Airavat: Improving Energy Efficiency of Heterogeneous Applications

Trinayan Baruah[1], Yifan Sun[1], Shi Dong[1], David Kaeli[1], and Norm Rubin[2]

[1]Department of Electrical and Computer Engineering,Northeastern University, Boston (MA), USA
[2]NVIDIA Research
{tbaruah,yifansun,shidong,kaeli}@ece.neu.edu, nrubin@nvidia.com

***Abstract*—An emerging class of applications attempt to make use of both the CPU and GPU in a heterogeneous system. The peak performance for these applications is achieved when both the CPU and GPU are used collaboratively. However, along with this increased gain in performance, power and energy management is a larger challenge. In this paper we address the issue of executing applications that utilize both the CPU and GPU in an energy efficient way. Towards this end, we propose a power management framework named Airavat that tunes the CPU, GPU and memory frequencies, synergestically, in order to improve the energy efficiency of collaborative CPU-GPU applications. Airavat uses machine learning-based prediction models, combined with feedback based Dynamic Voltage and Frequency Scaling to improve the energy efficiency of such applications. We demonstrate our framework on the NVIDIA Jetson TX1 and observe an improvement in terms of Energy Delay Product (EDP) by 24% with negligible performance loss.**

I. Introduction

Given their ability to provide data-parallel processing, Graphics Processing Units (GPUs) have been widely used to accelerate a broad range of applications. Heterogeneous architecture combine cores with different computing capabilities in a single system. This class of systems include big-little architectures and CPU-GPU integrated system. They are presently found on mobile devices, and based on their attractive power-performance characteristics, have great potential to be used in future exascale systems. Many of these systems are designed to share a common memory system, reducing the overhead of device-to-device transfers.

Traditional GPU workloads only utilize the host CPU for data transfer and kernel launching, resulting in low utilization of the CPU. Employing the computational power of the CPU, in parallel with executing on the GPU, improves performance for a variety of applications, and has been the target for a new class of workloads [1], [2].

However, when we integrate a CPU and a GPU on the same die, there are some additional issues to consider, including power management, thermal management, and hardware resources management of shared elements. Thermal management has been addressed by Paul et al, [3], while runtime management of shared hardware resources has been studied by Lee et al. [4]. Previous work on runtime power management [5] on AMD APUs mainly focused on improving the energy efficiency of applications that utilize only the GPU.

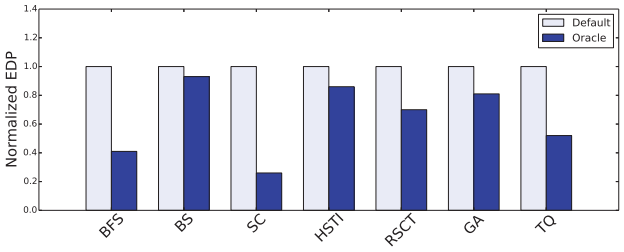
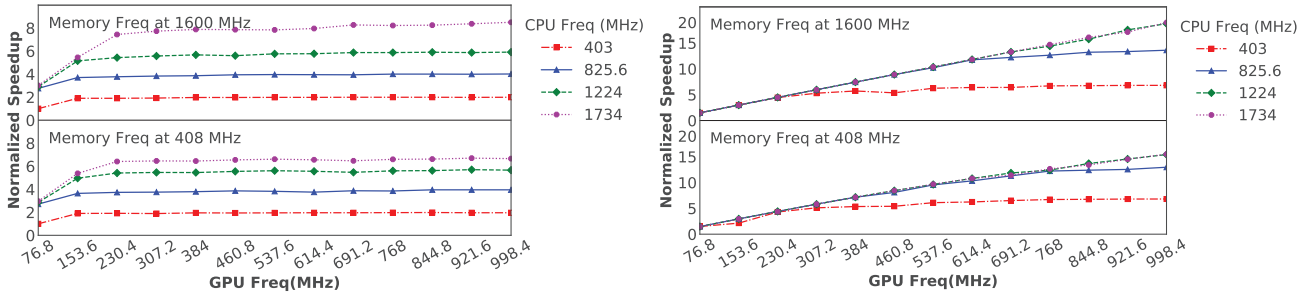


Fig. 1: EDP comparison of oracle vs. the baseline power manager for collaborative applications.

With the growing presence of applications that utilize both the CPU and GPU collaboratively, runtime power management for such applications on integrated heterogeneous architectures is becoming increasingly important. Unlike simple CPU-only or GPU-only applications, this class of collaborative applications can have different sets of tasks, or even portions of the same task, running concurrently on both the CPU and GPU. If we utilize both the CPU and GPU in parallel, this will result in greater power consumption and increased temperature of the chip. This suggests we need a power management framework to ensure these applications execute in an energy efficient manner, resulting in more power efficient execution. However, we also need to ensure that there is minimal performance degradation to the system, otherwise energy usage will increase.

Dynamic Voltage and Frequency Scaling (DVFS) is a well-known method to improve the energy efficiency of computing systems. The voltage and frequency levels are typically adjusted through the use of system-level software known as *frequency governors*. The mechanism used by these governors to control the frequency knobs tend not to be application aware, which leads to energy inefficiencies [6]. In addition, this problem is aggravated by the fact that on modern SOCs such as the NVIDIA Jetson TX1 (which is used for this work), the number of possible frequency combinations can be very large, rendering a brute-force search for the energy-optimal point infeasible. In order to evaluate the energy efficiency of the Jetson TX1's default DVFS system, we study a diverse set of collaborative applications, and collect performance and power data on the TX1. To understand the potential benefits of finding the best voltage-frequency settings for the CPU,

978-3-9819263-0-9/DATE18/©2018 EDAA



(a) CPU compute and Memory bound application: GA.

(b) GPU compute and Memory bound application: SC.

Fig. 2: Performance scaling for the GA (left) and SC (right) applications.

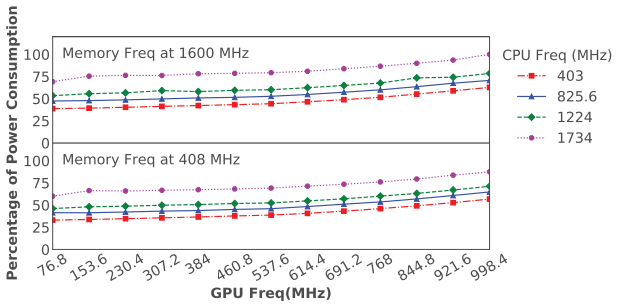


Fig. 3: Normalized power consumption in percentage of GA across different frequency settings.

GPU and memory on the TX1, we ran experiments across all possible frequency combinations. We perform this study, while running collaborative applications taken from the Chai [2] and Hetero-Mark [1] benchmark suites. What is challenging is that the TX1 has three different voltage-frequency domains, with the memory domain being shared between the CPU and GPU. The detailed experimental platform and benchmark setup are described later in Section III. In Fig. 1 we plot and compare the Energy Delay Product (EDP) of the default power manager versus an oracle manager, evaluating several applications from the Chai and Hetero-Mark benchmark suites. The plot clearly identifies the inefficiencies of the default power manager in terms of energy efficiency, and motivates us to develop a new scheme to address this gap.

To approach this multi-dimensional optimization problem of managing energy efficiency on APU-class devices, we utilize application characteristics, coupled with runtime feedback, to tune the frequency across all three domains (CPU,GPU and Memory). We present Airavat, a power management framework that uses application characteristics to guide the system to find a more energy efficient operating point, while ensuring there is negligible performance loss. Airavat employs two layers of frequency tuning, that includes: i) a Coarse-grained Frequency Approximation Layer and ii) a Collaborative Fine-grained Frequency Tuning layer. The first layer uses predictive modeling to approximate the best frequency combination (CPU, GPU, and memory) for a given workload, whereas the second layer uses runtime feedback to further tune the frequency to find a better energy-efficient operating point. We implement our power management framework on the NVIDIA Jetson TX1 and observe that it achieves an improvement in terms of average Energy Delay Product (EDP) by 24% over the baseline power manager, with a maximum EDP improvement of 70%. The contributions of our work include: 1) we perform a power-performance characterization of emerging CPU-GPU collaborative applications, 2) we propose Airavat, a two-tiered power management framework which is geared towards improving energy efficiency of emerging heterogeneous applications, and 3) we evaluate the energy savings and performance benefits of Airavat and find that, on an average, we observe a 24% EDP improvement over the baseline power manager.

II. Background and Motivation

Before introducing our power management framework, we characterize how the power consumption and performance changes with CPU, GPU, and memory frequencies. If we can gain a better understanding how applications scale with the frequency of different subsystems (CPU/GPU/memory), we can arrive at an optimized power/performance operating point.

We begin by reporting performance and the power consumption of two applications: the *Gene Alignment (GA)* and the *Stream Compaction (SC)* from the Hetero-Mark and Chai benchmark suite, respectively. We vary the CPU, GPU and memory frequencies, and compare performance and power consumption at different voltage-frequency pairs. We select these two applications given their very different characteristics. The GA benchmark runs unique tasks on the CPU and GPU collaboratively, whereas the SC benchmark divides the data between the CPU and GPU and runs the same task on both devices.

Fig. 2a and Fig. 2b shows the normalized performance at different combinations of GPU, CPU and memory frequencies for the GA and SC applications. Fig. 3 shows the variation in the System-on-Chip (SOC) power as the frequency scales. Since the power consumption trends in each of the applications were very similar (i.e., power consumption increases with increase in frequency), we plot the power consumption for only GA. From Fig. 2a, we observe that the performance of GA scales well with CPU and memory frequency. However, the application does not heavily use the GPU, which is evident

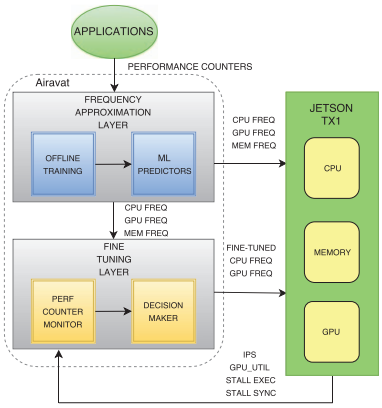


Fig. 4: The Airavat power management framework.

since, as we increase the GPU frequency, performance does not change. This trend is present because the GPU portion of the application does very little computation as compared to the amount of data movement, and is fairly memory bound. As a result, increasing the GPU frequency elevates the power consumption of the chip, as indicated in Fig. 3. A power governor that is unaware of an application's behavior might introduce significant power dissipation without any return in performance. The CPU part of GA is compute bound, but not memory bound. Hence, this highlights the need for a power management framework to take into account the bandwidth requirements of the CPU and the GPU, and then scale the memory frequency in an power efficient manner.

Fig. 2b shows the scaling behavior of the SC application. We observe that this application is primarily GPU bound and memory (GPU-based memory traffic) bound, based on the amount of data movement and the presence of divergent GPU threads. Although the CPU runs the same task, it processes only a fraction of the data, so the application is only CPU compute-bound for a short amount of time. Unlike the GPU, the CPU does not suffer from divergence thanks to branch predictors and out-of-order execution. Increasing the CPU frequency in this case just wastes CPU power. By increasing the CPU frequency, the CPU is just waiting faster until the GPU finishes its work.

The observations in this section suggest that to improve the energy efficiency of emerging collaborative applications, a power management framework should make decisions based on application characteristics rather than just tuning the frequencies according to the load. Most default power governors today utilize load as their only guidance.

III. Methodology

Since Airavat is targeted at power and performance management of collaborative applications running on integrated heterogeneous architectures, we have selected an NVIDIA Jetson TX1 platform to evaluate the potential of our power management framework. The TX1 incorporates an NVIDIA Maxwell architecture GPU with a quad-core ARM A57 CPU on the same die. The CPU and GPU share a common memory.

TABLE I: GPU Performance Counters

GPU Counters	Description
IPC	Instructions executed per cycle
ALUUtil	Utilization level of the arithmetic units
L2Util	Utilization level of the shared L2 cache
DRAMUtil	Utilization level of the main memory
IssueSlotUtil	Utilization level of the issue slots
ExecStalls	% of stalls because input unavailable
MemStalls	% of stalls because memory operation pending
SyncStalls	% of stalls because of thread synchronization
ALUInsts	% of arithmetic instructions
ControlInsts	% of control flow instructions
LDSTInstructions	% of load store instructions
HitRate	Cache hit rate of the unified texture cache
Occupancy	Avg. active warps / max supported warps

The TX1 supports 17 different CPU frequencies, 13 different GPU frequencies and 10 different memory frequencies, resulting in a total of 2210 possible combinations. We measure the power consumption of the SOC using built-in INA power monitors. The INA monitor is a Texas Instruments chip that continuously monitors the power and current consumed by the Jetson TX1 SOC at a granularity of $\approx 1ms$.

Since our focus is on power-performance of collaborative CPU-GPU execution, we use applications from the Chai [2] and Hetero-Mark [1] benchmark suites. We extend the Hetero-Mark benchmark suite to support CUDA and use the collaborative applications from Hetero-Mark. From the Chai benchmark suite, we use all of the benchmarks except Transposition and Padding. This is because Transposition only uses the GPU during execution, whereas Padding suffers from invariably long runtimes, rendering exhaustive experiments impractical.

Although DVFS policies can be designed for different goals in mind we chose to optimize our framework for Energy-Delay Product (EDP).Given the fact that embedded SOCs such as the Tegra are designed to run standalone on batteries, we believe EDP is a good metric, as opposed to the Energy Delay Squared Product (ED^2P), which gives more weight to performance than power.

IV. Airavat System Design

In this section, we describe the design of Airavat, a power management framework aimed at optimizing the energy efficiency of fused CPU-GPU platforms. Airavat consists of two layers, as shown in Fig. 4. The first is the *Frequency Approximation Layer (FAL)*, which predicts near optimal frequencies for a given workload based on offline trained machine learning models. The second layer is the *Collaborative Tuning Layer (CTL)*, which utilizes runtime feedback to jointly tune the CPU and GPU frequencies so as to balance the CPU-GPU performance according to application-specific characteristics and further improve the energy efficiency.

A. Layer 1: Frequency Approximation Layer (FAL)

The FAL of Airavat leverages machine learning models to guide the hardware to a frequency configuration that minimizes the EDP. The goal of FAL is to take offline-profiled CPU and GPU performance counters as input, and then predict the

TABLE II: CPU Performance Counters

CPU Counters	Description
CPI	Cycles per Instruction
L1Miss	Number of L1 data cache misses
L2Miss	Number of L2 data cache misses
L2Access	Number of accesses to the L2 cache
MemAccess	Number of accesses to Main Memory
BranchMispred	Number of branch mispredictions

Input: $(CPU_{fal}, GPU_{fal}, Mem_{fal})$
Output: Tuned CPU and GPU frequency
$StopCond1 = (IPS_{f2} - IPS_{f1}) < 10\%$;
$StopCond2 = (IPS_{f2} == IPS_{fal})$;
/*f1 is the frequency at previous step. f2 is the new frequency*/;
/*Repeat while loops for both CPU and GPU*/;
while $(!StopCond1)$ **do**
  TuneUpDeviceFrequency(Device);
  MeasureIPS();
  if $(DeviceTuned == CPU)$ **then**
    if $(GPUUtil_t < GPUUtil_{t-1})$ **then**
      RevertBack(CPU_{freq});
      break;
    end
  end
end
while $(!StopCond2)$ **do**
  TuneDownDeviceFrequency(Device);
  MeasureIPS();
  if $(DeviceTuned == GPU)$ **then**
    /*Check Execution and Sync Stall counters*/;
    if $(Sexec_t > Sexec_{t-1}) or (Ssynq_t > Ssynq_{t-1})$ **then**
      RevertBack(GPU_{freq});
      break;
    end
  end
end

Algorithm 1: CTL Algorithm

best frequency combination for the CPU, the GPU, and the shared memory. We employ a Random Forest (RF) model for frequency selection. The RF algorithm is an ensemble learning approach that uses decision trees as its essential elements. As the best combination of frequencies are highly dependent on selecting the right performance counters, versus using all of them, a decision tree is a natural fit for selecting the best set of counters and then using them to predict the frequency configuration. Additionally, the ensemble learning approach of RF helps to avoid overfitting, creating better predictions.

We train and validate the RF model with 70 CPU and GPU applications from a number of benchmark suites including Rodinia [7], Parboil [8], Polybench [9] and GraphBig [10]. These benchmark suites consist of applications with diverse power and performance characteristics, providing a rich training corpus. To generate diverse training data sets for multiple predictors, we use both single-threaded and multi-threaded (with OpenMP) implementations of these benchmarks to train the CPU core and CPU memory predictors. Likewise, we use the CUDA version of these benchmark suites for training the GPU core and memory predictors. We use CPU-only and GPU-only benchmarks for training our predictors, since collaborative applications are new, and there are not a large enough set yet to serve as a training corpus.

Performance and power data for the 70 applications are collected at different frequency settings so that we can identify the setting that results in the lowest EDP. In addition, we collect the performance counters listed in Table I and Table II at an intermediate frequency. We use an intermediate frequency value since measuring counters at extreme frequencies results in less representative behavior, since application behavior can change drastically when run at extreme frequencies [11]. We then train our RF models to predict the best EDP frequencies using the measured performance counters.

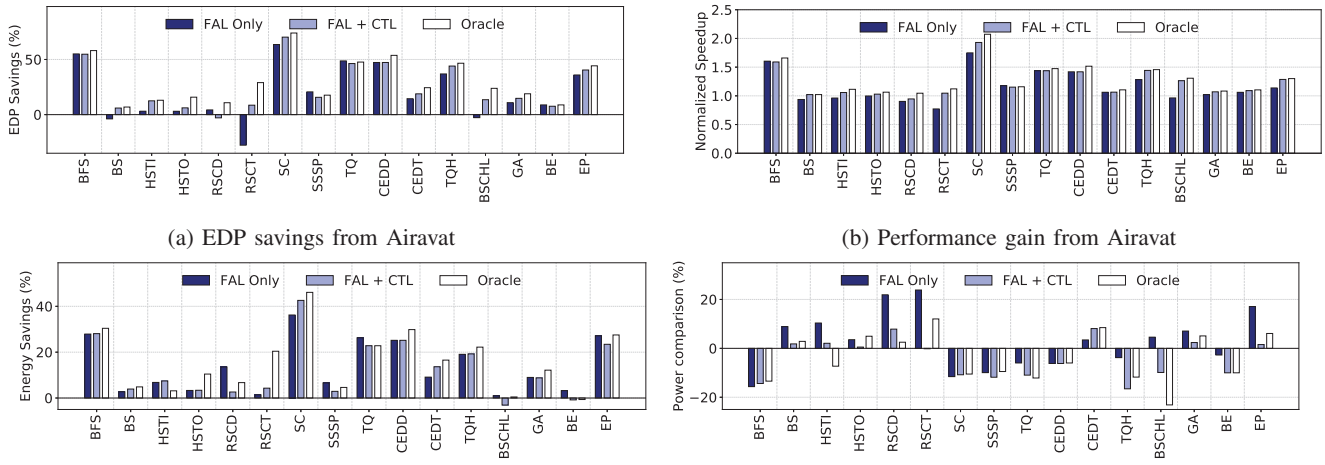
All of the performance counters used for building the prediction models are either taken directly, or derived from, `nvprof` [12] and `perf` [13]. Except for CPI, all the rest of the CPU performance counters from Table II are converted into events per thousands of instructions. We do this because it captures application dynamics more precisely rather than simply using a raw count of the total events. Our performance counter selection is based on our own analysis, as well as strategies presented in previous studies [11], [14]. We build four predictive models to predict the CPU core frequency (F_{CPU}), GPU core frequency (F_{GPU}), CPU memory frequency (F_{C_mem}), and GPU memory frequency (F_{G_mem}). However since memory is shared on our platform, we can select only one out of the two predicted memory frequencies. While selecting the higher memory frequency out of the two predicted values will lead to good performance, it is not necessarily the best solution to achieve improved energy efficiency. To arrive at a better memory frequency value, we propose to use a utilization-based selection policy for the memory frequency. We record the average utilization of the CPU (U_{CPU}) and average utilization of the GPU (U_{GPU}) during the execution of the program, and then apply the following formula to select the new memory frequency (F_{mem}):

$$F_{mem} = \lceil U_{GPU} * F_{G\_mem} + U_{CPU} * F_{C\_mem} \rceil \quad (1)$$

where the $\lceil \rceil$ sign denotes the ceiling operation. As indicated in equation 1, we leverage the utilization of the CPU and the GPU as weights when selecting the proper memory frequency. We then round the value obtained to the closest memory frequency supported by the system. We keep the memory frequency fixed and do not tune it further. Further tuning of the memory frequency requires finer-grained utilization monitoring of the CPU and GPU at a millisecond granularity level. We leave this as future work.

B. Layer 2: Collaborative Tuning Layer (CTL)

Although FAL succeeds in providing a good initial frequency combination, it cannot make use of runtime information. Therefore, FAL alone is insufficient for energy efficient execution of CPU-GPU collaborative applications. This limitation of FAL is addressed by CTL (see Algorithm 1), which uses runtime feedback from performance counters and utilization metrics to fine tune the frequency, with the goal of further improving the energy efficiency. We use the following metrics and performance counters to guide our tuning. 1) CPU Instructions Per Second (IPS) derived from `perf` to tune the CPU frequency, 2) CPU IPS to tune the CPU and GPU collaboratively, 3) Stall Execution Dependency(StallExec) and Stall Synchronization Dependency(StallSynq) of GPU from nvprof as secondary guards 4) GPU utilization as a secondary



(a) EDP savings from Airavat

(b) Performance gain from Airavat

(c) Energy savings from Airavat

(d) Power degradation from Airavat

Fig. 5: Evaluation of Airavat in terms of EDP, performance, energy and power.

guard. IPS has been found to be strongly correlated to CPU performance [6]. We observed that for collaborative applications, the CPU IPS also serves as a strong proxy for GPU performance. For example, given a collaborative workload, if the GPU portion behaves as the limiting factor of the overall workload, the IPS of the CPU will drop because the execution on CPU will be stalled, waiting for the GPU. As a result, the CPU cannot proceed to the next available task or iteration until the GPU finishes its work. In a similar manner, if no IPS gain is observed for an application while increasing the GPU core frequency, it can be implied that the workload cannot benefit from increasing GPU frequency anymore. Thus we use the IPS from CPU to tune the frequency of both CPU and GPU.

A special situation however arises in cases where using IPS as the only metric for collaborative frequency tuning can lead to massive performance degradation. This happens with work stealing applications, where a reduction in the GPU's frequency will increase the IPS for the CPU since the CPU has more work to do. This however does not lead to improved performance, and can, in fact, degrade performance. In a similar manner, an increase in CPU frequency beyond a certain point for work stealing applications will increase the CPU IPS, but lead to less work for the GPU, resulting in lower application throughput (the GPU is more efficient in most cases). To prevent Airavat from aggressively throttling the GPU frequencies in such cases, the GPU utilization, as well as the two GPU performance counters (StallExec and StallSynq) are used as secondary guards. They detect if the CPU is degrading the GPU performance and take corrective actions accordingly.

The algorithm continues until both devices have been tuned and the stopping conditions defined in Algorithm 1 have been met for both the devices, or until the application completes, whichever happens first. Future reexecution of the same application uses the frequency configuration reached at the stopping condition.

V. Results

In this section, we showcase and discuss the energy-delay improvement of Airavat over the default DVFS manager. We also compare our results against an *oracle* predictor which has perfect knowledge of the EDP values at each and every frequency configuration for a given application.

Fig. 5a shows the EDP savings of Airavat over the default DVFS manager. As observed in the figure, we achieve significant EDP savings for a majority of the applications, providing significant improvements in terms of both energy consumption and performance. The average EDP reduction is approximately 24%, and can be as high as 70%. We also see the benefits of having a two-tiered system for tuning EDP. In situations where FAL causes some amount of EDP degradation, CTL is able to take corrective actions to improve the EDP. One such situation where we can see this clearly occurs is the RSCT application. This application is GPU cache sensitive (100% L2 cache utilization) and is heavily memory bound. However, the GPU L2 cache frequency is controlled by the GPU core clock, and since FAL characterized RSCT to be a memory-bound application, it assigns a lower GPU core frequency, leading to EDP degradation. However, CTL recognizes this issue based on feedback, and takes corrective action by increasing the GPU core frequency, improving the EDP. Small degradation in RSCD is caused due to CTL increasing the frequency to more than what is required. We also observe how the presence of a secondary guard with CTL helps to prevent EDP degradation for work-stealing applications, such as BlackScholes (BSCHL). In the absence of secondary guards, CTL will ramp up the frequency of the CPU, or ramp down the frequency of the GPU, as the IPS improves in both cases. But this can severely degrade performance. The secondary guards help to correct this situation. Given the growing popularity of heterogeneous programming frameworks that support work-stealing, it becomes more important for power governors to evolve and integrate a system of checks and balances, so

that energy is not degraded. Furthermore, it can be observed from Fig. 5b that Airavat achieves a sizeable speedup, with geometric mean of 1.12x, when compared against the baseline. As Airavat optimizes for EDP, it takes execution time into account ensuring there is minimal performance degradation.

For most of the collaborative applications, we also observe significant improvements in energy reduction Fig. 5c (average of 17%). This is lower than the EDP savings, since our framework weighs EDP more heavily than energy. We also observe that in some cases, FAL alone provides more energy savings compared to CTL. This is because CTL considers performance improvement as a major criterion and trades performance over energy savings.

Airavat clearly trades off energy improvement over power optimization, as seen from Fig. 5d. The average power loss from Airavat is 4% which is a modest value. It is clearly visible that the power trends of Airavat follow the power trend of the oracle, which is also worse than the default power governor in terms of power consumption. We believe that power managers should be optimized for energy efficiency, rather than power. In Section I we showed how there exists a wide gap between the energy efficiency of the default power manager versus oracle. We believe our power management scheme has succeeded to a large extent in addressing this gap. In terms of the average EDP, Airavat is within 7% of the oracle compared to 25% of the default governor.

VI. Related Work

Predictive DVFS mechanisms for CPU applications have been demonstrated on real hardware before [14]. As GPUs have become more pervasive in computing platforms, developing an effective DVFS algorithm for GPUs has gained a lot of attention. Previous work by Paul et al. [15] proposed a two-tiered power management framework named *Harmonia*. They use a linear regression-based model to compute sensitivities and adjust GPU core and memory frequencies, specifying the number of active GPU cores to arrive at an optimized (ED^2P) for a suite of applications. DVFS for concurrent kernels using neural network-based predictive models on GPUs have been studied by Jiao et al. [16]. Power management strategies on AMD APUs for traditional offload-based GPU applications have been addressed in prior work by Paul et al. [5]. All of these studies focus on applications that either only use the CPU or only the GPU. Our work differs because we focus on improving the energy efficiency of emerging collaborative applications on an integrated heterogeneous platform. To the best of our knowledge, this is the first study that evaluates power management strategies for emerging collaborative heterogeneous workloads and proposes an effective solution that ensures high energy efficiency with negligible performance loss.

VII. Conclusion

This paper presents Airavat, a power management framework that improves the energy efficiency of collaborative applications running on integrated CPU-GPU platforms. Airavat is inexpensive to implement since it leverages existing performance counters available on most commercial integrated CPU-GPU processors. Our framework combines predictive models, along with runtime feedback, to achieve high energy efficiency without sacrificing performance. In the future we plan to augment Airavat with performance and power models to enable predictive DVFS as well as integrate future aware power management schemes like Model Predictive Control.

References

[1] Y. Sun, X. Gong, A. K. Ziabari, L. Yu, X. Li, S. Mukherjee, C. McCardwell, A. Villegas, and D. Kaeli, "Hetero-mark, a Benchmark Suite for CPU-GPU Collaborative Computing," in *Workload Characterization (IISWC), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1–10.

[2] J. Gómez-Luna, I. El Hajjy, L.-W. Changy, V. García-Floreszx, S. G. de Gonzaloy, T. B. Jablin, A. J. Pena, and W.-m. Hwu, "Chai: Collaborative Heterogeneous Applications for Integrated-Architectures," in *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 43–54.

[3] I. Paul, S. Manne, M. Arora, W. L. Bircher, and S. Yalamanchili, "Cooperative Boosting: nEedy versus Greedy Power Management," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 285–296.

[4] J. Lee and H. Kim, "TAP: A TLP-aware Cache Management Policy for a CPU-GPU Heterogeneous Architecture," in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*. IEEE, 2012, pp. 1–12.

[5] I. Paul, V. Ravi, S. Manne, M. Arora, and S. Yalamanchili, "Coordinated energy management in heterogeneous processors," *Scientific Programming*, vol. 22, no. 2, pp. 93–108, 2014.

[6] K. Rao, J. Wang, S. Yalamanchili, Y. Wardi, and Y. Handong, "Application-Specific Performance-Aware Energy Optimization on Android Mobile Devices," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 169–180.

[7] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. Ieee, 2009, pp. 44–54.

[8] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-M. W. Hwu, "Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing," *Center for Reliable and High-Performance Computing*, vol. 127, 2012.

[9] S. Grauer-Gray, L. Xu, R. Searles, S. Ayalasomayajula, and J. Cavazos, "Auto-Tuning a High-level Language Targeted to GPU Codes," in *Innovative Parallel Computing (InPar), 2012*. IEEE, 2012, pp. 1–10.

[10] L. Nai, Y. Xia, I. G. Tanase, H. Kim, and C.-Y. Lin, "GraphBIG: Understanding Graph Computing in the Context of Industrial Solutions," in *High Performance Computing, Networking, Storage and Analysis, 2015 SC-International Conference for*. IEEE, 2015, pp. 1–12.

[11] G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou, "GPGPU Performance and Power Estimation using Machine Learning," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 564–576.

[12] Nvidia, "CUDA Toolkit Documentation," 2016.

[13] V. M. Weaver, "Linux Perf_event Features and Overhead," in *The 2nd International Workshop on Performance Analysis of Workload Optimized Systems, FastPath*, vol. 13, 2013.

[14] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang, "PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*. IEEE, 2014, pp. 445–457.

[15] I. Paul, W. Huang, M. Arora, and S. Yalamanchili, "Harmonia: Balancing Compute and Memory Power in High-Performance GPUS," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3. ACM, 2015, pp. 54–65.

[16] Q. Jiao, M. Lu, H. P. Huynh, and T. Mitra, "Improving GPGPU energy-efficiency through concurrent kernel execution and DVFS," in *Code Generation and Optimization (CGO), 2015 IEEE/ACM International Symposium on*. IEEE, 2015, pp. 1–11.