Value-Based Resource Management at SoC Scale

Serhan Gener, Sahil Hassan, Ali Akoglu
{gener,sahilhassan,akoglu}@arizona.edu
Department of Electrical & Computer Engineering, University of Arizona
Tucson, AZ, USA

ABSTRACT

Value-based resource management heuristics, which are traditionally deployed in heterogeneous HPC systems, maximize system productivity by assigning resources to each job based on its priority and estimated value gain relative to each job's completion time. We investigate the utility of value-based resource management at heterogeneous SoC scale and demonstrate its ability to make effective scheduling decisions for time-constrained jobs in oversubscribed systems where system resources are shared by multiple users and applications arrive dynamically. The proposed value-based resource management approach drops tasks that are estimated to result with lower-value gain dynamically with the aim of completing more number of high-value jobs with a scheduling decision time at 120μs scale. The value-based resource management treats scheduling as a global optimization problem, therefore this study sets a path forward for deploying a unified value-based resource management on a system composed of front-end SoC-based edge devices and a back-end HPC system.

CCS CONCEPTS

• **Computer systems organization** → **Heterogeneous (hybrid) systems**; **System on a chip**; **Real-time systems**.

KEYWORDS

Heterogeneous Computing, Scheduling, HPC, SoC, Value Heuristics

ACM Reference Format:
Serhan Gener, Sahil Hassan, Ali Akoglu. 2023. Value-Based Resource Management at SoC Scale. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3624062.3624243

1 INTRODUCTION

Heterogeneous computing systems, while offering a large potential for performance gains relative to homogeneous counterparts, traditionally pair efficiency gains with reductions in ease of utilizing system resources and programmer productivity. Platforms such as Domain-specific System on Chip (DSSoC) devices have been proposed as a solution for addressing this divergence with the hope that the focus on a smaller domain of applications will enable more productive software and programming abstractions.

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11.
https://doi.org/10.1145/3624062.3624243

The development of heterogeneous SoCs has amplified the demand for new capabilities from resource management algorithms. In a real world deployment scenario, the optimal resource manager should be able to efficiently handle dynamic execution scenarios-which may include an arbitrary number of periodically, and randomly arriving applications, in an overlapping manner. This dynamic workload, paired with a heterogeneous resource pool substantially grows the exploration space for making scheduling decisions, naturally increasing the complexity and latency of the scheduler. Traditional runtime systems undermine the potential of heterogeneous SoCs as the decision-making time can be orders of magnitude larger than the execution time of the task itself [6, 9].

The increase in heterogeneity level in terms of types and number of processing elements when coupled with the demand of multiple dynamically arriving applications require novel approaches to developing runtime resource management and scheduling algorithms that enable effective utilization of heterogeneous SoC platforms and take full advantage of their underlying hardware. Additionally, the heterogeneous compute systems may operate in environments whose characteristics are difficult to fully specify at design time. Furthermore, the dynamically arriving applications may have varying levels of priority or compute requirements. Ability to take application priority into consideration presents an opportunity to relax timing constraints and enable schedulers to dynamically drop low-priority applications. For such dynamically changing execution flows, a resource manager that relies on static cost functions cannot achieve a balance between goals that usually compete with each other (e.g., completion time and energy consumption). The time-constrained applications need to be completed on time. Otherwise, they completely lose their value, and a scheduler needs to take that into account to improve productivity. For this, we propose to investigate the usability of value-based scheduling methods in the heterogeneous SoCs, where we treat scheduling as a global optimization problem rather than an optimization problem to achieve high performing execution for a single application.

In the High Performance Computer (HPC) context, value-based schedulers have been demonstrated as an effective method, while considering deadline and energy efficiency as parameters to optimize for [17–19]. In a job-value function, the value of a time-constrained job represents the worth of completing that job as a function of completion time [19]. In our study, the system-value will be a measure of SoC productivity over a time duration. We calculate the system-value by accumulating the job-value earnings for all the submitted jobs over a duration of time. We aim to maximize the SoC productivity by maximizing the system-value earnings while ensuring the system-wide constraints on the processing element are never exceeded. In an oversubscribed and resource-constrained system, the problem of scheduling jobs to maximize the net system-value is NP-hard [7]. Earlier studies in HPC domain

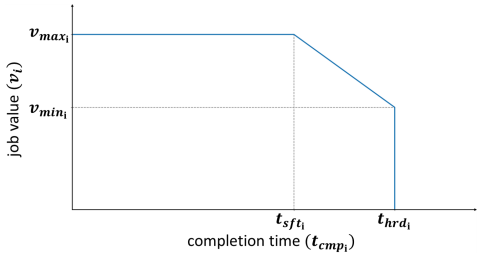

Figure 1: Application gaining maximum value till it's soft threshold (t_{sft_i}) and linearly decaying value as a function of completion time till it's hard threshold (t_{hrd_i}).

have shown that, for an oversubscribed system, value-based heuristics are more productive in scheduling time-constrained jobs as opposed to the traditional backfilling, priority, and deadline-based heuristics [11, 14, 20, 27, 32]. Even though value-based resource management has been applied exclusively in the HPC context for workloads composed of scientific computing applications with execution times in the order of minutes to hours, we believe they are applicable to heterogeneous SoC context as these heuristics execute in polynomial time. Furthermore, value-based optimization allows us to consider factors such as the relative importance of energy efficiency and throughput, the submission time of the task (e.g., during peak load vs. non-peak load period), or the nature of the task. Therefore in this study, for the first time, we investigate the feasibility of value-based resource management for workloads composed of low latency applications from the domain of radar and communications systems in scenarios where multiple applications arrive dynamically. Following are the main contributions of this paper:

- Deployment of value-based scheduler in heterogeneous SoC domain,
- Demonstration of value-based schedulers' ability to prioritize critical applications in dynamic workload scenarios,
- Evaluation of value-based schedulers' resilience in terms of their ability to recover from oversubscribed system state and resume system value gain by meeting the deadlines of applications,
- Perform scheduling overhead analysis and show value-based schedulers suitability for managing low-latency jobs on heterogeneous SoCs.

The rest of the paper is structured as follows: Section 2 introduces the proposed value-based scheduling heuristic. In Section 3, we provide details of the application and simulation environment utilized in our experiments. Subsequently, in Section 4, we analyze the results obtained from diverse experimental workloads. Section 5 offers an in-depth discussion of related works on schedulers for both HPC and embedded domains. Finally, Section 6 presents the concluding remarks and potential avenues for future research.

2 VALUE PER TIME HEURISTIC

In this work, we define application value (v_i) as a monotonically decreasing function that defines the importance of completing the application by quantifying the value earned with respect to the time of completion for that application, as shown in Fig. 1. We show the mathematical representation of the application value function using Eq. 1. The completion time of application i is represented with t_{cmp_i}. An application earns the maximum value (v_{max_i}) when completed before reaching the soft-deadline (t_{sft_i}), and it earns zero value when completed after its hard-deadline (t_{hrd_i}). Any application completed between its soft and hard deadlines earns a value between its maximum and minimum (v_{min_i}) values based on a linearly decreasing function. The slope of this linear function represents the relative importance of the application. We define *system-value* ($SV(t)$) as the sum of application-values over a period of time using Eq. 2. Definitions of parameters are listed in Table 1.

$$v_i = \begin{cases} 0, & t_{cmp_i} > t_{hrd_i} \\ v_{max_i}, & t_{cmp_i} < t_{sft_i} \\ v_{max_i} - \frac{(t_{cmp_i} - t_{sft_i}) \times (v_{max_i} - v_{min_i})}{t_{hrd_i} - t_{sft_i}}, & otherwise \end{cases} \quad (1)$$

$$\text{maximize } SV(t) = \sum_{i=1}^{S(t)} v_i(t_{cmp_i}) \quad (2)$$

We represent each application as a DAG where each node is a task in the application and each edge indicates the data flow. We implement a greedy heuristic as illustrated in Algorithm 1, which searches for the processing element (PE) among a pool of heterogeneous set of PEs that yields the highest value gain for a given task. We consider a runtime system where tasks whose dependencies have been resolved are stored in the ready queue. A mapping event occurs when there is at least one task in the ready queue, which needs to be assigned to a PE for execution. Possible value gain (val_{task_i,PE_j}) for a task is the list of values obtained based on the anticipated execution of that task on any of the PEs in the system using Equation 1, where task completion time $t_{cmp_{task}}$ is used instead of application completion time t_{cmp} to calculate the potential value gain of the task. At each mapping event, we calculate the possible value gain for each task in the ready queue (lines 2-11) and keep track of the maximum value a task can gain ($v_{max_{task_i}}$). We sort the tasks based on the maximum value gain (line 12), and from highest to lowest gain, each task in the ready queue is mapped onto its designated PE that yields maximum possible value gain for the task. For the case where a task may have two or more PEs yielding the same maximum value gain, we choose the PE that offers the earliest completion time ($t_{cmp_{task_i,PE_j}}$) (line 6) that is calculated by accumulating the task execution time on each PE ($t_{exec_{task_i,PE_j}}$) and the time when that PE will become available ($t_{avil_{PE_j}}$). In case of a tie among completion times, a task is assigned to the PE with the lowest number of task assignments in the current mapping event.

While the approach presented in Algorithm 1 (VoS) first evaluates all tasks in the ready queue and assigns the task based on max value gains, a better solution would be assigning tasks one by one and re-calculating possible value gains based on new assignments and PE availability time changes. This new approach no longer needs a sorting step as in line 12 of Algorithm 1. However, as the task assignments are performed one by one and value gains are recalculated after each assignment, the computational complexity of this approach is $O(n^3)$ while it is $O(n^2)$ for the initial approach. We refer to this second approach as VoS3n. Both versions of Algorithm 1 have a feature that drops applications that have failed their hard

Table 1: Definitions for Equation 1 and 2.

parameter	definition
i	application id in the order of arrival
$v_i(t)$	value earned by i^{th} application at time t
t_{hrd_i}	hard deadline for the application i
t_{sft_i}	soft deadline for the application i
t_{cmp_i}	completion time for the application i
v_{max_i}	maximum value application i can obtain
v_{min_i}	minimum value application i can obtain
t	time since the start of an experiment
$S(t)$	number of applications submitted till t
$SV(t)$	system value at time t

Algorithm 1 VPT Based Scheduling

procedure SCHEDULER(*task_ready_queue*, *PE_list*)
  for $task_i$ in *task_ready_queue* **do**
    $v_{max_{task_i}} \leftarrow 0$
    for PE_j in *PE_list* **do**
      $val_{task_i,PE_j} \leftarrow$ POSSIBLEVALUEGAIN($task_i, v_i$)
      $t_{cmp_{task_i,PE_j}} \leftarrow t_{avil_{PE_j}} + t_{exec_{task_i,PE_j}}$
      if $v_{max_{task_i}} < val_{task_i,PE_j}$ **then**
        $v_{max_{task_i}} \leftarrow val_{task_i,PE_j}$
      end if
    end for
  end for
  $sorted\_ready\_queue \leftarrow$ SORT($val_{task_i,PE_j}, v_{max_{task_i}}$) ▷ Sort *val* dictionary based on v_{max}
  $assigned\_tasks_{PE_j} \leftarrow 0$
  for $task_i$ in *sorted_ready_queue* **do**
    if $v_{max_{task_i}}$ not unique in val_{task_i,PE_j} **then**
      if min($t_{cmp_{task_i,PE_j}}$) not unique $\forall j$ **then**
        Assign $task_i$ to PE_j with min($assigned\_tasks_{PE_j}$)
      else
        Assign $task_i$ to PE_j with min($t_{cmp_{task_i,PE_j}}$)
      end if
    else
      Assign $task_i$ to PE_j that gives $v_{max_{task_i}}$ value
    end if
    $assigned\_tasks_{PE_j} \leftarrow assigned\_tasks_{PE_j} + 1$
  end for
end procedure

deadlines. This gives priority to the other applications that still have not failed their t_{hrd}. During a mapping event, while calculating the possible value gain for a task in the ready queue, the value-based scheduler can identify the tasks that would result in a value gain of 0. Once a task has a possible value gain of 0, the scheduler no longer schedules the task to any PE. It also directly removes the task from the ready queue and the application to which it belongs from the system. However, any task already scheduled to a PE for the failed application will still be executed. This provides the scheduler with more opportunities to schedule tasks belonging to applications that are worth completing.

Table 2: Application characteristics used in the experiments

	RC	WiFi-TX	LD
EFT Exec	5,865	61,910	78,522
v_{max}	1	10	100
Maximum Branches	2	5	1
Accelerateable Tasks	3	5	4
Acceleration Gain	10x	10x	15x-8x

3 EXPERIMENTAL SETUP

We utilize the system-level domain-specific SoC simulation [3] (DS3) platform to test our value-based scheduler. DS3 enables us to simulate real-world applications given in DAG form to validate the correctness of our scheduling approach. Moreover, DS3 has several built-in schedulers, such as Earliest Finish Time (EFT), Earliest Task First (ETF), and Minimum Execution Time (MET) for performance comparison. We evaluate the performance of a scheduler in terms of the number of applications that fail their hard deadlines and the total system value gained for the duration of the experiments. System value is using the completion time of tail nodes as t_{cmp_i} in Eq. 2 for all the completed applications at the end of the experiment for each scheduler. As for the failed hard deadlines, the application is considered failed if it cannot be successfully completed before its hard threshold (t_{hard}). With DS3, we can compose any SoC configuration and run experiments as close to real-life execution as possible. We simulate an NVIDIA Jetson Xavier MPSoC [24] (Jetson) development board composed of 8 ARM cores and a Volta GPU operating at 2.3GHz and 1.3GHz, respectively on DS3.

In our experiments, we utilized three applications: radar correlator, WiFi-TX, and lane detection, each serving distinct purposes. The radar correlator employs a maximum of two DAG branches and involves computationally intensive tasks, specifically three FFTs. For the WiFi-TX application, its DAG branches can consist of up to five parallel tasks, with 5 FFTs being the high-load tasks. On the other hand, the lane detection application exclusively follows a sequential task execution pattern, with four convolution-2D tasks acting as the compute-intensive ones. To determine the value of system parameters, several factors are considered and specified in Table 2. The expected execution time of a DAG node (task) represented in DS3 is measured by running the application using a heterogeneous runtime system on the Jetson board. While there are other runtime systems such as StarPU [4] and IRIS [15], in this study we utilize CEDR [21] for profiling purpose, as DS3 has been calibrated with CEDR. We used the average runtime of 500 trials to gather reliable values. The execution time of an application is defined as the execution of the application using EFT based scheduler. We categorize RC as a low latency job and LD as a high latency job with higher criticality than the other two applications. Therefore we set the value gained for completing RC, WiFi-TX, and LD as 1, 10, and 100, respectively. All three applications share a common v_{min} value of 0.5, while t_{sft} and t_{hrd} are scaled with 1.0 and 2.1 based on EFT Exec times shown in Table 2, respectively. Throughout the experiments, the injection period (application arrival interval) is set to 78.5μs. This value was selected experimentally after sweeping injection rates between 10 and 100μs to determine an injection rate that satisfies generating an oversubscribed system yet provides a workload composition where we can evaluate VoS, EFT, ETF, and MET schedulers through trend-based analysis. Additionally,

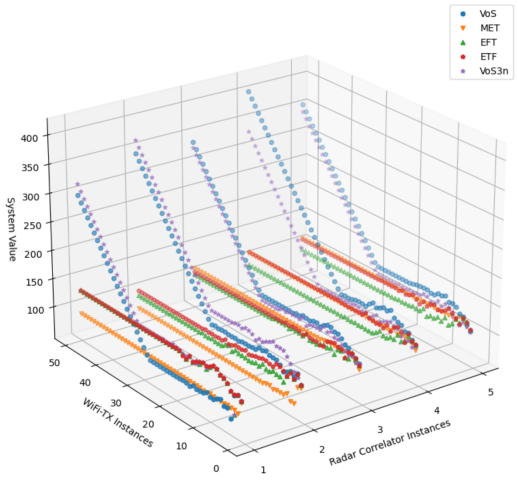

Figure 2: System value gained by each scheduler with single LD instance while WiFi-TX and RC instances vary from 1 to 50 and 1 to 5, respectively.

the scheduling decision allows computationally expensive tasks (FFT and Convolution) to be executed on the accelerator, while the remaining tasks are executed on the CPUs.

4 RESULTS AND ANALYSIS

In this section, we run various simulations, evaluate the performance of value-based schedulers (VoS and VoS3n) with respect to EFT, ETF, and MET schedulers through various workload scenarios, and demonstrate the robustness of value-based schedulers.

4.1 System Value Trends: Undersubscribed to Oversubscribed System

In this first experiment, we set the number of LD instances to 1, and observe the changes in the system value as we sweep the number of RC instances from 1 to 5 and WiFi-TX instances from 1 to 50. This experiment aims to analyze the behavior of schedulers in a scenario where only a few high-priority jobs are available in the system while increasing the number of applications belonging to the mid-value job. Consequently, the system will initially operate in a more relaxed state with low demand and gradually become oversubscribed as the number of application instances increases. We present a 3D plot illustrating the system value gained by each scheduler in this workload composition using Fig. 2. While adding RC applications to the system does not significantly impact the observed saturation trends, it influences overall system behavior. Therefore we focus on the trendline where the number of RC instances is fixed to 3 and show the 2D plot using Fig. 3. There are three regions with distinct trends in this plot.

In the first region, the number of WiFi-TX instances ranges from 1 to 10 and the system gradually becomes over-subscribed. The value-based schedulers show continuously increasing value gains, while other schedulers observe an early saturation at three WiFi-TX instances. This shows that the value-based scheduler is more effective in application prioritization.

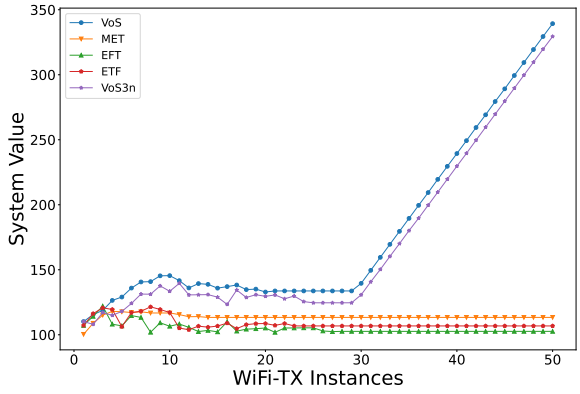

Figure 3: System value gained by each scheduler with single LD instance and three RC instances, where we vary WiFi-TX instacnes from 1 to 50.

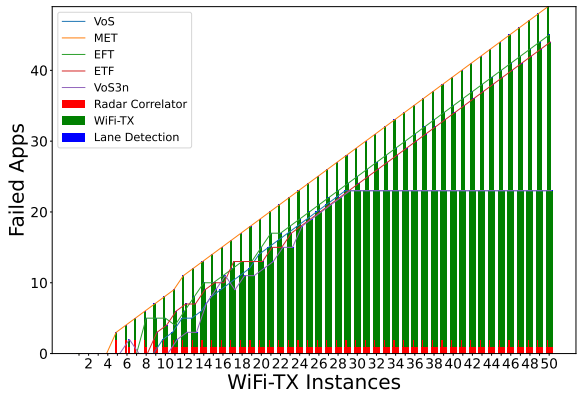

Figure 4: Number of failed applications for RC, WiFi-TX and LD individually along with trend lines for each scheduler showing the total number of failed application instances for the scenario with single LD, three RC instances and WiFi-TX instances range from 1 to 50.

During the second region where WiFi-TX instances vary from 10 to 30, all schedulers show saturation and particularly between 10 and 15 value-based schedulers show a reduction in system value along with ETF and EFT. The inability of these schedulers to handle the high workload of incoming jobs leads to value loss. However, the proposed value-based schedulers consistently achieve a higher system value gain in this region. Interestingly, the greedy MET scheduler is performing better than sophisticated schedulers EFT and ETF as it favors assigning FFT and convolution tasks to accelerators and allows completing high-value applications before their soft-deadlines and in turn gains the maximum value. In return, MET results with having more number of lower-valued applications fail their hard deadlines as shown in Fig.4. This outcome is primarily influenced by the system value calculations, which employ a linearly decreasing value function depicted in Fig. 1. Consequently, completing high-value jobs, such as LD, well in advance of their deadlines results with greater overall value gain.

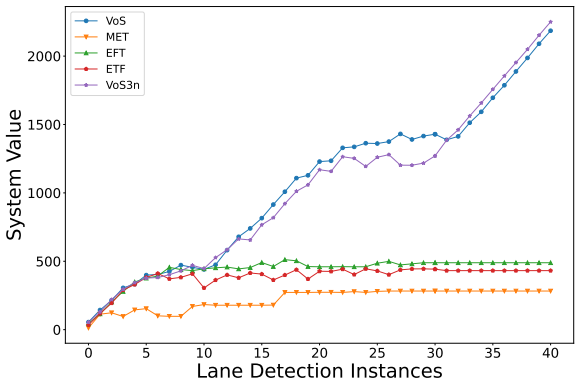

Figure 5: System value gain with respect to number of LD instances where workload composition varies with injection of 10 of RC and WiFi-TX applications.

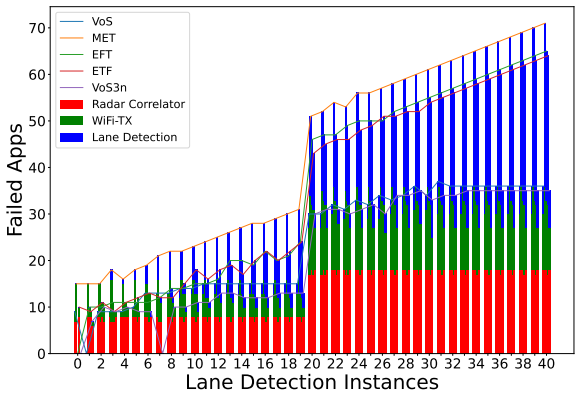

Figure 6: Number of failed applications with respect to number of LD instances where workload composition varies with injection of 10 of RC and WiFi-TX applications.

In the last region, where the number of WiFi-TX applications ranges from 30 to 50, we observe that value-based approaches recover from the saturated state and reach a steady state system value gain trend. The recovery is possible due to the capability of these schedulers to drop failed applications based on their awareness of the applications' hard deadline. In contrast, other schedulers cannot recover from the saturated system value state. We show the number of application instances that fail to meet their hard deadlines in Fig. 4 for the scenario where we fix LD to one and RC to three and vary WiFi-TX instances from 1 to 50. We show the number of failed application instances for RC, WiFi-TX, and LD individually and include a line trend for each scheduler showing the trend in the total number of failed application instances with respect to the number of WiFi-TX instances. Notably, none of the schedulers fail the hard deadline of the single instance of LD application in the system. However, after 10 instances of WiFi-TX, all schedulers except MET fail 2 out of 3 RC instances, while MET fails all 3. As we continuously introduce more instances of WiFi-TX up to the highest tested number of 50 instances, MET, EFT, and ETF schedulers continue to fail more WiFi-TX instances. In contrast, value-based schedulers stop failing WiFi-TX starting with 30 instances, consistent with the observed recovery trend in Fig. 3.

4.2 Reproducing Saturation Trends

As a follow up to the observations from the first experiment, we designed a subsequent experiment to investigate the system's behavior when multiple applications arrive in bursts followed by steady state application arrival followed by another period of applications arriving in bursts. Our aim is to demonstrate the robustness of the value-based scheduler in terms of its ability to recover from saturation as observed in the previous experiment under varying workload scenarios.

We maintain a constant injection rate of 78.5 μs throughout the experiment, starting with a phase of 10 instances of RC and WiFi-TX applications, and a single LD application arrives periodically till we reach a total of 10 LD instances. After this high-workload phase, we only inject LD application periodically till we reach a total of 20 LD instances representing a steady state phase. After reaching the 20 LD instances, we resume a high-workload phase by injecting 20 RC and 20 WiFi-TX instances, while LD is arriving periodically till we reach a total of 40 LD instances. During the high workload phase, when three applications arrive, they are injected simultaneously. We plot the total system value gain for each heuristic with respect to the number of LD instances in Fig.5 and the total number of failed applications in Fig.6.

We observe four distinct trends in Fig. 5 influenced by the nature of the workload generated in this experiment. During the initial high workload arrival phase up to 10 LD instances where WiFi-TX and RC are fixed to a total of 10 instances each, all schedulers except MET result with a similar trend in system value gain saturating around 500 at approximately 7 LD instances. The MET during this high workload phase fails to complete a much less number of applications compared to other heuristics and as shown in Fig.6 and saturates at 2 LD instances. Similarly, after 6 LD instances, ETF starts lagging behind EFT and VoS schedulers. Since we consider LD application as a high-priority job (with the highest max value), examining the failed applications count reveals interesting insights. The MET fails its first LD application when there are 3 LD instances, while ETF and EFT fail to complete the first LD application at LD instance of 7 and 8, respectively. Both versions of value-based schedulers fail their first LD application at 9 LD instances. This demonstrates the ability of value-based schedulers to compensate for high-value jobs and prevent them from missing their hard deadlines more effectively compared to the other schedulers. The MET, EFT, and ETF schedulers remain in the saturated state even during the steady state phase (LD instances 11 to 20) where we only inject LD periodically, while value-based schedulers quickly recover from the saturated state and as early as LD instance of 11, starts gaining value indicating that they can complete the applications before their hard deadline. During this phase, we observe that value-based schedulers stabilize and maintain a similar number of failed applications. On the other hand, the other schedulers continue to experience an increase in the failed number of applications as shown in Fig.6. The trend for VoS schedulers changes back to the saturated state when we disturb the system with a new high workload phase (LD instances 21 to 30) where we introduce 20 instances of WiFi-TX and RC periodically. During this second high-workload phase, the gap between VoS and VoS3n is more visible

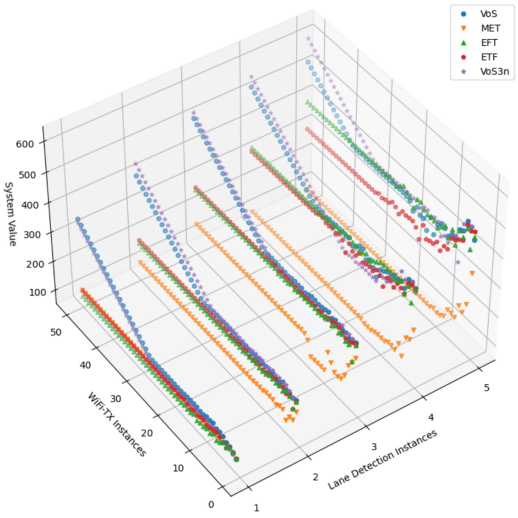

Figure 7: SystemValue3D with 5 RC instances across varying WiFi-TX and LD instances

compared to the first high-workload phase. In general, both the VoS and VoS3n approaches achieve similar system values in this experiment. However, there is a gap between the two approaches in the 13 to 31 LD instance section. While the VoS3n approach shows a lower number of total failed apps in this region, the system value gained by the VoS-based approach is slightly higher. This difference can be attributed to the VoS3n approach favoring three fewer WiFi-TX instance failures than VoS. But as a result, one more LD instance fails its deadline at 17 instances, which causes VoS3n to have a lesser total system value compared to VoS. Later in the second high-workload phase starting from 27 on until 31 instances of LD, every additional LD instance in the VoS3n approach starts to complete much closer to its t_{sft} with respect to VoS, causing VoS3n to catch up in total system value. Consequently, VoS3n eventually matches the total system value of VoS at 31 instances of LD. Immediately after switching back to the steady-state phase with LD instances of 31 and beyond, both schedulers recover as expected and maintain a steady increase in system value gain, demonstrating their robustness and adaptability. In conclusion, value-based schedulers are resilient in managing failed applications and can effectively prevent high-value jobs from missing deadlines.

4.3 System Value Trends in an Oversubscribed System

In our final experiment, we put higher stress on the schedulers by varying the number of mid- and high-priority jobs running on the system, WiFi-TX and LD, while fixing the number of RC instances to 5. We perform a sweeping experiment as shown in the 3D plot (Fig. 7) illustrating the system value gain trends for 1 to 50 WiFi-TX and 1 to 5 LD instances. The trend lines for 1, 2, and 3 LD instances show that all schedulers except MET exhibit a similar trend observed in the earlier two experiments. All schedulers gain system value as the number of WiFi-TX instances increases and eventually saturate. However, when we consider plots for 4 and 5 LD instances, a different trend emerges. All schedulers experience

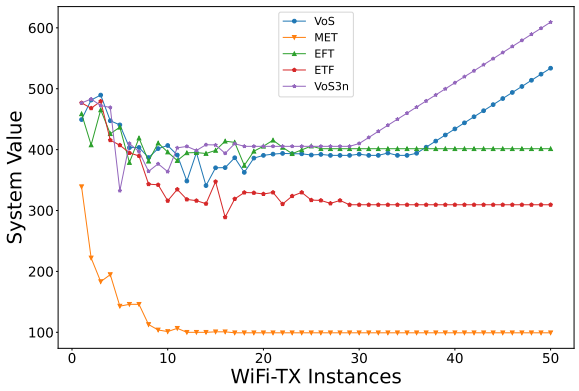

Figure 8: SystemValue with 5 instances of RC and LD across varying WiFi-TX instances

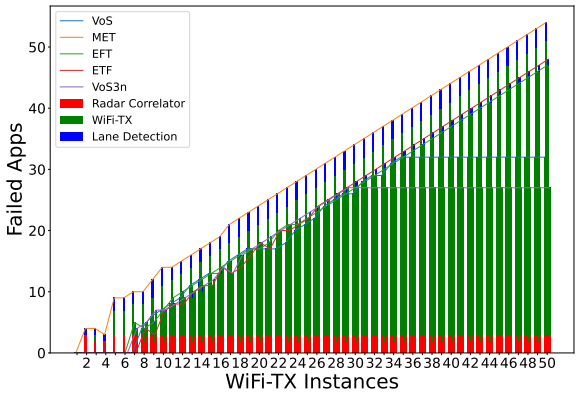

Figure 9: FailedApps with 5 instances of RC and LD across varying WiFi-TX instances

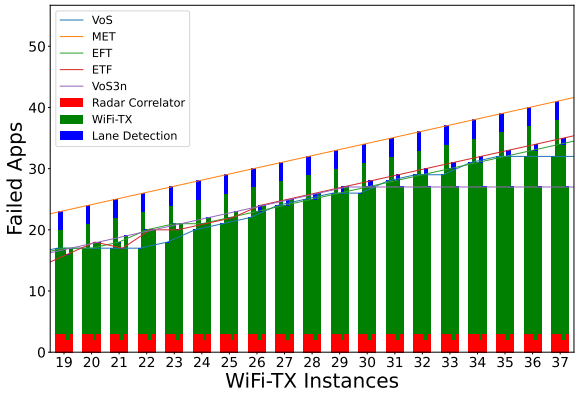

Figure 10: FailedApps with 5 instances of RC and LD across WiFi-TX instances 19-37 region

an initial loss in system value as we introduce WiFi-TX instances and eventually saturate. This behavior can be attributed to starting with a greater number of high-priority jobs. As WiFi-TX instances are submitted, all schedulers initially gain less value, while trying to accommodate the additional WiFi-TX applications. This trend is particularly evident for MET for LD instances of 2 to 5.

To investigate this trend in more detail, we focus on the scenario with 5 LD instances, as illustrated in Fig. 8 and Fig. 9. Analyzing the range of 1 to 10 WiFi-TX instances, we observe a decrease in system value across all schedulers as we increase the number of WiFi-TX instances. This decrease is due to the competition for resources in the system as more applications are running simultaneously, resulting in longer completion times for the same application. As shown with Fig. 9, the MET starts to fail RC and LD applications with only 2 instances of WiFi-TX, while the other schedulers begin failing their first set of RC and WiFi-TX applications around 7 to 8 instances of WiFi-TX.

We observe a stable system value for all schedulers in the range of 10 to 30 WiFi-TX instances. However, after 29 instances of WiFi-TX, VoS3n starts gaining system value, and VoS resumes gaining value after 35 WiFi-TX instances. This is also shown in Fig. 9 where the number of failed applications does not increase anymore when WiFi-TX instances reach 29 and 35 for VoS3n and VoS, respectively. MET consistently fails at least 1 instance after introducing 2 WiFi-TX instances. ETF starts failing at least 1 LD instance after 26 WiFi-TX instances, whereas EFT and both versions of value-based schedulers do not show continuous LD application failures. The number of failed applications for MET, EFT, and ETF exhibit a linear increase starting with 17, 24, and 26 WiFi-TX instances, respectively.

A more detailed analysis in the region of 19 to 37 WiFi-TX instances, shown in Fig. 10, highlights the value-based schedulers' ability to evaluate each application instance based on its deadline and priority. VoS3n stabilizes after 29 WiFi-TX instances, and VoS stabilizes after 35 instances, effectively preventing further failed applications. In the most stable state, MET consistently fails 3 instances of LD, while ETF always fails 1 instance. For RC instances, all schedulers except ETF fail 3 instances, while ETF fails 2 instances. Meanwhile, the number of failed WiFi-TX instances gradually increases for all schedulers until 29 WiFi-TX instances.

These results confirm the resilience of value-based schedulers in handling varying workloads and their ability to prioritize applications based on deadlines, effectively reduce the number of failed applications, and recover from saturation, providing further evidence of the advantages of value-based approaches.

4.4 Scheduling Overhead and Resource Utilization

We analyze the scheduling overhead measured in terms of average time spent per scheduling decision by each heuristic based on a workload scenario composed of 40, 20, and 20 instances of LD, RC, and WiFi-TX, respectively. This workload stresses FFT usage on the accelerator and creates an oversubscribed system as discussed earlier in Section 4.2.

In Table 3 we present the scheduling overhead along with the average resource utilization achieved by each heuristic. ETF has the highest complexity among all the heuristics and as a sophisticated scheduler, it favors utilizing all the compute resources targeting high throughput execution. As a result, ETF achieves the highest resource utilization with the cost of highest scheduling overhead. Both EFT and ETF, while achieving higher resource utilization than the value-based heuristics, they fall short in effectively prioritizing critical jobs, leading to more number of applications miss their hard deadlines. As a result, high resource utilization involves dispatching tasks of those critical tasks that eventually miss their deadlines resulting with wasted cycles as shown earlier in Section 4.2. The greedy MET heuristic has the least scheduling overhead but it fails to utilize the system resources. In environments where task criticality is to be taken as a factor, value-based heuristics make smarter decisions and achieve comparable resource utilization.

5 RELATED WORK

Value-based schedulers have been extensively shown to be beneficial in the HPC domain. In [13, 14], the authors show the use of a value-based scheduler that uses the execution of the applications for value calculation. In [17–19], the authors present the benefits of value-based schedulers with multi-objective time, energy, and dynamic allocation for applications. While these show the benefits of using value-based schedulers, they are designed for the HPC domain with numerous available resources, which is not directly suited for the SoC domain. Additionally, the scheduling decisions are made on the application level rather than finer grain task level scheduling decisions that are more suited for the SoC system.

In the embedded domain [1, 10, 12, 28, 33–36] proposes a task-level scheduling algorithm for static workloads, prioritizing constraints like deadline, energy, or security. In [35], the scheduler computes task-level deadlines based on the HEFT [31] scheduler resulting in the recomputation of HEFT scheduling decisions in each scheduling event, which increases the complexity of the scheduling decisions making it only suitable for static workloads. While in [1], task-level deadlines are set based on the critical path of the task, which causes a loss of information regarding the bigger picture of the application, and in [1], the scheduling decisions are made based on a static workload. Dynamic schedulers [2, 5, 8, 16, 22, 23, 26, 29, 30, 37] have been proposed for energy and security optimizations. In [23], authors consider memory contention's correlation with energy and performance for kernel ordering and placement in integrated shared memory heterogeneous systems. In the case of [22, 29], authors do not consider inter-task dependencies in their decision-making. While in [5, 37] the main focus is on the security of the system.

Deadline-based scheduler for a uniprocessor with periodic tasks using FreeRTOS in embedded systems is proposed in [25]. They propose two approaches based on the earliest-deadline first (EDF) algorithm; static slack EDF (SS-EDF) and dynamic slack EDF (DS-EDF). Dynamic power management based energy saving approach is also adopted in both approaches. Their results show that SS-EDF

Table 3: Average Scheduling Overhead (μs) and Resource Utilization (%) for each Scheduler

Scheduler	Scheduling Overhead (μs)	Resource Utilization (%)
VoS	126.38	84.25
MET	13.74	48.09
EFT	130.20	86.71
ETF	207.59	87.02
VoS3n	129.24	84.10

performs better than DS-EDF when the system is over-subscribed due to large overhead requirements for DS-EDF in dynamic calculations in the runtime; otherwise, DS-EDF is more effective.

6 CONCLUSION AND FUTURE WORK

In this work, we explored the usability of value-based scheduling methods in heterogeneous SoC devices to maximize system productivity. Our results demonstrate that value-based heuristics outperform traditional approaches, particularly for time-constrained jobs in oversubscribed systems. Our study shows that value-based scheduling allows for effective resource management in heterogeneous SoCs, enabling the platforms to adapt dynamically to changing execution flows by incorporating task priority and dropping low-priority tasks in the decision-making process. While doing so still stays in orders of 120μs scheduling overhead, which is comparable to well-known schedulers for embedded systems.

Our analysis opens up several avenues for future research to further advance the capabilities of value-based scheduling in heterogeneous SoCs. We plan to investigate multi-objective heuristics targeting execution time and energy in workload scenarios where application priority itself changes dynamically.

ACKNOWLEDGMENTS

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7860. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defence Advanced Research Projects Agency (DARPA) or the U.S. Government.

REFERENCES

[1] Hamid Arabnejad, Jorge G Barbosa, and Radu Prodan. 2016. Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources. *Future Generation Computer Systems* 55 (2016), 29–40.

[2] Vahid Arabnejad, Kris Bubendorfer, and Bryan Ng. 2019. Dynamic multi-workflow scheduling: A deadline and cost-aware approach for commercial clouds. *Future Generation Computer Systems* 100 (2019), 98–108.

[3] Samet E. Arda, Anish Krishnakumar, A. Alper Goksoy, Nirmal Kumbhare, Joshua Mack, Anderson L. Sartor, Ali Akoglu, Radu Marculescu, and Umit Y. Ogras. 2020. DS3: A System-Level Domain-Specific System-on-Chip Simulation Framework. *IEEE Trans. Comput.* 69, 8 (2020), 1248–1262. https://doi.org/10.1109/TC.2020.2986963

[4] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. 2009. StarPU: A unified platform for task scheduling on heterogeneous multicore architectures. In *Euro-Par 2009 Parallel Processing: 15th International Euro-Par Conference, Delft, The Netherlands, August 25-28, 2009. Proceedings 15.* Springer, 863–874.

[5] Xing Chen, Umit Ogras, and Chaitali Chakrabarti. 2022. Probabilistic Risk-Aware Scheduling with Deadline Constraint for Heterogeneous SoCs. *ACM Transactions on Embedded Computing Systems (TECS)* 21, 2 (2022), 1–27.

[6] Alexander Fusco, Sahil Hassan, Joshua Mack, and Ali Akoglu. 2022. A Hardware-based HEFT Scheduler Implementation for Dynamic Workloads on Heterogeneous SoCs. In *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC).* https://doi.org/abs/2207.11360

[7] Michael R Gary and David S Johnson. 1979. *Computers and intractability: A guide to the theory of NP-completeness.* WH Freeman and Company, New York.

[8] Zhishan Guo, Ashikahmed Bhuiyan, Di Liu, Aamir Khan, Abusayeed Saifullah, and Nan Guan. 2019. Energy-efficient real-time scheduling of DAGs on clustered multi-core platforms. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS).* IEEE, 156–168.

[9] John L Hennessy et al. 2019. A New Golden Age for Computer Architecture. *Commun. of the ACM* 62, 2 (2019), 48–60.

[10] Jing Huang, Renfa Li, Jiyao An, Haibo Zeng, and Wanli Chang. 2021. A DVFS-Weakly Dependent Energy-Efficient Scheduling Approach for Deadline-Constrained Parallel Applications on Heterogeneous Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 12 (2021), 2481–2494.

[11] E Douglas Jensen, C Douglas Locke, and Hideyuki Tokuda. 1985. A time-driven scheduling model for real-time operating systems. In *Real-Time Systems Symposium (RTSS).* 112–122.

[12] Wei Jiang, Xia Zhang, Jinyu Zhan, Yue Ma, and Ke Jiang. 2017. Design optimization of secure message communication for energy-constrained distributed real-time systems. *J. Parallel and Distrib. Comput.* 100 (2017), 1–15.

[13] Bhavesh Khemka, Ryan Friese, Luis D Briceno, Howard Jay Siegel, Anthony A Maciejewski, Gregory A Koenig, Chris Groer, Gene Okonski, Marcia M Hilton, Rajendra Rambharos, et al. 2014. Utility functions and resource management in an oversubscribed heterogeneous computing environment. *IEEE Trans. Comput.* 64, 8 (2014), 2394–2407.

[14] Bhavesh Khemka, Dylan Machovec, Christopher Blandin, Howard Jay Siegel, Salim Hariri, Ahmed Louri, Cihan Tunc, Farah Fargo, and Anthony A Maciejewski. 2015, 10 pp. Resource management in heterogeneous parallel computing environments with soft and hard deadlines. In *11th Metaheuristics International Conference (MIC).*

[15] Jungwon Kim, Seyong Lee, Beau Johnston, and Jeffrey S. Vetter. 2021. IRIS: A Portable Runtime System Exploiting Multiple Heterogeneous Programming Systems. In *2021 IEEE High Performance Extreme Computing Conference (HPEC).* 1–8. https://doi.org/10.1109/HPEC49654.2021.9622873

[16] Jong-Kook Kim, Sameer Shivle, Howard Jay Siegel, Anthony A Maciejewski, Tracy D Braun, Myron Schneider, Sonja Tideman, Ramakrishna Chitta, Raheleh B Dilmaghani, Rohit Joshi, et al. 2007. Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment. *Journal of parallel and distributed computing* 67, 2 (2007), 154–169.

[17] Nirmal Kumbhare, Aniruddha Marathe, Ali Akoglu, Salim Hariri, and Ghaleb Abdulla. 2019. Adaptive power reallocation for value-oriented schedulers in power-constrained HPC. In *2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT).* IEEE, 133–139.

[18] Nirmal Kumbhare, Aniruddha Marathe, Ali Akoglu, Howard Jay Siegel, Ghaleb Abdulla, and Salim Hariri. 2020. A value-oriented job scheduling approach for power-constrained and oversubscribed HPC systems. *IEEE Transactions on Parallel and Distributed Systems* 31, 6 (2020), 1419–1433.

[19] Nirmal Kumbhare, Cihan Tunc, Dylan Machovec, Ali Akoglu, Salim Hariri, and Howard Jay Siegel. 2017. Value based scheduling for oversubscribed power-constrained homogeneous HPC systems. In *2017 International Conference on Cloud and Autonomic Computing (ICCAC).* IEEE, 120–130.

[20] Dylan Machovec, Cihan Tunc, Nirmal Kumbhare, Bhavesh Khemka, Ali Akoglu, Salim Hariri, and Howard Jay Siegel. 2016. Value-based resource management in high-performance computing systems. In *Workshop on Scientific Cloud Computing.* 19–26.

[21] Joshua Mack, Sahil Hassan, Nirmal Kumbhare, Miguel Castro Gonzalez, and Ali Akoglu. 2023. CEDR: A Compiler-integrated, Extensible DSSoC Runtime. *ACM Transactions on Embedded Computing Systems* 22, 2 (2023), 1–34.

[22] Sambit Kumar Mishra, Deepak Puthal, Joel JPC Rodrigues, Bibhudatta Sahoo, and Eryk Dutkiewicz. 2018. Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. *IEEE Transactions on Industrial Informatics* 14, 10 (2018), 4497–4506.

[23] Mohammad Alaul Haque Monil, Mehmet E Belviranli, Seyong Lee, Jeffrey S Vetter, and Allen D Malony. 2020. Mephesto: Modeling energy-performance in heterogeneous socs and their trade-offs. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques.* 413–425.

[24] Nvidia AGX [n. d.]. *Jetson AGX Xavier Evaluation Board.* Retrieved Dec 22, 2022 from https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/

[25] Gesse Oliveira and George Lima. 2023. Scheduling and energy savings for small scale embedded FreeRTOS-based real-time systems. *Design Automation for Embedded Systems* (2023), 1–27.

[26] Santiago Pagani, Anuj Pathania, Muhammad Shafique, Jian-Jia Chen, and Jörg Henkel. 2016. Energy efficiency for clustered heterogeneous multicores. *IEEE Transactions on Parallel and Distributed Systems* 28, 5 (2016), 1315–1330.

[27] Binoy Ravindran, E Douglas Jensen, and Peng Li. 2005. On recent advances in time/utility function real-time scheduling and resource management. In *International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC).* 55–60.

[28] Monireh Safari and Reihaneh Khorsand. 2018. PL-DVFS: combining Power-aware List-based scheduling algorithm with DVFS technique for real-time tasks in Cloud Computing. *The Journal of Supercomputing* 74, 10 (2018), 5578–5600.

[29] Sampa Sahoo, Bibhudatta Sahoo, and Ashok Kumar Turuk. 2019. A learning automata-based scheduling for deadline sensitive task in the cloud. *IEEE Transactions on Services Computing* (2019).

[30] Georgios L Stavrinides and Helen D Karatza. 2015. A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds. In *2015 3rd International Conference on Future Internet of Things and Cloud*. IEEE, 231–239.

[31] Haluk Topcuoglu, Salim Hariri, and Min-You Wu. 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems* 13, 3 (2002), 260–274.

[32] Cihan Tunc, Nirmal Kumbhare, Ali Akoglu, Salim Hariri, Dylan Machovec, and Howard Jay Siegel. 2016. Value of service based task scheduling for cloud computing systems. In *International Conference on Cloud and Autonomic Computing (ICCAC)*. 11.

[33] Quanwang Wu, Fuyuki Ishikawa, Qingsheng Zhu, Yunni Xia, and Junhao Wen. 2017. Deadline-constrained cost optimization approaches for workflow scheduling in clouds. *IEEE Transactions on Parallel and Distributed Systems* 28, 12 (2017), 3401–3412.

[34] Guoqi Xie, Yuekun Chen, Xiongren Xiao, Cheng Xu, Renfa Li, and Keqin Li. 2017. Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems. *IEEE Transactions on Sustainable Computing* 3, 3 (2017), 167–181.

[35] Guoqi Xie, Gang Zeng, Xiongren Xiao, Renfa Li, and Keqin Li. 2017. Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems. *IEEE Transactions on Parallel and Distributed Systems* 28, 12 (2017), 3426–3442.

[36] Sonia Yassa, Rachid Chelouah, Hubert Kadima, and Bertrand Granado. 2013. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *The Scientific World Journal* 2013 (2013).

[37] Junlong Zhou, Jin Sun, Peijin Cong, Zhe Liu, Xiumin Zhou, Tongquan Wei, and Shiyan Hu. 2019. Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT. *IEEE Transactions on Services Computing* 13, 4 (2019), 745–758.