Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

USING DVFS TO OPTIMIZE TIME WARP SIMULATIONS

Ryan Child
Philip A. Wilsey

University of Cincinnati
School of Electronic and Computing Systems
Cincinnati, OH 45221-0030, USA

ABSTRACT

Some emerging high performance many-core chips have support to enable software control of an individual core's operating frequency (and voltage). These controls can potentially be used to optimize execution for either performance (accelerating the critical path) or power savings (green computing). In Time Warp parallel simulators using the Virtual Time synchronization paradigm, some cores may be executing events that are well off the critical path and likely to be undone. In this work, we explore the adjustment of operating frequencies of cores executing on and off the critical path to reduce rollback and power consumption, while maintaining or, in some cases, enhancing performance.

1 INTRODUCTION

Although raw performance is the main goal of Parallel Discrete Event Simulation (PDES) (Fujimoto 1990a), power consumption has become a concern due to factors such as operational cost, component lifetime, and the environment. Virtually all modern single Chip Multiprocessors (CMPs) include features that help to reduce power consumption. One of these features, Dynamic Voltage and Frequency Scaling (DVFS), provides exciting opportunities for PDES, both in terms of speedup and power consumption.

DVFS achieves power savings by scaling the voltage and clock frequency of the CPU. Equation (1) shows that dynamic power is proportional to clock frequency multiplied by voltage squared. A lower clock frequency makes lower voltages possible, and since reductions in frequency are usually accompanied by mostly proportional changes in voltage, DVFS can potentially reduce power by the cube of voltage reduction. This is known as the "cube-root" rule (Brooks et al. 2000).

$$P_{dynamic} \propto CV^2 f \tag{1}$$

It has been shown that DVFS techniques on clusters can significantly reduce energy at little or no performance cost if the load is imbalanced and the critical path is not slowed down (Rountree et al. 2009). Intel Turbo Boost and AMD Turbo CORE are hardware implementations that take this approach. Turbo Boost independently boosts the clock frequency of one or more cores when they are under heavy load, while Turbo CORE simultaneously boosts up to three cores. Both of these technologies attempt to simultaneously optimize for performance and energy. This is possible because over-utilized cores are over clocked only if other cores are under clocked, resulting in approximately the same power. At the same time, the task is completed sooner, meaning better performance and less energy consumed. Because technologies such as Turbo Boost and Turbo CORE dynamically identify a load imbalance and accelerate the critical path, they can be understood as a form of load balancing.

Like many parallel systems, Time Warp can become imbalanced which negatively affects simulation performance. The primary difference between traditional load balancing and load balancing in Time Warp is the metric used to determine loading. In traditional load balancing scenarios, raw CPU utilization determines relative load. In Time Warp, the concept of utilization is entirely different. Time Warp aggressively uses

all available CPU resources throughout the duration of a simulation, so it must instead use the concept of "useful work" (Palaniswamy and Wilsey 1996) to detect and balance the critical path. Turbo Boost and Turbo CORE are unable to detect imbalances in Time Warp simulations and thus may not be able to correctly detect and accelerate the critical path.

One of our goals is to show that both performance and energy savings could be attainable in Time Warp on chip multiprocessors (CMPs) if technology such as Turbo Boost and Turbo CORE could be controlled from software such that useful work could be used as a metric for detecting load imbalance. Our second goal is to explore the possibility of using DVFS to further increase energy savings in Time Warp both on CMPs and clusters by scaling down the frequency of cores off the critical path.

The remainder of this manuscript is organized as follows. Section 2 provides background information on Time Warp synchronized parallel simulation and on Dynamic Voltage and Frequency Scaling (DVFS) in modern processors with a quick review of the corresponding specifications in the ACPI standard that allow operating systems to utilize DVFS. Section 3 describes and compares possible metrics of useful work of the parallel executing Logical Processes (LPs) in a Time Warp synchronized simulator. Section 4 describes the DVFS algorithms that we have developed to balance the critical path using estimates of useful work. Section 5 gives the equations used to model the energy efficiency of Time Warp simulations under the control of our DVFS algorithms. Section 6 describes the experiments and the experimental results we obtained for this study. Finally, Section 7 contains some concluding remarks and suggestions for future research.

2 BACKGROUND

2.1 Time Warp

Time Warp is a form of PDES that optimistically executes events using Jefferson's Virtual Time paradigm for synchronization (Jefferson 1985, Fujimoto 1990a). Events are characterized by a send and receive time (timestamp) and are sent as messages between objects that model physical processes. These objects are grouped in to sets called Logical Processes (LPs), which in turn are mapped onto parallel execution units (PEs). The lowest timestamped event in the event queue of an LP is known as the LP's Local Virtual Time (LVT). The minimum LVT of all LPs in the simulation, together with the send times of any in-flight messages is known as the Global Virtual Time (GVT), and is used as a measure of simulation progress. Because events are optimistically executed and not guaranteed to be in their correct causal order, they must sometimes be undone and re-executed (if an event with an timestamp lower than LVT arrives; such events are called *straggler* events). When a straggler event arrives, the LP must recover by performing a *rollback* to a previously saved state that will allow the execution of the input events (including the straggler event) in their proper timestamp order. LPs that spend more time doing erroneous (or premature) computation can be said to be doing less *useful work*. Depending on network topology and the application model being simulated, it is possible that some LPs will be doing significantly more (or less) useful work than the others, resulting in an imbalanced simulation. Various adaptive techniques have been proposed to dynamically balance Time Warp simulations at runtime (Das 1996), many of which slow down the offending processes by blocking event execution for a period of time.

The DVFS approach described in this paper presents an alternative method for adaptive throttling of Time Warp. Scaling the frequency of the core executing an LP is most closely related to existing approaches that employ some form of real-time blocking, such as those described in Ball and Hoyt (1990), Ferscha and Lüthi (1995), Hamnes and Tripathi (1994), Srinivasan and Reynolds Jr. (1998), Quaglia (2001). Just as these approaches are orthogonal to other optimization techniques that focus on areas of Time Warp such as scheduling and checkpointing, so is the DVFS-based approach described in this paper. Child and Wilsey (2012) introduced the concept of DVFS-based tuning of Time Warp simulations. The present study is complementary in that *both* power consumption and performance are considered. Moreover, three different DVFS algorithms ("governors") are implemented and experimentally compared.

2.2 DVFS

Weiser was the first to investigate the use of DVFS in an operating system scheduler to decrease power consumption and show that huge energy savings are possible (Weiser et al. 1994). Today, DVFS is ubiquitous and is used by operating systems to boost performance and conserve power. DVFS features are highly device-specific, so contemporary operating systems running on x86 hardware interface with DVFS mechanisms in hardware through the *Advanced Configuration and Power Interface* (ACPI) (Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation 2010).

ACPI defines a maximum of 16 discrete frequency-voltage pairs called performance states (P-States). The P-states are labeled `P0` $\cdots$ `Pn` (in *decreasing* order of frequency and voltage) and they are made available to the operating system via tables in firmware. P-states are related to, but different than, the Intel Turbo Boost and AMD Turbo CORE technologies discussed in Section 1. These technologies will temporarily over clock some of the chip cores when others are idle. They are implemented in hardware and may take effect when one or more cores is set to state P0. P-state definitions, along with the rest of the ACPI specification, are architecture-independent, allowing implementations on a range of devices.

The number of available P-states in a chip is limited due to complex timing constraints between various frequency and voltage domains on the chip. Typically, only four to six P-states are made available. To further simplify the design, most microprocessors require coordination of P-states between cores (*i.e.,* they are not fully independent). The 4-core Intel Core i7 and the ARM-based dual-core TI OMAP4460 require P-State coordination, while the six-core AMD Phenom II X6, the twelve-core Opteron 6168, and the 8-core IBM Power 7 do not. Per-core DVFS may become more prevalent in the future, as the Intel Single-Chip Cloud Computer (SCC) research chip provides 24 independently-configurable frequency islands (Labs 2010). In this study, we perform our experiments with the Linux operating system and so we will briefly review the control of P-state in Linux.

The Linux kernel subsystem responsible for dynamically setting the P-states of the CPU cores is known as `cpufreq` (Hopper 2009). It includes several in-kernel "governors" — daemons that detect workloads and set P-states according to some power and performance goals. The current available governors in the Linux kernel are `userspace`, `ondemand`, `performance`, `powersave` and `conservative`. The `userspace` governor is of the most interest to us as it allows userspace implementation of DVFS algorithms. It has been used for other DVFS research, most recently to study the effect of DVFS in High Performance Computing (HPC) environments (Etinski et al. 2012, Rountree et al. 2009).

3 USEFUL WORK METRICS

Because in Time Warp all PEs have an equally high raw CPU utilization, we need to use a different metric to identify the critical path of the simulation. While there are many possible choices, a good candidate must be both accurate and inexpensive to compute. It should be noted that useful work can only be estimated dynamically at run-time, not calculated. This is because it is not known at the time of event execution whether the event will be rolled back. Useful work is estimated based on the measurement of recent behavior.

Many metrics have been proposed in the literature (Ball and Hoyt 1990, Palaniswamy and Wilsey 1996, Fleischmann and Wilsey 1995, Reiher and Jefferson 1990, Tay, Teo, and Kong 1997). Since rollbacks occur because certain LPs advance too far ahead of other LPs in virtual time, it is natural to consider LVT in determining useful work. Indeed, this has been used as a metric by Tay, Teo, and Kong (1997) to implement an adaptive throttling scheme. In their scheme, event scheduling is throttled based on distance from GVT. However, using LVT for this purpose can unintentionally prevent the simulator from uncovering hidden parallelism in the application model (Palaniswamy and Wilsey 1996). For example, consider a simulation with two LPs executing on separate PEs. Assume one LP has a much higher LVT than the other. If a rollback occurs, there must be a difference in LVT between the two LPs. However, the converse

is not necessarily true — the difference in LVT itself does not guarantee a rollback. If the application model contains sufficient parallelism, the two LPs could be executing along two parallel critical paths. Slowing down the LP with the higher LVT would then restrict the parallelism and thus slow down the overall simulation.

In the experiments reported in this paper, we have implemented and tested 3 different "useful work" estimation algorithms. Each algorithm estimates useful work based on a different metric. We refrain from using virtual time information, in order to circumvent the problems discussed above.

This first metric is based on "effective utilization," defined by Reiher and Jefferson (1990) as the fraction of work on a given node that will not be rolled back. The work performed while executing events is obtained in the form of CPU cycles by examining the `rdtsc` time stamp counter register. The use of this register is generally discouraged in modern multi-core CPUs because of the possibility of migration to other cores by the operating system and frequency switching during the lifetime of a process. However, in our case, each LP is bound to a specific core and frequencies are switched at a relatively slow rate, thus the use of the `rdtsc` register is acceptable. If an event is rolled back at a later time, the CPU cycles measured at its time of execution are subtracted from the effective work estimator. At each frequency adjustment interval, the number of cycles that were not undone is divided by the total number of cycles spent in execution to give an effective utilization number that is generally between 0 and 1. In some rare cases, effective utilization may be negative if many events that took a large number of cycles are rolled back at the beginning of a frequency adjustment interval and the interval is small (although this was not observed in our experiments). The effective utilization equation is given in (2), where $C_{executed}$ represents the number of CPU cycles spent executing events and C_{undone} is the number of CPU cycles spent rolling back events. Both these quantities are reset to 0 after each measurement of effective utilization.

$$EffectiveUtilization = \frac{C_{executed} - C_{undone}}{C_{executed}} = 1 - \frac{C_{undone}}{C_{executed}} \tag{2}$$

The second metric of useful work that we implemented and tested is called number-of-rollbacks. It is defined as the number of rollbacks that occur during a measurement cycle. This estimator is derived from observations made by Palaniswamy (Palaniswamy and Wilsey 1996). In his work, Palaniswamy showed that useful work is generally proportional to the number of times an LP rolls back. A rollback occurs when a "straggler" event arrives at an LP having an LVT that is greater than the event timestamp. It is associated with a virtual time span during which any number of events could have been incorrectly executed, however the number-of-rollbacks metric does not take this fact into account. Unlike effective utilization, this metric can be measured to be arbitrarily high, which makes the problem of classification slightly more complicated than with effective utilization. Note that a fourth metric could be derived to solve this problem simply by normalizing against the number of events received. We limit this study to three metrics, as the primary focus of this paper is DVFS-based optimization.

The third and final metric that we studied is "efficiency," defined by Equation (3). In this equation, $E_{executed}$ represents number of *event executions*, while E_{undone} represents the number of *event rollbacks*. $E_{executed}$ and E_{undone} are reset to 0 every time efficiency is measured. When used as a useful work metric at various points during a simulation, efficiency is an *estimate of the fraction of event executions that will be committed*. Specifically, the number of committed events during a measurement cycle is estimated to be $E_{executed} - E_{undone}$, as shown in the numerator of (3). Unlike the number-of-rollbacks metric, efficiency *does* take into account the number of events undone during each rollback, and is only slightly more expensive to compute. Similar to effective utilization, efficiency is generally a fraction between 0 and 1, although it can theoretically be negative if more events are undone than executed during a frequency adjustment interval. Note that efficiency and effective utilization are equal if the grain of computation is equal for all events.

$$Efficiency = \frac{E_{executed} - E_{undone}}{E_{executed}} = 1 - \frac{E_{undone}}{E_{executed}} \tag{3}$$

```
 1. while there exists a pair (LP_H,LP_L)
    with useful work (above,below) threshold:
 2.   while LP_H is at maximum frequency:
 3.     LP_H := LP with next lower useful work index
 4.   end while
 5.   while LP_L is at the minimum frequency:
 6.     LP_L := LP with next higher useful work index
 7.   end while
 8.   if (LP_H,LP_L) still have useful work (above,below) threshold:
 9.     adjust the frequency of LP_L down
10.     adjust the frequency of LP_H up
11.   end if
12. end while
```

Figure 1: Time Warp Performance governor algorithm.

```
 1. while there exists a pair (LP_H,LP_L)
    with useful work (above,below) threshold:
 2.   while LP_L is at the minimum frequency:
 3.     LP_L := LP with next higher useful work index
 4.   end while
 5.   if LP_L still has useful work below threshold:
 6.     adjust the frequency of LP_L down
 7.   end if
 8. end while
```

Figure 2: Time Warp LowPower governor algorithm.

4 TIME WARP DVFS GOVERNORS

Using the Linux `userspace` kernel governor, we implemented a number of our own "Time Warp governors" that dynamically set the P-states of each core based on the useful work of the LP that is bound to it. In particular we implement three distinct governors that we have named *Performance*, *LowPower*, and *Hybrid*. Each governor employs a slightly different algorithm depending on the optimization goal, further discussed below. In each algorithm, each LP number and its useful work index have been collected into an array of records and sorted by useful work index in ascending order. The useful work metric is selectable among the three discussed in the previous section.

The Performance governor (Figure 1) only scales the frequency of one LP up if it can find another LP that it can scale down. The LowPower governor (Figure 2) aggressively scales down the frequencies of LPs with a low useful work index, regardless of whether there is another LP it can scale up. The Hybrid governor (Figure 3) attempts to combine the Performance and LowPower governors. It operates by first scaling down the frequency of an LP with a below-average useful work index, and then scales up an LP with an above-average useful work index if one exists. The upper and lower thresholds of useful work depend on the metric. For the effective utilization and efficiency metrics, the thresholds are the average useful work across all LPs $\pm$ 0.1. For the number-of-rollbacks useful work metric, they are the average $\pm$ 10%.

```
 1. while there exists a pair (LP_H,LP_L)
    with useful work (above,below) threshold:
 2.   while LP_H is at maximum frequency:
 3.     LP_H := LP with next lower useful work index
 4.   end while
 5.   while LP_L is at the minimum frequency:
 6.     LP_L := LP with next higher useful work index
 7.   end while
 8.   if LP_L has useful work below threshold:
 9.     adjust the frequency of LP_L down
10.     if LP_H has useful work above threshold:
11.       adjust the frequency of LP_H up
12.     end if
13.   end if
14. end while
```

Figure 3: Time Warp Hybrid governor algorithm.

5 MODELING TIME WARP ENERGY USAGE

Power consumption in VLSI circuits is divided into three components: *dynamic*, *static* (or leakage), and *direct path*. These three components sum to equal the total device power, given by Equation (4) (Rabaey 1996, p. 225). Dynamic power consumption results from energy lost while charging and discharging capacitances during $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions. Static power dissipation results from leakage currents from V_{DD} to ground, and has become a significant part of the power equation with today's nanoscale transistors and near-threshold voltages. Direct path power results from short circuit current that occurs in the brief partially-on partially-off state between transitions.

$$P_{total} = P_{dynamic} + P_{leakage} + P_{direct\,path} \tag{4}$$

It is common practice to neglect everything but the dynamic power (Brooks et al. 2000). We have no way of measuring leakage power on the Phenom II X6 1055T, but we can assume that the operating points designed by AMD (Table 2) were chosen such that leakage power is kept under control. Indeed, the minimum operating voltage of 1.18V in P3 seems to be sufficiently above the typical threshold voltage (a few hundred millivolts) of a 45nm CMOS process. Static power is neglected in our calculations for these reasons. Direct path power is also neglected as it is the smallest part of the power equation and is easily kept under control with careful design (Rabaey 1996, p. 225).

Dynamic power across an N-core CMP is approximated by Equation (5). C is the output capacitance of all transistors on one core (assumed to be approximately equal across all cores), a is an activity factor that represents the switching frequency of transistors on the core, V is CPU voltage (V_{DD}), and f is clock frequency (Brooks et al. 2000).

$$P_{dynamic} \approx \frac{aC}{2} \sum_{i}^{N} V_i^2 f_i \tag{5}$$

We multiply the total dynamic power given by Equation (5) by the simulation runtime (delay D) to obtain an approximation of the energy used by the simulation. This approximation is given in Equation (6).

$$E \approx P_{dynamic} \left(t_{stop} - t_{start}\right) \tag{6}$$

$$= P_{dynamic} D \tag{7}$$

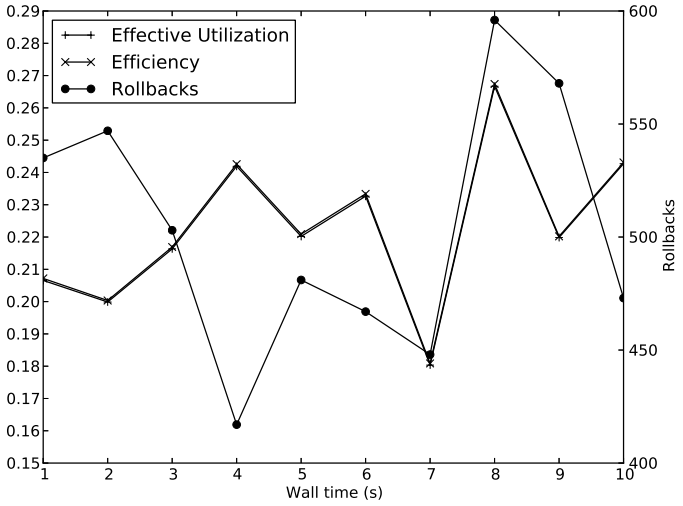

Figure 4: Comparisons of the number-of-rollbacks, effective utilization, and efficiency useful work metrics.

6 EXPERIMENTS

The WARPED Time Warp simulator is used as a platform to implement and evaluate the DVFS algorithms and useful work estimators discussed in this paper. As WARPED was originally designed for use on distributed memory clusters, it was upgraded to use the Nemesis communication channel of MPICH2, an MPI implementation that uses non-blocking shared memory queues for intra-node message passing (Buntinas, Mercier, and Gropp 2006). An LP in WARPED is a Linux heavyweight thread that contains any number of simulation objects and maintains a single Least Time Stamp First (LTSF) queue of events for all of its objects. Four and six-LP WARPED simulations were run on a custom PC equipped with an AMD Phenom II X6 1055T CPU and 16GB RAM.

First, the number-of-rollbacks, effective utilization, and efficiency useful work metrics were evaluated using the parallel hold (PHold) simulation model (Fujimoto 1990b). Figure 4 shows a trace of a PHold simulation with 2 LPs each bound to a core of the X6, 6 simulation objects (3 per LP) and a message density of 32. This is a single simulation run in which all three useful work metrics are computed. In this case, the effective utilization and efficiency metrics are roughly the same. This is expected, as all events in the PHold model have a constant granularity. Any slight differences are likely due to `rdtsc` readings taken between context switches.

The number-of-rollbacks metric, however, is strikingly different than the other two. Until 5 seconds into the simulation, it appears to be proportional to the inverse of the other two metrics, as we would expect. However, from that point on it deviates from our expectations significantly. The measurement at time 7 could be explained by a single rollback causing a large number of events to be undone (a long rollback). At the same time, the measurement of a large number of rollbacks together with a high efficiency measurement at time 8 could be explained as either a sharp drop in the number of events executed that interval due to a sudden GVT computation, or a large number of rollbacks that only go back to the recent virtual past and therefore only undo the work of a few events.

Although the simplicity of the number-of-rollbacks metric is appealing, these results seem to be in favor of the efficiency and effective utilization metrics. Real application models in Time Warp have different types of events with different computation grains, so we would expect to see some deviation between the efficiency and effective utilization when running simulations of different models. For PHold, the efficiency metric is the clear winner, as it is less expensive to estimate.

Table 1: Steady-state frequencies arrived at by the Time Warp governors, in MHz.

Governor	f_{LP0}	f_{LP1}	f_{LP2}	f_{LP3}	f_{LP4}	f_{LP5}
Performance	2800	1500*	2200	2200	2200	2200
Low Power	2200	1500	1500	1500	1500	1500
Hybrid	2800	1500	1500	1500	1500	1500
Fixed	2200	2200	2200	2200	2200	2200

*Performance will set the LP with the smallest useful work index to 1500 MHz, which may not necessarily be LP1.

A modified version of PHold was used to evaluate and compare the performance of the Time Warp governors. The PHold model is an extension of the hold model used in sequential simulators, and is designed to test the spatial and temporal locality of Time Warp. The PHold model by itself, however, is completely balanced provided LPs are partitioned evenly across all available PEs, which made it nearly useless for the purposes of evaluating our load balancing DVFS governors. Fortunately Ahmed, Ronngren, and Ayani (1994) showed how to introduce imbalances into a PHold model. In particular, the authors used a modified version of PHold in scheduling experiments to model a heterogeneous system in which a "hot spot" node is N times more likely to receive an event than any other node. The use of a hot spot node is justified because it represents a type of model in which some LPs are more active than others, a property that is common in realistic models such as digital logic simulators (Ahmed, Ronngren, and Ayani 1994). Below, we use this idea to evaluate our Time Warp governors.

The hot spot node creates an imbalance in the simulation such that none of the governors are able to restore balance, meaning the P-states of each LP are already set to their final value within the first seconds of the simulation. The frequencies inevitably arrived at for a 6-LP simulation with LP0 configured as a hotspot are given in Table 1. In the case of the Performance governor, any one of the LPs other than LP0 could be selected for scaling down, depending on which has the absolute lowest useful work index. Initial results showed that WARPED under DVFS performed better with a restricted range of frequencies, so the available P-states for these experiments were limited to the first three, P0-P2. The frequencies and empirically obtained voltages of these P-states are given in Table 2.

Figure 5 compares the three Time Warp governors presented in this paper. The Fixed governor simply sets the frequencies of all cores to the nominal (2.2 GHz). Message densities (number of initial events per LP) were varied from 2 to 32, and the computation grain was set to 10μs. Each LP is bound to a core of the X6 and contains only one simulation object. Efficiency was used as the useful work metric. The frequency adjustment interval was 1 second. Four performance metrics commonly used to evaluate Time Warp were measured: *simulation time*, *event rate*, *efficiency*, and *number of rollbacks*. Simulation time is the wall time of the simulation run. Event rate is the number of committed events per second and is thus inversely related to simulation time. Here, as opposed to useful work estimation, efficiency is the *actual* number of committed events divided by the number of executed events throughout the whole simulation, and rollbacks gives the total number of rollbacks across all LPs. Each data point is the arithmetic mean of the results of 10 simulation runs.

Simulation time and number of rollbacks are the most affected by the DVFS algorithm and number of LPs. This dependence becomes more apparent at higher message densities. The Fixed governor has the highest efficiency of the 4-LP simulations at message densities less than 32.

It can be gathered from the simulation times in Figure 5 that the Performance and Hybrid governors were able to speed up simulations, at least for some message densities. We now turn our attention to quantifying these measurements in terms of energy savings and performance gains. Table 2 lists measured operating points of the AMD Phenom II X6 1055T, which were empirically obtained using the `sensors` tool of the `lm-sensors` package and the `cpufreq` userspace governor to set the P-states. This was necessary because the authors have so far been unable to obtain a datasheet. `cpufreq` also shows that the transition latency between P-states is 8μs for all operating points.

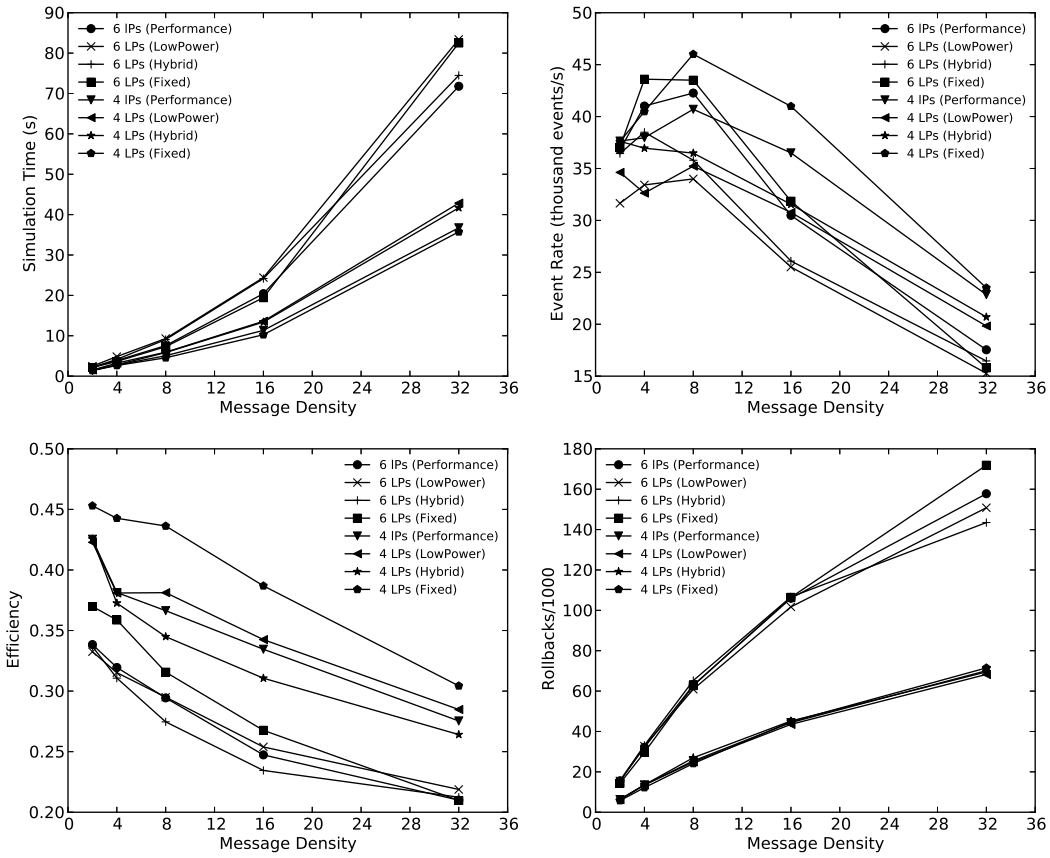

Figure 5: Simulation time, event rate, efficiency, and rollback comparison of Time Warp governors for 4 and 6 LPs. LP0 is configured as a hot spot node with N=10.

Table 2: Empirical AMD Phenom II X6 1055T Operating Points.

P-State	Voltage (V)	Frequency (MHz)	Used
P0	1.44	2800	Yes
P1	1.26	2200	Yes
P2	1.25	1500	Yes
P3	1.18	800	No

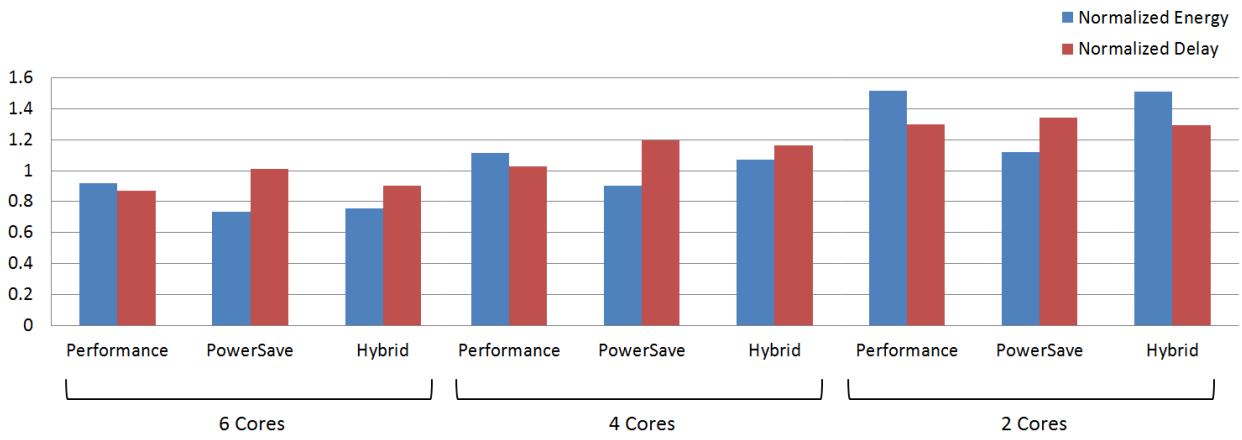

Figure 6: Energy and Performance of various Time Warp governors relative to the Fixed governor.

Normalized energy is given by Equation (8). To obtain normalized energy, Equation (6) was evaluated using the steady-state frequencies set by each governor (See Table 1) and divided by the energy of the fixed governor. Normalized energy is plotted alongside normalized delay for each Time Warp governor in Figure 6 for 2, 4 and 6 cores.

$$NormalizedEnergy_i = \frac{P_i D_i}{P_{fixed} D_{fixed}} \tag{8}$$

Figure 6 shows normalized (with respect to the Fixed governor) energies and delays for each of the other governors, for 6, 4, and 2 cores. It was generated from the same data plotted in Figure 5. Values less than 1 represent a performance/energy gain, while values greater than one represent a loss. For this energy-performance analysis, data points of message density 32 were taken, as those show the greatest variance between Time Warp governors, and also appear to be the most promising in terms of performance and energy savings.

The results show that for 6 cores, both the Performance and Hybrid governors are faster and more energy efficient than the Fixed governor. The Performance governor achieves slightly better performance but with significantly more energy than the Hybrid governor. As expected, the LowPower governor saves significant energy ($\sim 27\%$) with a minimal increase in simulation time ($\sim 1\%$). Also as expected, the performance of the Hybrid and Performance governors is roughly the same for 2 cores.

It was expected that fewer cores would result in less performance and energy savings, but curiously all three governors ran slower than the Fixed governor for 2 and 4 core simulations, which affected both performance and energy. The results appear to show that savings increases with number of cores, but it is unclear where savings will level off. Emerging many-core CMPs will provide the answer in the near future.

The Performance and Hybrid Time Warp governors represent theoretical gains if Turbo Boost-style over clocking were possible with software. In this scenario, the Fixed governor provides an artificial frequency ceiling, and Performance and Hybrid are replacements for the Turbo Boost algorithm. Performance and Hybrid show that performance and energy can be simultaneously optimized in Time Warp, but are not directly applicable to Time Warp simulations on current CMPs. The results of the LowPower governor, however, are more general and show that significant energy savings with minimal performance loss are achievable with DVFS in Time Warp, on CMPs and possibly on DVFS-enabled clusters. Note that the reported savings in performance and energy seem to be highly dependent on message density.

7 CONCLUSIONS AND FUTURE WORK

We have discussed Time Warp and the concept of useful work, and compared load balancing in Time Warp to traditional load balancing. Various metrics of useful work were compared. It was shown how estimates of useful work can be used to dynamically balance Time Warp simulations using DVFS, achieving faster performance, energy savings and fewer rollbacks. Unfortunately, gains appear to be highly sensitive to model parameters such as message density, as well as number of LPs.

It is known that CPU-boundness affects the sensitivity of application runtime to changes in clock frequency (Etinski et al. 2012), so it would be interesting to investigate how event granularity affects Time Warp simulations under DVFS. Also, we have demonstrated power savings with the LowPower on a shared-memory multi-core processor, but the same algorithm should theoretically be applicable to Time Warp on a DVFS-enabled cluster, by using `cpufreq` to raise the P-states of nodes with low useful work indexes. This is a possible avenue of exploration in future Time Warp research.

The hot spot variant of the PHold model with N=10 is extremely imbalanced and remains that way throughout the duration of the simulation if DVFS is not used. The DVFS algorithms presented here could be evaluated against other imbalanced models with dynamically changing workloads to test their ability to adapt. We also have not seen the effect of using DVFS with Time Warp on a CMP with more than 6 cores, where further energy savings might be possible.

Finally, the threshold of change from the useful work measurements used to trigger a frequency adjustment in this study were determined by comparing each measurement to the average. This approach can theoretically be problematic if the average is heavily skewed by an outlier. A possible solution that remains to be tested is setting the threshold to the average plus and minus the inverse standard deviation scaled by some constant factor.

ACKNOWLEDGMENTS

Support for this work was provided in part by the National Science Foundation under grant CNS–0915337.

REFERENCES

Ahmed, H., R. Ronngren, and R. Ayani. 1994, January. “Impact of event scheduling on performance of time warp parallel simulations”. In *Proceedings of the Twenty-Seventh Hawaii International Conference on Systems Sciences*, Volume 2, 455–462.

Ball, D., and S. Hoyt. 1990, January. “The Adaptive Time-Warp Concurrency Control Algorithm”. In *Distributed Simulation*, 174–177. Society for Computer Simulation.

Brooks, D. M., P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. 2000, November. “Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors”. *IEEE Micro* 20 (6): 26–44.

Buntinas, D., G. Mercier, and W. Gropp. 2006, may. “Design and evaluation of Nemesis, a scalable, low-latency, message-passing communication subsystem”. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, Volume 1, 10 pp. –530.

Child, R., and P. Wilsey. 2012, july. “Dynamically Adjusting Core Frequencies to Accelerate Time Warp Simulations in Many-Core Processors”. In *Principles of Advanced and Distributed Simulation, 2012. PADS ’12. ACM/IEEE/SCS 26th Workshop on*.

Das, S. R. 1996. “Adaptive protocols for parallel discrete event simulation”. In *Proceedings of the 28th conference on Winter simulation*, WSC ’96, 186–193. Washington, DC, USA: IEEE Computer Society.

Etinski, M., J. Corbalan, J. Labarta, and M. Valero. 2012, April. “Understanding the future of energy-performance trade-off via DVFS in HPC environments”. *J. Parallel Distrib. Comput.* 72 (4): 579–590.

Ferscha, A., and J. Lüthi. 1995, April. “Estimating Rollback Overhead for Optimism Control in Time Warp”. In *Annual Simulation Symposium*.

Fleischmann, J., and P. A. Wilsey. 1995, June. "Comparative Analysis of Periodic State Saving Techniques in Time Warp Simulators". In *Proc. of the 9th Workshop on Parallel and Distributed Simulation (PADS 95)*, 50–58.

Fujimoto, R. M. 1990a, October. "Parallel discrete event simulation". *Commun. ACM* 33:30–53.

Fujimoto, R. M. 1990b, January. "Performance of time warp under synthetic workloads". In *Proceedings of the SCS Multiconference on Distributed Simulation, vol. 22*, 23–28. Society for Computer Simulation.

Hamnes, D. O., and A. Tripathi. 1994, July. "Investigations in Adaptive Distributed Simulation". In *Proc. of the 8th Workshop on Parallel and Distributed Simulation (PADS 94)*, 20–23. Society for Computer Simulation.

Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation 2010. *Advanced Configuration and Power Interface Specification*. 4.0a ed. Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation.

Hopper, J. 2009, September. "Reduce Linux power consumption, Part 1: The CPUfreq Subsystem". Technical report, IBM.

Jefferson, D. 1985, July. "Virtual Time". *ACM Transactions on Programming Languages and Systems* 7 (3): 405–425.

Labs, I. 2010, May. "The SCC Platform". Technical report, Intel Corporation.

Palaniswamy, A., and P. A. Wilsey. 1996, September. "Parameterized Time Warp: An Integrated Adaptive Solution to Optimistic PDES". *Journal of Parallel and Distributed Computing* 37 (2): 134–145.

Quaglia, F. 2001. "A scaled version of the elastic time algorithm". In *Proceedings of the fifteenth workshop on Parallel and distributed simulation*, PADS '01, 157–164. Washington, DC, USA: IEEE Computer Society.

Rabaey, J. M. 1996. *Digital integrated circuits: a design perspective*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Reiher, P. L., and D. Jefferson. 1990. "Virtual Time Based Dynamic Load Management In The Time Warp Operating System". *Transactions of the Society for Computer Simulation* 7:103–111.

Rountree, B., D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch. 2009. "Adagio: making DVS practical for complex HPC applications". In *Proceedings of the 23rd international conference on Supercomputing*, ICS '09, 460–469. New York, NY, USA: ACM.

Srinivasan, S., and P. F. Reynolds Jr.. 1998, April. "Elastic time". *ACM Transactions on Modeling and Computer Simulation* 8 (2): 103–139.

Tay, S., Y. Teo, and S. Kong. 1997, June. "Speculative parallel simulation with an adaptive throttle scheme". In *Proc. of 11th Workshop on Parallel and Distributed Simulation (PADS97)*, 116–123.

Weiser, M., B. Welch, A. Demers, and S. Shenker. 1994. "Scheduling for reduced CPU energy". In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, OSDI '94. Berkeley, CA, USA: USENIX Association.

AUTHOR BIOGRAPHIES

RYAN CHILD is a graduate student at the University of Cincinnati School of Electronics and Computing Systems. He is currently researching DVFS techniques to optimize performance and power consumption of Time Warp Parallel Discrete Event Simulations on multi-core and distributed systems. His email address is ryan.child@gmail.com and his web page is http://homepages.uc.edu/~childrn.

PHILIP A. WILSEY is a professor in the School of Electronic and Computing Systems at the University of Cincinnati. His is an experimentalist working in parallel and distributed systems, embedded system, and point-of-care medical devices. He is currently studying the challenges of parallelism in multi-core and many-core platforms and is studying the optimization of virtualized Beowulf clusters composed of multi-/many-core processors to support efficient parallel execution of fine grained applications. His email address is wilseypa@gmail.com and his web pages are http://secs.ceas.uc.edu/~paw and http://github.com/wilseypa.