application related issue limiting the achieved performance. In the CPU's case, the use of the advanced Math Kernel Library (MKL) from Intel shows they can achieve more of their computational capacities. The CPU's high RU scores and lower CD scores further strengthen the trend of reduced RU performance with higher computational capacity.

The highest scoring GPU device in the second kernel area, matrix decomposition (Figure 3), is the GeForce 8800 GTX with a RU score of 55.56%. This device has the same basic architecture as the Tesla C870, the highest scoring GPU in matrix-multiplication, and is one of the most frequently used devices in the study. It should be noted that the GeForce 8 Series devices are more mature than the newer GeForce 200 or 400 devices.

The matrix-decomposition RU scores further reveal the trend of decreasing RU scores with increasing device capacity and show similar performance patterns as matrix-

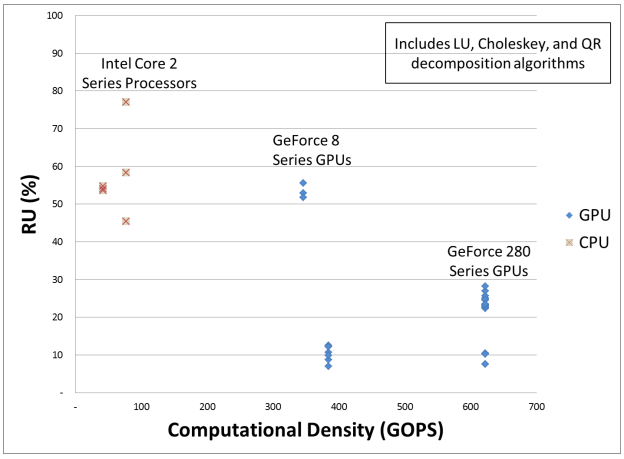

Fig. 3. Realizable Utilization for Matrix-Decomposition Kernels

multiplication. Once again we observe that low-CD CPUs can achieve more of their theoretical performance, but as the CD increases in GPUs the realized portion decreases.

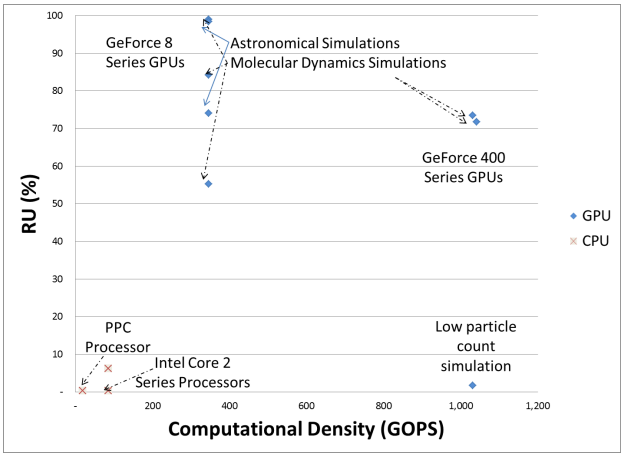

Fig. 4. Realized Utilization for N-Body Simulations

The final kernel area in this study is n-body simulations. Both molecular dynamics simulations as well as astrophysical simulations are included in this data set plotted in Figure 4. The data in n-body simulations shows the highest range of scores (1% to 99%) and includes the highest scores overall. These high scores point to a better fit between the application's development and the GPU device's hardware resources. Conversely, the CPUs didn't score as well in this kernel area and that shows that the n-body kernels shown are not as efficient on the CPU's architectures. While the devices with higher CD scored significantly better on RU than they did in other application areas, they still fell behind the older GeForce 8 Series GPUs. The data and references for all the figures are posted online for reference [4].

The general trend across all kernels shows the higher CD devices tend to have lower RU scores especially in matrix-based kernels. While overall raw performance is increasing as the device CD grows, the downward trend in RU shows that applications are not able, at this time, to capitalize fully on the added computational resources. This trend could be caused by many different issues including device tools, developer experience, application characteristics, architecture bottlenecks, and others. Determining which of these issues are most responsible is being considered for future work.

V. Conclusions

RU quantifies the difference between the theoretical performance of a device and the actual performance observed by developers. This work outlines the RU metric and methodology of using data from available technical publications and benchmarking experiments to quantify the effects of various factors limiting device performance. This RU score is the fraction of the theoretical capacity that the specific application is achieving on a specific device.

Our study focuses on three arithmetic kernels: matrix multiplication, matrix decomposition, and n-body simulations. Within these application areas, higher CD devices, such as larger GPUs, show significantly lower RU scores than their lower CD counterparts. While the relative performance is increasing, there is a significant reduction in the computational power that the applications are utilizing. Within the matrix-oriented kernels, the range of RU scores varied from nearly 0% to 60%. Further strengthening the trend, the CPU devices with lower CD, such as the Intel Core2 based processors, ranged in RU scores from 45% to 90% for matrix-based kernels. In contrast, n-body simulations show the highest RU scores for the GPU devices and the widest range reaching from 1% to almost 99%. The data shows that the GPU's architecture design is very effective at the computations in n-body simulations; however, it is not as good a fit for generic matrix-multiplication.

Future work on this topic will include adding additional devices, such as ARM processors and FPGAs, in addition to adding power and memory characteristics to the analysis. Advanced benchmarking studies will also explore the specific causes of different architecture's lower RU scores.

VI. Acknowledgements

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422.

References

[1] J. Williams, A. George, J. Richardson, K. Gosrani, C. Massie, and H. Lam, "Characterization of fixed and reconfigurable multi-core devices for application acceleration," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 3, no. 4, pp. 19:1 – 19:29, 2011.

[2] J. Williams, A. George, J. Richardson, K. Gosrani, and S. Suresh, "Computational density of fixed and reconfigurable multi-core devices for application acceleration," *Proc. of Reconfigurable Systems Summer Institute 2008 (RSSI)*, July 7-10, 2008.

[3] J. Williams, A. George, J. Richardson, K. Gosrani, and S. Suresh, "Fixed and reconfigurable multi-core device characterization for HPEC," *Proc. of High-Performance Embedded Computing Workshop (HPEC)*, Sep. 23-25, 2008.

[4] J. Richardson, A. George, and H. Lam, "Performance analysis of GPU accelerators with realizable utilization of computational density," http://www.hcs.ufl.edu/~richardson/Data/Papers/JR-SAAHPC-2012.pdf, June 2012.