Autonomic Power Management in Speculative Simulation Runtime Environments

Stefano Conoci
conoci@diag.uniroma1.it
Sapienza, University of Rome
Rome, Italy

Mauro Ianni
ianni@diag.uniroma1.it
Sapienza, University of Rome
Rome, Italy

Romolo Marotta
marotta@diag.uniroma1.it
Sapienza, University of Rome
Rome, Italy

Alessandro Pellegrini
pellegrini@diag.uniroma1.it
Sapienza, University of Rome
Rome, Italy

ABSTRACT

While transitioning to exascale systems, it has become clear that power management plays a fundamental role to support a viable utilization of the underlying hardware, also performance-wise. To meet power restrictions imposed by future exascale supercomputers, runtime environments will be required to enforce self-tuning schemes to run dynamic workloads under an imposed power cap. Literature results show that, for a wide class of multi-threaded applications, tuning both the degree of parallelism and frequency/voltage of cores allows a more effective use of the budget, compared to techniques that use only one of these mechanisms in isolation. In this paper, we explore the issues associated with applying these techniques on speculative Time-Warp based simulation runtime environments. We discuss how the differences in two antithetical Time Warp-based simulation environments impact the obtained results. Our assessment confirms that the performance gains achieved through a proper allocation of the power budget can be significant. We also identify the research challenges that would make these form of self-tuning more broadly applicable.

CCS CONCEPTS

• **Computing methodologies** → **Discrete-event simulation**; • **Hardware** → **Chip-level power issues**; • **Software and its engineering** → **Software performance**.

KEYWORDS

Power capping; share everything; parallel discrete event simulation

ACM Reference Format:
Stefano Conoci, Mauro Ianni, Romolo Marotta, and Alessandro Pellegrini. 2020. Autonomic Power Management in Speculative Simulation Runtime Environments. In *Proceedings of the SIGSIM Principles of Advanced Discrete Simulation (SIGSIM-PADS '20), June 15–17, 2020, Miami, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3384441.3395980

1 INTRODUCTION

The power consumption of computing systems has emerged as the key limiting factor in the up-hill battle of achieving application speed-up in the post Dennard scaling era. The portion of cores' circuitry that must be turned off at any given time due to thermal constraints for the phenomenon known as Dark Silicon [12] is expected to increase significantly in the following years. In this context, the allocation of the power budget of computing units should be considered as a first-class citizen when targeting both energy efficiency and application performance.

Hardware manufacturers introduced different mechanisms over the years that provide a reduction in the system power consumption in operational contexts, often at the expense of performance. The most significant are Dynamic Voltage and Frequency Scaling (DVFS), which allows lowering the voltage and the frequency (hence the power consumption) of a processor/core in a controlled manner, and Clock Gating, which disables some processor/core circuitry during idle periods. DVFS can be controlled directly from software, while Clock Gating can be exploited indirectly by limiting the number of threads running in parallel. Literature results [9] show that a proper use of these mechanisms at runtime can provide a significant increase in the application performance achievable within a fixed power budget, also known as a *power cap*.

Most literature approaches rely on either exploration-based solutions or model-based solutions, both of which exploit the runtime sampling of the application performance and power consumption for different configurations to predict the most favorable setting. This sampling is generally not trivial for speculative simulations, based on the Time-Warp paradigm [19], where the rate of application progress can only be measured accurately at specific points in time—actual committed events are typically identified when a new Global Virtual Time instant is cooperatively agreed upon by the concurrent processing elements of the simulation. At the same time, most Time Warp runtime environments [5, 23, 26] rely on a (more or less static) binding between Logical Processes (LPs) and Processing Elements. This approach—which usually boils down to each thread managing a subset of all LPs for an execution phase or even for the lifetime of the simulation—is an easy technical solution to guarantee data separation and reduce the number of synchronization points in the runtime environments. Changing the number of threads involved in the simulation at runtime requires a rebinding of LPs to Processing Elements, which might introduce a significant degradation in performance. Similarly, simulation engines which do not require explicit rebinding (see, e.g., [18, 22]) could suffer from memory asymmetry or locality drawbacks.

In this paper, we study the problem of allocating a fixed power budget of a system running speculative simulations at runtime with the goal of maximizing application performance. We apply a state-of-the-art power cap optimization technique to two antithetical Time-Warp based runtime environments with the goal of

SIGSIM-PADS '20, June 15–17, 2020, Miami, FL, USA

ACM ISBN 978-1-4503-7592-4/20/06…$15.00
https://doi.org/10.1145/3384441.3395980

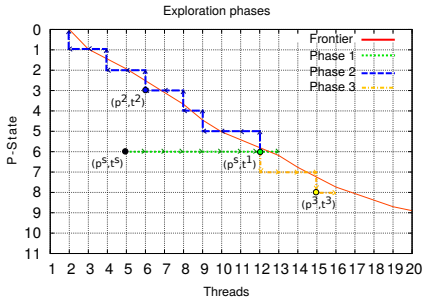


Figure 1: Example of the exploration phases of the power capping technique presented in [9].

showing how the diversities in the underlying Time Warp implementation mechanics can result in significantly different outcomes. We experimentally show how dynamically changing the degree of parallelism in combination with DVFS enables higher performance within the power cap compared to techniques that rely solely on tuning the frequency and voltage of cores. Finally, we outline the characteristics that a Time-Warp based simulation runtime should bear to empower an effective use of the power budget.

2 TARGET MODEL AND MOTIVATIONS

We investigate the challenges and opportunities of optimizing the performance under a power cap of speculative simulations by applying a state-of-the-art power capping technique that relies on the exploration of a portion of the space of configurations, intended as the CPU P-states (cores frequency and voltage) and the number of threads running in parallel, that provides the highest performance within the power constraint [9]. This technique has been proved to find the optimal configurations under a set of assumptions—which we experimentally prove in this paper to also apply for speculative simulations—as long as the returned values of performance and power consumption of the sampled configurations are accurate with respect to their real values.

The selected technique has the distinctive feature of considering application scalability as a decisive element in the power budget allocation, and has been proved to be effective for a wide range of multi-threaded application with very different scalability profiles. This is particularly relevant for speculative simulations—and in general for applications that rely on optimistic synchronization schemes—where an increase in the degree of parallelism can lead to a higher rollback probability, and consequently to an increase in the fraction of power spent on operations that do no contribute to application progress. In this context, considering a fixed power budget, the selection of a configuration with a relatively low number of cores at a relative high core frequency over configurations with a higher number of cores set at a lower operating frequency depends on the specific characteristics of the model, the runtime dynamics of the simulation engine, and the hardware running the simulation.

The exploration procedure relies on three phases, where each phase selects the optimum of a different subset of configurations. Phase 1 explores the space of configurations for a fixed P-state P^1. We note that the selection of P^1 is irrelevant for the correctness of the procedure. At the end of phase 1, the optimum configuration within the sub-space is selected, denoted as (P^1, T^1). Subsequently, Phase 2 explores the space of configurations with P-state lower than P^1 and with the number of active threads lower than T^1. On the opposite, Phase 3 explores the space of configurations with P-state higher than P^1 and with the number of active threads higher than T^1. An example of the exploration steps performed by the technique (with phases highlighted) is provided in Figure 1. By comparing the configurations returned by the three phases, the exploration procedure can select the optimum of the whole bi-dimensional space of configurations in linear time.

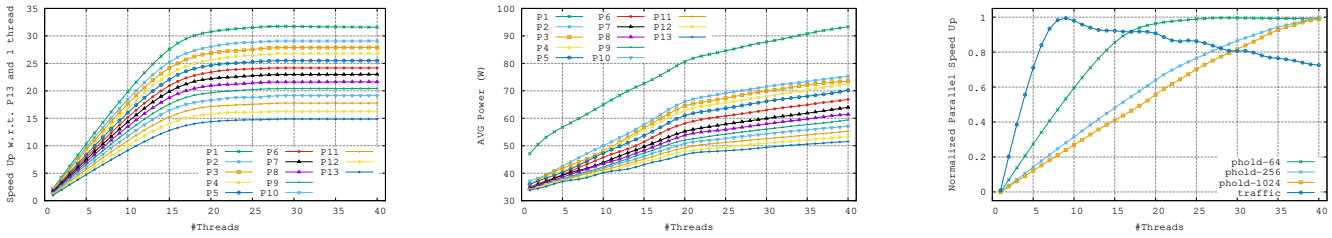
To achieve this result, the technique relies on a set of experimental evidences that allow the exclusion of portions of the space of configurations which surely cannot contain the optimum. These experimental results are obtained by studying the effects on performance and power consumption of changing in combination both the CPU P-state and the degree of parallelism. The authors proved the optimality of the proposed technique under the assumption of these experimental results, which were found to be true for the vast majority of the considered multi-threaded workloads. These assumptions are:

(1) by fixing the CPU P-state and increasing the number of parallel threads, the throughput either increases monotonically and then decreases, always decreases monotonically, or always increases monotonically;
(2) the throughput curves preserve their shape when changing P-state, i.e. the ordering relations on the throughput values when changing the number of threads are not affected by the P-state;
(3) by decreasing P-state, which results in an increase in performance and voltage, the throughput always increases for any fixed number of parallel threads;
(4) either decreasing P-state (increasing frequency and voltage) or increasing the number of parallel threads increases the system power consumption.

In this work, we consider the throughput as a generic metric that measures the rate of application progress. In the context of speculative simulation, we consider throughput as the rate of committed events per second. Different general metrics could also be used, such as the response time, as long as they reflect the different rate of application progress when comparing different configurations. We investigated these assumptions in the context of speculative simulations based on the Time Warp paradigm by studying the results of performance and power consumption of different models run on top of two distinct simulation runtime environments, namely the ROme Optimistic Simulator (ROOT-Sim) [26] and the *Ultimate Share-Everything Simulator* (USE) [18].

ROOT-Sim is based on non-blocking coordination algorithms both on the single node, and in the distributed deployment, thus enabling for enhanced scalability. For the purpose of this work, it is interesting to note that ROOT-Sim enforces a loose binding between LPs and worker threads, supporting a periodic rebalancing operation [33], whose principal goal is to even out the workload assigned to every worker thread, so as to reduce clock skew probability.

USE is a highly-optimized PDES engine for shared-memory multi-core machines, which provides non-blocking progress in both virtual and wall-clock time. The key characteristic of USE is the presence of a unique pool of events to be processed, which is shared among all the compute units (worker threads) of the runtime



(a) Speedup **(b) Power** **(c) Scalability**

Figure 2: Preliminary results which show that the assumptions used by the power capping technique in [9] apply also for speculative simulation.

environment. This guarantees that the computing power is always assigned to the processing of highest-priority events, thus reducing the occurrence of causality violations, and consequently improving the overall efficiency.

The four assumptions on which the power capping technique relies on were found to hold experimentally for different models on both runtime environments. As an example, Figures 2a–2c show the speed-up and power consumption obtained for the whole space of configurations considering the PHOLD simulation model [14] with 64 LPs run on top of USE—further details on the considered models, and the characteristics of the hardware will be discussed in Section 4. The results shown in Figure 2a offer an example to assess the validity of Assumption 2 and Assumption 3. Similarly, Figure 2b affirms the soundness of Assumption 4. Finally, Figure 2c shows the scalability curves of different models run on USE. The curves adhere to the trends identified in Assumption 4. Similar results have been observed also on ROOT-Sim with the same models.

3 RELATED WORK

Runtime tuning and self optimization is a technique which has been thoroughly studied in the context of speculative simulation, focusing on diverse subsystems of traditional Time Warp runtime environments [17]. For example, it has been used to fine tune the checkpoint interval [1, 13, 30], to select the best checkpointing strategy [27], for event scheduling [4, 24, 29, 31], to decide upon the technique to implement a rollback (e.g. state saving vs. reverse computation [7]), to find a proper binding between LPs and processing elements performance-wise [23, 33], or to control overall memory consumption [10].

Most of these optimizations have been tailored for a reduction of the rollback probability [21], for a containment of the effects of rollbacks on performance [11], for accelerating the critical path [32] by dynamically adjusting the number of events executed in each LP cycle, or to jointly reach many of these goals by limiting optimism [20], also in the face of irregular workloads [6]. Previous research [28] has investigated the possibility to rely on DVFS to improve the performance of a Time Warp simulation. Nevertheless, the approach proposed in [28] aimed at controlling the effects of rollbacks by means of core's clock frequency adjustment.

Although recently witnessed as a fundamental aspect [15, 16], energy efficiency in Time Warp simulation has only seen limited attention, having been incidentally deal with either in indirect ways [25], or with a focus on synchronization in distributed deploys [2]. A recent work [8] has proposed a new organization of Time-Warp runtime environments, to tackle performance under a power cap, with a future eye on heterogeneous architectures. This latter architecture could easily benefit from the results in this paper.

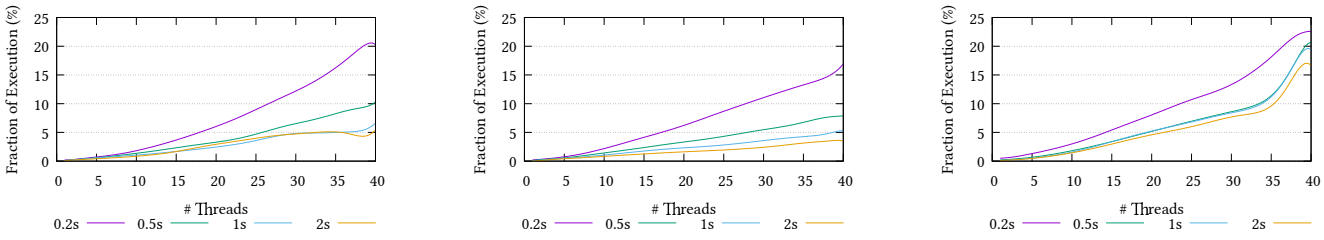
4 EXPERIMENTAL EVIDENCES

We have relied on one synthetic benchmark and one real-world application. The synthetic benchmark is the classical PHOLD [14]. In this benchmark, every LP schedules events for any other LP in the system, with an exponential timestamp increment. We have set the loop duration to provide an event granularity ranging from 70μs to 150μs, depending on the selected P-state. We have run different configurations, using from 60 to 1024 LPs.

We have also modified the standard PHOLD configuration to have some LPs work as *hot spots*. In this varied configuration, all LPs send 50% of their events to one of the hot-spot LPs. We configured this PHOLD variant to use 1024 LPs, and only two hot spots. The hot-spot dynamic is activated periodically, resulting in the model alternatively working with and without hot spots. This kind of workload is traditionally known to be difficult to be managed by PDES runtime environments, since they increase both the (cascading) rollback probability, and the rollback length. It is therefore a good worst-case candidate to stress test the performance-maximization exploration strategy under a power cap. Moreover, its dynamism makes it an ideal test case to assess the benefits of real-time tuning.

The second benchmark is *traffic*, a vehicular model initially presented in [34]. This model offers micro-simulation capabilities of traffic jams at the details of a single car in a street. We have configured the traffic model to simulate the entire Italian highway system (islands excluded, to have a fully-connected graph), using 137 LPs. Every node is described in terms of car inter-arrival time and car leaving probability (therefore describing its load), while edges are described in terms of their length. Given the high amount of data to be processed, the duration of events in this case is much larger, on the order of milliseconds, and varies significantly during the simulation run. Moreover, the small number of LPs involved in the simulation is an actual worst-case scenario, because it shows a very high degree of speculation.

All experiments have been run on a dedicated bare-metal machine, equipped with dual Intel Xeon Silver 4210 CPUs, each with 10 physical cores and 20 threads (for a total of 40 hyper-threads),



(a) 60 LPs (b) 256 LPs (c) 1024 LPs

Figure 3: Fraction of time spent in LP rebinding with varying time interval

Table 1: Normalized standard deviation of throughput samples for PHOLD.

	ROOT-SIM			USE		
SAMPLING PERIOD(S)	64 LP	256 LP	1024 LP	64 LP	256 LP	1024 LP
0.05	6.57	25.28	139.4	4.55	2.20	4.27
0.1	4,04	14.40	76.23	3.25	1.56	2.85
0.2	3.27	9.22	37.39	3.01	1.08	1.30
0.5	2.35	4.64	21.29	1.90	0.65	0.74
1.0	2.30	3.82	15.63	1.85	1.15	1.19
2.0	2.42	3.55	8.62	1.33	1.00	1.23

and 64 GB of main memory. As for environmental software, we have used Debian 10.2, running Linux 4.19.0, and gcc 8.3.0.

Table 1 shows the normalized standard deviation of the throughput samples taken by the two runtime environments when running PHOLD, considering different sampling periods for 64, 256 and 1024 LPs. We computed the normalized standard deviation for each number of active threads (from 1 to 40) with fixed P-state over multiple hundreds of samples and calculated the average. The goal of this study is to investigate the variability of the throughput measurements of the samples for the two runtimes with both a static configuration and a static workload. We recall that the power capping technique relies on the accuracy of the throughput samples acquired during the exploration procedure to select the configuration that maximizes the performance. We have varied the sampling period from 0.05 to 2 seconds. A higher period is impractical: in ROOT-Sim it corresponds to the GVT computation, which directly triggers other fundamental operations, such as fossil collection.

As expected, the normalized standard deviation decreases as the size of the sampling period increases. Longer intervals cover an overall higher percentage of the execution time, which is generally very similar when considering multiple executions of the same static workload with the same configuration. In USE, we observe significantly lower values than ROOT-Sim. This is related to the fact that the implementations of the two runtime environments are significantly different. In particular, USE relies on a single shared event pool which is accessed in a non-blocking fashion. This allows to avoid costly synchronization protocols to determine the GVT (and hence the throughput), because the value corresponds to the head element in the queue. Converesely, ROOT-Sim relies on a traditional consensus protocol to determine the GVT (and hence the number of committed events). Moreover, the need to rely on multiple queues in ROOT-Sim determines increased costs when the number of LPs varies, which depend on message-exchange patterns, rollback dynamics, and coasting forward effects for state reconstructions—this is confirmed by the fact that when running models with only 64 LPs the ratio (ROOT-Sim deviation/USE deviation) is 1.44, while with 1024 LPs the ratio is 32.64. All these aspects determine a stabler behavior in USE. Overall, this means that traditional multi-queue implementations for Time Warp simulation engines easily violate the heuristic assumption that, given a certain configuration, throughput estimation is kept stable.

In Figure 3 we provide experimental data related to the rebinding operation in ROOT-Sim. As mentioned, this operations is triggered for two purposes in our experiment: on the one hand, we keep the original goal of balancing at runtime the load of the different Processing Elements, to reduce the rollback probabilty. On the other hand, rebinding is required every time that the heuristic decides upon the (de)activation of some worker thread. In our implementation, this operation is triggered at every new GVT computation, which is the time instant at which a thread is possibly (de)activated. By the results, where we assume to change the number of threads at each GVT, we can observe that when the GVT computation is triggered often (e.g., every 200 milliseconds), the impact of the rebinding operation is as high as 25% of the overall execution time. The impact grows linearly with the number of active threads, due to the coordination nature of this operation, which requires at least one barrier among all the threads, to avoid that an LP is concurrently scheduled by multiple threads. Similarly, there is a proportional growth in the cost with the number of LPs, which is also expected due to the fact that the rebinding operation has to estimate the future workload, to perform an even rebalancing. This result emphasizes that, to allow for dynamic management of threads for energy optimization, this operation requires careful attention, not to hamper the overall performance.

Figure 4 shows the effectiveness of the power capping technique (referred to as DVFS+TS, where TS stands for thread scheduling) for ROOT-Sim, considering different values of the sampling interval for PHOLD with 64, 256 and 1024 LPs and two different power caps (55 and 65 Watt). These results are compared to an oracle, which selects the configuration that provides the maximum performance within the cap, and a technique referred to as DVFS that only changes P-state to meet the cap while keeping all the threads running. The latter does not rely on throughput samples but only on power consumption samples which are generally stable for intervals in the tens of milliseconds or higher. Both DVFS+TS and DVFS rely on a exploration procedure followed by an exploitation phase. We

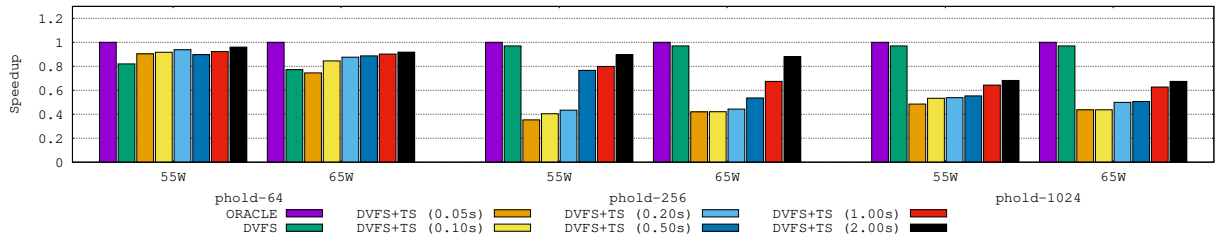


Figure 4: Performance Results using ROOT-Sim with different duration of the sampling interval.

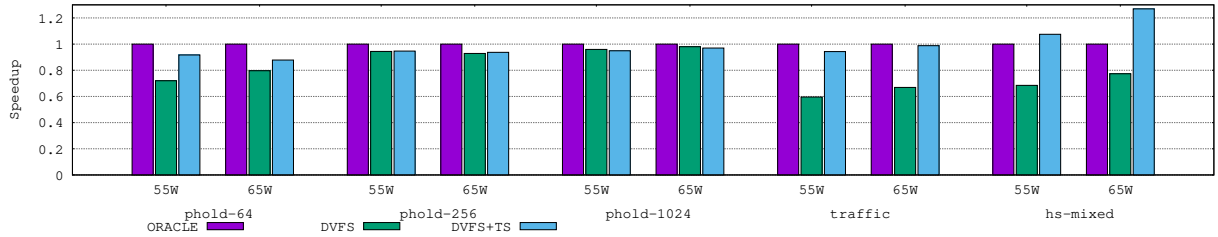


Figure 5: Performance Results using USE with 0.2 seconds sampling interval.

set the duration of the exploitation phase dynamically such that the percentage of time spent in the exploration is equal to 10% of the overall execution time. We recall that during the exploration the application is still generating progress, and that the techniques tends to visit configurations that are close to the optimum, which limits the overall exploration overhead. The percentage of power cap violation is insignificant for both DVFS and DVFS+TS as they both reduce core frequency whenever the cap is exceeded.

PHOLD with 256 and 1024 LPs are very scalable workloads as their optimal configurations for both 55 watt and 65 watt use 40 active threads at scaled frequencies. These workloads are very favorable to the DVFS-only technique—which obviously selects the optimal configurations—showing only minor slow-downs compared to the oracle due to the rebinding and exploration overheads. According to the data shown in Table 1, the variance of the throughput samples for these models is very significant, which makes it unable to find the optimal configuration for any duration of the sampling interval. As expected, with higher duration of the sampling interval and higher accuracy of the measurements, the performance of DVFS+TS gets closer to optimum, despite not reaching it. Conversely, the low variability of the samples for PHOLD on ROOT-Sim with 64 LPs allows the power capping technique to find the optimum when the duration of the sampling interval is equal to 0.2 seconds or higher. The DVFS technique shows a significant slowdown compared to the optimum, which selects 20 threads with P-state 8 and 3 for 55 Watt and 65 Watt respectively. Overall, these results show that a non-stable throughput, typical of traditional Time-Warp runtimes, is a limitation to the adoption of autonomic energy-aware self-tuning schemes.

Conversely, Figure 5 shows the results on USE considering a fixed sample interval of 0.2 seconds, which offers a sufficient accuracy for all the considered workloads. The technique always manages to find the optimum, thus matching the DVFS-only technique in its favorable workloads, while also being very close to the oracle whenever a lower number of active threads at higher frequency can be more beneficial to performance. The overhead is also lower compared to ROOT-Sim, as USE does not require the rebinding of LPs when changing the number of active threads. In Figure 5 we also provide the results with traffic, where the high contention leads to very poor performance for the DVFS technique, while DVFS+TS reaches the performance level of the oracle. Morever, in the hotspot configuration (hs-mixed in the plots) the dynamic technique manages to follow the variability of the workload, offering up to 26% increased performance compared to the best static configuration.

Table 2: PHOLD: Speedup with 40 threads wrt to Sequential.

	60 LP	256 LP	1024 LP
ROOT-Sim	5.17	13.37	36.27
USE	8.86	16.96	38.43

Finally, to show that we have been discussing competitive parallel implementations, in Table 2 we provide the speedup over a sequential simulation built relying on a fast Calendar Queue-based scheduler [3] of the PHOLD model, when using all 40 cores.

5 DISCUSSION AND FUTURE WORK

In this work, we have provided experimental evidences of the behavior of state-of-the-art PDES runtime environments, when trying to maximize their performance under a power cap. We have relied on a technique [9] that explores a sub-set of the configurations of number of active threads and P-States to find the configuration that provides the maximum performance under the power cap. However, the optimality of the technique depends on the accuracy of the samples acquired during the exploration. Our results attest the benefit of this approach from an energy and performance point of view. Beyond the numerical results, we can draw two different conclusions, which call for an additional research effort in the upcoming

years to allow runtime environments to benefit even more from this strategy, also in preparation for exascale computing systems.

We have emphasized the need for loose coupling between LPs and processing elements. While this conclusion has already been discussed in the literature (see, e.g., [23, 33]), we have shown to what extent it is fundamental when dealing with energy-efficient simulation runs. Indeed, as we have shown, being able to promptly change the number of active threads at runtime is mandatory to identify configurations which can provide effective speedups under an imposed power cap. While we have shown that effective implementations of the rebalancing operation for traditionally-organized runtime environments exist [33], this is a research topic which well deserves attention, also trying to identify mixed configurations between share-everything and share-nothing runtime environments.

At the same time, we have shown that the greatest impairment to enacting effective energy-efficient configurations in traditional PDES runtime environments comes from the extremely slow rate at which accurate throughput measurement can be provided. Indeed, traditional runtime environments can accurately provide such a measurement only when a new GVT value is computed. Increasing the frequency of such a coordination protocol could be detrimental for the performance, making any optimization strategy based on exploration ineffective. Different approaches, such as the share-everything one, allow to rely on much more accurate and stable measurements. Nevertheless, by design, such runtime environments cannot be immediately ported to distributed architectures, thus making extremely large models intractable. Moreover, our throughput metric relies on both event granularity and workload being constant during exploration.

There is therefore an urgent need to either identify technical solutions to bring the share-everything paradigm to distributed environments, without violating the assumptions this paradigm has been built upon, or to identify new accurate metrics to estimate the throughput, also when committed information is not available.

We claim that if all the above aspects are not properly addressed, then it will be extremely difficult to jointly meet performance and energy-efficiency goals in the upcoming years.

REFERENCES

[1] Laurent R G Auriche, Francesco Quaglia, and Bruno Ciciani. 1998. Run-time selection of the checkpoint interval in time warp based simulations. *Simulation Practice and Theory* 6, 5 (1998), 461–478.

[2] Aradhya Biswas and Richard Fujimoto. 2018. Zero energy synchronization of distributed simulations. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*.

[3] Randy Brown. 1988. Calendar queues: a fast O(1) priority queue implementation for the simulation event set problem. *Commun. ACM* 31, 10 (1988), 1220–1227.

[4] Christopher Burdorf and Jed B. Marti. 1990. *Non-Preemptive Time Warp Scheduling Algorithms*. Technical Report. RAND Corporation.

[5] Christopher D Carothers, David W Bauer, and S Pearce. 2000. ROSS: a High Performance Modular Time Warp System. In *Proceedings of the 14th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 53–60.

[6] Christopher D Carothers and Richard M Fujimoto. 2000. Efficient execution of Time Warp programs on heterogeneous, NOW platforms. *IEEE Transactions on Parallel and Distributed Systems* 11, 3 (2000), 299–317.

[7] Davide Cingolani, Alessandro Pellegrini, and Francesco Quaglia. 2017. Transparently Mixing Undo Logs and Software Reversibility for State Recovery in Optimistic PDES. *ACM Transactions on Modeling and Computer Simulation* 27, 2 (may 2017), 1–26.

[8] Stefano Conoci, Davide Cingolani, Pierangelo Di Sanzo, Alessandro Pellegrini, Bruno Ciciani, and Francesco Quaglia. 2018. A Power Cap Oriented Time Warp Architecture. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM.

[9] Stefano Conoci, Di Sanzo Pierangelo, Ciciani Bruno, and Quaglia Francesco. 2018. Adaptive Performance Optimization under Power Constraint in Multi-thread Applications with Diverse Scalability. In *Proceedings of the 9th ACM/SPEC International Conference on Performance Engineering*.

[10] Samir R Das and Richard M Fujimoto. 1997. An Empirical Evaluation of Performance-Memory Trade-Offs in Time Warp. *IEEE Transactions on Parallel and Distributed Systems* 8, 2 (1997), 210–224.

[11] Phillip M. Dickens, David M. Nicol, Paul F. Reynolds, and J. M. Duva. 1996. Analysis of bounded time warp and comparison with YAWNS. *ACM Transactions on Modeling and Computer Simulation* (1996).

[12] Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2012. Dark Silicon and the End of Multicore Scaling. *IEEE Micro* 32, 3 (may 2012), 122–134.

[13] Josef Fleischmann and Philip A. Wilsey. 1995. Comparative Analysis of Periodic State Saving Techniques in Time Warp Simulators. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 50–58.

[14] Richard M Fujimoto. 1990. Performance of Time Warp Under Synthetic Workloads. In *Proceedings of the Multiconference on Distributed Simulation*. Society for Computer Simulation, 23–28.

[15] Richard M. Fujimoto. 2019. Power consumption in modelling and simulation. *Journal of Simulation* (oct 2019), 1–12.

[16] Richard M. Fujimoto, Michael Hunter, Aradhya Biswas, Mark Jackson, and Sabra Neal. 2017. Power efficient distributed simulation. In *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*.

[17] Anurag Gupta, Ian F. Akyildiz, and Richard M. Fujimoto. 1991. Performance analysis of time warp with multiple homogeneous processors. *IEEE Transactions on Software Engineering* (1991).

[18] Mauro Ianni, Romolo Marotta, Davide Cingolani, Alessandro Pellegrini, and Francesco Quaglia. 2018. The Ultimate Share-Everything PDES System. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM Press, 73–84.

[19] David R Jefferson. 1985. Virtual Time. *ACM Transactions on Programming Languages and System* 7, 3 (1985), 404–425.

[20] Kevin Jones and Samir R. Das. 1998. Combining optimism limiting schemes in time warp based parallel simulations. In *Winter Simulation Conference Proceedings*.

[21] Boris Lubachevsky, Adam Schwartz, and Alan Weiss. 1991. An Analysis of Rollback-Based Simulation. *ACM Transactions on Modeling and Computer Simulation* (1991).

[22] Dale E Martin, Timothy J McBrayer, and Philip A. Wilsey. 1996. WARPED: A Time Warp simulation kernel for analysis and application development. In *Proceedings of the 29th Hawaii International Conference on System Sciences. Volume 1: Software Technology and Architecture*. IEEE Computer Society, 383–386.

[23] Eric Mikida and Laxmikant Kale. 2018. Adaptive methods for irregular parallel discrete event simulation workloads. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*.

[24] Avinash C Palaniswamy and Philip A. Wilsey. 1994. Scheduling Time Warp Processes Using Adaptive Control Techniques. In *Proceedings of the 1994 Winter Simulation Conference*. Society for Computer Simulation, 731–738.

[25] Alessandro Pellegrini and Francesco Quaglia. 2017. A Fine-Grain Time-Sharing Time Warp System. *ACM Transactions on Modeling and Computer Simulation* 27, 2 (jul 2017), 1–25.

[26] Alessandro Pellegrini, Roberto Vitali, and Francesco Quaglia. 2011. The ROme OpTimistic Simulator: Core internals and programming model. In *Proceedings of the 4th ICST Conference of Simulation Tools and Techniques*. ACM, 96–98.

[27] Alessandro Pellegrini, Roberto Vitali, and Francesco Quaglia. 2015. Autonomic State Management for Optimistic Simulation Platforms. *IEEE Transactions on Parallel and Distributed Systems* 26, 6 (2015), 1560–1569.

[28] Patrick Putnam, Philip A. Wilsey, and Karthik Vadambacheri Manian. 2012. Core frequency adjustment to optimize Time Warp on many-core processors. *Simulation Modelling Practice and Theory* (2012).

[29] Francesco Quaglia and Vittorio Cortellessa. 2002. On the processor scheduling problem in time warp synchronization. *ACM Transactions on Modeling and Computer Simulation* 12, 3 (jul 2002), 143–175.

[30] Robert Rönngren and Rassul Ayani. 1994. Adaptive Checkpointing in Time Warp. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation*. Society for Computer Simulation, 110–117.

[31] Tapas K Som and Robert G Sargent. 1998. A Probabilistic Event Scheduling Policy for Optimistic Parallel Discrete Event Simulation. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*. IEEE, 56–63.

[32] Seng Chuan Tay, Yong Meng Teo, and Siew Theng Kong. 1997. Speculative parallel simulation with an adaptive throttle scheme. In *Proceedings of the 11th Workshop on Parallel and Distributed Simulation*.

[33] Roberto Vitali, Alessandro Pellegrini, and Francesco Quaglia. 2012. Load sharing for optimistic parallel simulations on multi core machines. *ACM SIGMETRICS Performance Evaluation Review* 40, 3 (jan 2012), 2–11.

[34] Roberto Vitali, Alessandro Pellegrini, and Francesco Quaglia. 2012. Towards Symmetric Multi-threaded Optimistic Simulation Kernels. In *Proceedings of the 26th Workshop on Principles of Advanced and Distributed Simulation*, 211–220.