Chameleon: Adapting Throughput Server to Time-Varying Green Power Budget Using Online Learning

Chao Li[*†], Rui Wang[*‡], Nilanjan Goswami[*†], Xian Li[‡], Tao Li[†], and Depei Qian[‡]

[†]*Departement of Electrical and Computer Engineering*
University of Florida, U.S.A

[‡]*School of Computer Science and Engineering*
Beihang University, P.R.C

Abstract — Eco-friendly energy sources (i.e. green power) attract great attention as lowering computer carbon footprint has become a necessity. Existing proposals on managing green energy powered systems show sub-optimal results since they either use rigid load power capping or heavily rely on backup power. We propose *Chameleon*, a novel adaptive green throughput server. *Chameleon* comprises of multiple flexible power management policies and leverages learning algorithm to select the optimal operating mode during runtime. The proposed design outperforms the state-of-the-art approach by 13% on performance, improves system MTBF by 42%, and still maintains up to 95% green energy utilization.

Keywords: throughput server, green power, adaptation, learning

I. INTRODUCTION

Computer system inevitably enters the landscape of design for sustainability as the IT power footprint has become a global concern. According to the Uptime Institute, the 3-year energy expenditure has already exceeded the server equipment cost since 2012 [1]. If we continue to rely on conventional fossil fuel based electricity, a 1MW data center will cause over 10000 metric ton of CO_2 emissions annually [2]. Driven by the rising energy price and the warning of climate change, industry and academia alike are focusing more attention than ever on eco-friendly sources of power such as wind and solar energy.

There have been several emerging system designs that exploit renewable energy. For example, HP has announced its carbon-free data center prototype that is entirely powered by onsite green energy sources. Similarly, eBay is experimenting with a small data center that integrates a 665kW solar array. Recently, Apple has also patented several solar energy driven electronic devices and renewable energy power management circuitry. For these systems, minimizing power consumption is no longer the sole aim of the design. Instead, they emphasize efficient use of renewable energy resources for lowering the reliance on conventional utility power.

The time-varying nature of renewable power supplies poses unconventional challenges for system power management. Due to their sensitivity to power disturbances, computing systems must maintain a continuous balance between the fluctuating renewable power budget and the variable IT power demand. To cope with this issue, recent studies have been working on two power management approaches, which we refer to as either *energy-oriented* design or *performance-oriented* design. The former approach emphasizes matching load power demand to power budget to maximize the benefit of renewable energy usage. The later approach, on the other hand, leverages backup power (i.e., battery or utility grid) to maintain desired workload performance when green energy generation is inadequate. Nevertheless, neither approach provides desirable trade-off between workload performance and energy efficiency. When the green power drops significantly or become intermittently unavailable, the *energy-oriented* design often put computer system into low-power states and therefore compromises the performance. In contrast, *performance-oriented* design suffers from degraded energy utilization since the typical round-trip energy efficiency of lead-acid battery is only 75% [3].

Furthermore, biased power management schemes can cause various troubles in addition to the performance and efficiency problems. For example, an avid user of *energy-oriented* design can experience thermal cycling (TC) issue due to the excessive on/off power cycles during aggressive load matching [4]. This issue refers to repeated heating and cooling of sections of the microelectronics and can lead to permanent devices failures (e.g., cracks on circuit boards). On the other hand, heavily relying on energy backup to handle the stochastic workload surges and supply sags increases the required energy storage capacity, which adds to the overall cost of infrastructure.

In this study we tackle the above challenges using a unique hybrid power management strategy. *The main idea is to gracefully switch between the two mutually exclusive power management policies to boost system performance and enhance reliability without sacrificing overall renewable energy usage effectiveness.*

We propose *Chameleon*, an adaptive many-core system that explores intelligent adaptation of power management policy for maximizing the benefits of green energy usage. Our system comprises of multiple power management schemes and each scheme is referred to as a power management mode. During operation, *Chameleon* learns from the feedback-based system-environment interaction and selects the optimal power management mode based on the previous experience and the observed outcome. We term this power management design as *mode-switching power management* (MSPM).

The advantage of *Chameleon* is two-fold. First, as it integrates multiple power management schemes, it could overcome the shortcomings of relying on any single of them. Second, the online learning capability enables the system to better adapt to the diverse and complex operating environment.

This paper makes the following contributions:

- We propose novel *mode-switching power management* (MSPM) mechanism to harvest the benefits of different renewable power management schemes. It enables green servers to adapt themselves to the time-varying renewable power budget automatically and efficiently. We show that MSPM could achieve up to 95% energy utilization depending on the actual renewable energy resource availability.
- We explore learning algorithm to help *Chameleon* achieve high adaptability in various environment conditions. We show that *Chameleon server* outperforms the state-of-the-art design by 12.8% on system throughput. We also feed the system with load power variability information to make it reliability-aware. By curtailing unnecessary power tuning activities, *Chameleon* could increase the processor MTBF by 42% on average.

* Authors contribute equally

The rest of this paper is organized as follows. Section 2 introduces *Chameleon* design and proposes the *mode switching power management* scheme. Section 3 describes evaluation framework. Section 4 presents experimental results. Section 5 discusses related work and Second 6 concludes the paper.

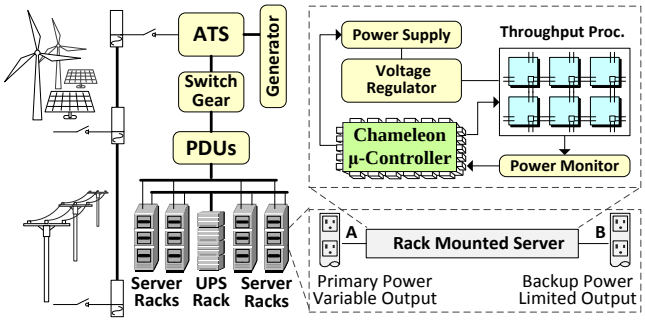

Figure 1: The *Chameleon* power management framework for server systems powered by intermittent green energy sources

II. Chameleon and the Mode-Switching Power Management

Chameleon is a power management framework for servers powered by emerging green energy sources. In particular, it targets throughput servers that are used for scientific computing or data processing jobs which typically do not have strict completion deadline. These servers often desire significant amount of time (days/weeks to run) and consume large amount of energy. Leveraging green energy to power these systems could greatly save conventional utility power bills and lower negative environmental impact.

A. Chameleon: Overview and Rationale

Figure 1 depicts the power provisioning hierarchy of a green datacenter across facility level and server level. At the facility level, onsite renewable energy generation (e.g., wind turbine or solar panel) and utility grid are connected together through appropriate circuit breakers and power conversion interfaces. An automatic transfer switch (ATS) can seamlessly switch the load to an onsite diesel generator in case that both renewable power system and utility grid fail. Eventually, power enters server racks through power distribution units (PDU). Each server rack is supported by distributed UPS batteries [5] for handling transient power interruptions. Such UPS placement topology has been adopted by Google and Facebook for improving energy conversion efficiency [5].

At the server level, each dual-corded server connects to two power strips (rack-level PDUs): one connects to the primary power and the other connects to local UPS batteries. Modern intelligent rack-level PDUs from HP and IBM have provided the capability of monitoring and managing the power budget on each individual power outlet. Normally, servers run at their designated speed if the rack-level PDU hasn't enforced power capping on the outlet. However, when renewable power supply fluctuates severely or becomes intermittently unavailable, it is crucial to change the power budget on each outlet and adjust the server power demand accordingly.

Chameleon provides a way to gracefully adapt server load to the time-varying green power budget. As shown in Figure 1, each server features a *Chameleon* micro-controller that interacts with the server power supply, throughput processor, and the power monitor. The controller enables the system to intelligently leverage energy resources (i.e., the variable renewable power budget and the limited stored energy) for a better tradeoff between efficiency and performance.

Chameleon does not emphasis using utility grid to provide backup energy due to three reasons. First, from a sustainability point of view, utility power yields low power efficiency (due to power transmission loss) and high carbon footprint. Second, in crowded urban areas utility power feeds are often at their capacity and electricity access for datacenter is restricted. Third, in remote areas or developing countries, utility grid is less reliable which in turn degrades the overall availability of grid-dependent system.

B. The Chameleon μ-Controller

Chameleon neither aggressively follows the variable power budget nor heavily borrows stored energy from the distributed UPS batteries. Instead, it combines the two approaches by defining multiple power management modes: *energy-oriented* control mode (*E-mode*) and *performance-oriented* control mode (*P-mode*). When *E-mode* is activated, the server gives high priority to primary power supply for maximizing the direct use of green energy. When *P-mode* is activated, the server yields high priority to stored energy (secondary power supply) for maintaining desired workload performance. Such approach is referred to as *mode-switching power management* (MSPM).

The *Chameleon* micro-controller is the major hardware addition in our design and the key component that supports MSPM. Figure 2 shows its internal architecture which consists of a *Chameleon* memory that stores important system profiling information, a mode tuning agent that manages the mode switching activities, a mode register that specifies current power management mode, and several pieces of firmware that execute actual load power control.

During runtime, the micro-controller performs three tasks concurrently: monitoring, analyzing, and tuning. First, it maintains a load power consumption history record for the last *N* evaluation frames. It also tracks the average IPC and the power states of each processor core in a profiling table. All these data are updated dynamically in the *Chameleon* memory, ensuring timely diagnosis of the system. Second, the micro-controller analyzes its past power management effectiveness based on the stored monitoring data. At the end of each coarse-grained mode-tuning interval, it updates the mode register with a newly selected mode code. Finally, at each fine-grained cycle, the power control firmware executes load tuning instruction based on the specified power management mode and the supply-load power mismatch at that given timestamp.

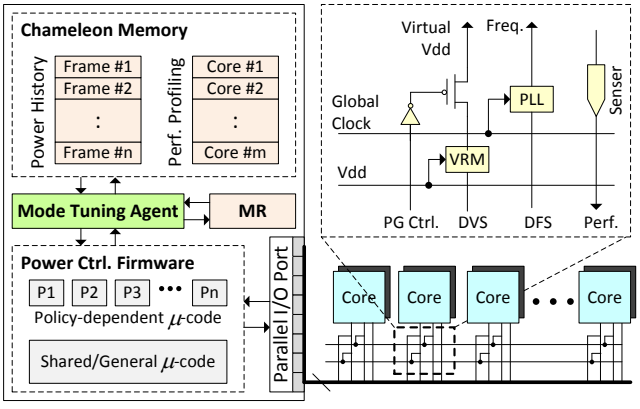

Figure 2: The control architecture of mode-switching power management mechanism in Chameleon μ-controller

In this study we mainly consider dynamic voltage and frequency scaling (DVFS) and processor power gating (PG) as major load tuning knobs. Each processor core is connected to an on-chip voltage regulator module (VRM) through a power-gating transistor. Therefore, the power control firmware can put the throughput processor into low power states by lowering the core V/F level or temporarily enable sleep state.

Note that the firmware based power control implementation is fairly scalable and easy to update. Although the entire power management loop passes through several levels, the control latency is a negligible factor compared to the much longer time interval of power budget variation.

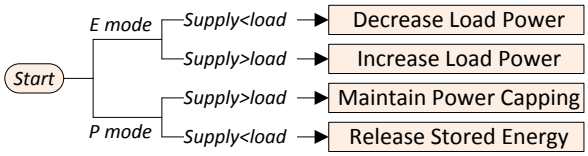

Figure 3: The mapping of power management modes to load adaptation behaviors

C. Learning-Based Mode Switching

The mode tuning agent is *Chameleon*'s key optimization engine. It determines the state of the mode register, and thereby affects how servers will react to the time-varying green power budget. In Figure 3 we show the load adaptation policy in different power management modes. When the server is set as *energy-oriented* mode (*E-mode*), it will strive to maximize direct green energy usage. That is, the micro-controller will dynamically tune the processor voltage and frequency levels to follow the power supply budget. On the other hand, in *performance-oriented* mode (*P-mode*), the server prefers to use stored energy to fill the power shortfall when the load power demand is higher than the power supply budget.

Power mode switching is challenging. An intuitive thinking is that one can simply adjust the mode based on pre-defined power threshold or workload performance threshold: i.e., use *energy-oriented* mode when power headroom increases and use *performance-oriented* mode when server throughput drops. However, such straightforward scheduling scheme lacks robustness and often shows unsatisfied results when faced with disturbing events. For instance, an increase in power budget is actually not a good indication of abundant renewable power supply – it is very likely that the renewable power output fluctuates heavily and it is just temporary power surge. Similarly, a temporary drop in server throughput does not necessarily mean that we are lack of power budget – probably the throughput processor is waiting due to memory access.

In this study we explore a reinforcement learning based mode tuning approach. Rather than specify a rigid mode switching policy, we provide *Chameleon* the privilege to make its own decisions through controller-system interaction. Such approach has two advantages. First, learning-based design is more adaptive and is able to optimize its power management effectiveness over the lifetime. Second, it shows better extensibility when we need more power management modes to handle the increasing complexity in hardware and workload. For threshold-based mode switching scheme, it is difficult to analyze the entire design space when facing a large number of modes and threshold values.

The behavior of the mode tuning agent (i.e., the learner) is given by a sequence of state-mode pairs (S_i, M_i), where S_i is the supply-demand mismatch (in Watts) in the last control period and M_i is the power management mode that the agent decides to select based on its experience and observation of the state. Meanwhile, the agent receives numerical rewards R_i from the system as a feedback for the mode switching action it takes. The reward signal indicates if the mode selection is favorable in an immediate sense and helps the mode tuning agent to update its knowledge base in the future.

The reward function specifies the objective of the mode tuning agent. The goal is to maximize the cumulative reward over the long time. We define the reward as:

$$R = \frac{Performance}{Power\ Variability} = \frac{\sum IPS}{A(p)F(p)}, \quad (1)$$

where IPS is the instruction per second for each individual processor core, $A_{(\,)}$ and $F_{(\,)}$ are the fluctuation amplitude and the fluctuation frequency of the power data series, respectively. $A_{(\,)}$ is calculated as the standard deviation of the load power. The definition of $F_{(\,)}$ is similar to the averaged zero-crossing rate widely used in the signal processing community:

$$F(p) = 0.5 \times \sum |\,\mathrm{sgn}[x(n) - \bar{x}] - \mathrm{sgn}[x(n-1) - \bar{x}]\,|, \quad (2)$$

Where $sgn_{[\,]}$ is the sign function and $\bar{x}$ is the mean value of x. Note that in Eq-1, the power variability is a lower-is-better metric. The definition of our reward function implies that we want the mode tuning agent to optimize the system throughput while be aware that severe load power variation is harmful. Note that frequent load power cycles can result in thermal cycles and potentially degrade the lifetime of the system.

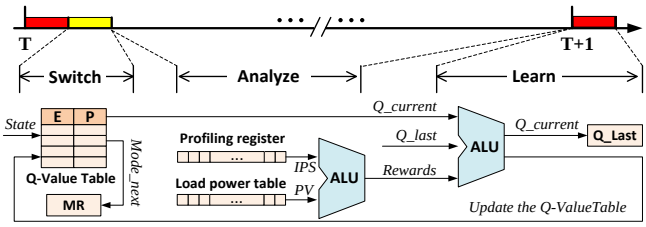

Figure 4: Online-learning control flow and timeline

D. Solving the Mode Switching Problem

We solve the mode switching problem using Q-learning algorithm [6]. It is easy to implement as hardware controller since the algorithm naturally operates in an on-line, fully incremental fashion. The basic idea of Q-learning is to store a so-called Q-value for each state-mode pair (S_i, M_i) in a lookup table. The Q-value is used to predict the possible return of each power management mode at any given system state. We evenly divided the range of supply-demand mismatch into four intervals, with each mapping to a given Q-table entry. We choose a four-entry table because more entries will increase the computation complexity significantly [6].

Figure 4 illustrates the control follow in a given mode tuning period. In the '*Switch'* stage, the mode tuning agent determines current state S_i based on monitored data and selects a power management mode. In the '*Analyze'* stage, the agent waits for a sequence of load tuning activities and collects profiling data (the feedback information) and calculates the reward value. In the '*Learn'* stage, the *Chameleon* controller updates Q-table based on a comprehensive evaluation of the previous mode decisions and reward value.

Figure 5 shows our learning-based mode tuning scheme. The agent chooses a mode that has larger Q-value at any given system state. Meanwhile, it occasionally takes a random power management mode instead of the optimal one observed from the Q-learning table. This is referred to as exploration of the environment and helps to avoid local optimal [6]. In Figure 5, the discount factor *r* determines the importance of future rewards; the learning rate determines to what extent the newly acquired information will override the old information.

```
Initialize: All state-action pairs are initialized to zero
Requires: Discount factor γ , learning rate α , exploration probability ε
1   Set mode register with mode
2   IF (rand () < ε )
3       mode ←Select random mode
4   ELSE
5       Evaluate current system performance states
6       mode ←select mode which maximizes Q-value
7   END IF
8   IF (new mode tuning cycle T begins)
9       Calculate reward R
10      Q' ← next Q-value read from the learning table
11      Q[s, a] ← (1 - α ) ×Q[s, a] + α ×(R + γ × Q'[s', a'])
12  END IF
```

Figure 5: Pseudo code for mode switching

III. Evaluation Methodology

In this section, we introduce our heavily instrumented framework that incorporates a cycle-based microarchitecture simulator and real-world traces of green energy sources.

Our evaluation framework of a throughput server processor is modeled based on the cycle-level simulator GPGPU-Sim v2.1.1b [7], which simulates throughput core, interconnection network, texture/constant cache, memory controller, L1/L2 cache, and off-chip DRAM. We have enhanced the simulator to model core-level power gating and DVFS. Table 1 summarizes our configuration that closely models NVIDIA GTX 580 [8] and Table 2 summarizes power control configurations.

We designed a throughput server power simulator based on McPAT [9] and incorporated power model for various throughput processor components such as fetch, decode, schedule, execution units, register file, shared memory etc. We also models non-core components including interconnect, L2 cache and memory controllers. These components account for a large portion of power consumption in a throughput processor. We also implemented runtime power management module that supports dynamic clock adjustment at the beginning of every DVFS period. To validate our model, we compared our simulation results with actual power measurement using current sensors attached to a GTX 470 GPU module and found satisfactorily close match.

We choose from a large set of available throughput workloads from Nvidia CUDA SDK [10], Rodinia Benchmark [11], Parboil Benchmark [12], and some third party applications. The selected workloads show good mix of memory accesses and compute instructions. The problem size of the workloads is scaled to avoid long simulation time or unrealistically small workload stress. Table 3 lists the evaluated workloads.

We use renewable power supply traces generated from the raw data of HOMER [13]. HOMER is developed by the National Renewable Energy Lab for simulating renewable energy technologies using real-world power data. As shown in Table 4, we generate four traces with different combinations of power supply variability and average power budget levels.

We also evaluate the impact of dynamic load power adaptation on processor temperature using HotSpot 5.0 [14]. We instrumented HotSpot5.0 to model the thermal characteristics of the throughput processor and also integrated it within our simulation framework. Table 5 summarizes the HotSpot parameters used in our experiment. In this study we monitored the highest temperature of the processor and counted the thermal cycles throughout the simulation. We adopt the reliability estimation method studied in [4].

TABLE 1. SIMULATION CONFIGURATIONS

Parameters	Configuration
Throughput core	16 core, 32 SIMD pipeline
Cache (L1/L2/Const/Tex)	32/512/16/16 (KB)
Memory system	48KB shared, 6 controller, 303Gb/s
Register count	16384 per shader core
Threads per SM	1024
Topology and bandwidth	Mesh topology, 16B channel BW
MC scheduling policy	FRFCFS

TABLE 2. LOAD POWER TUNING PARAMETERS

Parameters	Configuration
Core voltages (V)	0.95, 1.05, 1.15, 1.25, 1.35, 1.45
Core freq. (GHz)	1.2, 1.4, 1.6, 1.8, 2.0, 2.2
Interconnection (V/f)	1.05V/1.4G, 1.25/1.8G, 1.45/2.2G
Memory Ctrl. and L2 (V/f)	0.95V/1.0G

TABLE 3. SYNOPSIS OF EVALUATED WORKLOADS

Abbr.	Workload	Avg. Power	Avg. IPC
FWT	Fast Walsh Transform	128.1 W	107.8
HY	Hybrid Sort	80.6 W	86.1
LIB	LIBOR	112.3 W	96.5
MM	Matrix Multiplication	176.7 W	300.1
MT	Matrix Transpose	104.4 W	226.5
MUM	MUMmer GPU	202.8 W	11.9
NW	Needleman Wunsch	80.6 W	21.6
PF	Path Finder	122.4 W	113.3
PNS	Petri Net Simulation	143.7 W	67.4
ST3D	Stencil 3D	125.5 W	122.6

TABLE 4. RENEWABLE POWER SUPPLY TRACES

Trace Description	Avg. Power	σ (std)
High Power Budget + High Variability	135W	38W
High Power Budget + Low Variability	178W	17W
Tight Power Budget + High Variability	40W	41W
Tight Power Budget + Low Variability	101W	14W

TABLE 5. HOTSPOT PARAMETERS

Parameters	Value
Chip thickness / area	0.15 mm / 530 mm^2
Convection resistance	13.9 K/W
Heat sink width / thickness	0.07 m / 69 mm
Interface material thickness	0.025 mm

IV. Experimental Results

In this section we assess the impact of mode-switching power management on throughput servers in terms of energy, performance, and system lifetime. Our main baseline is *Threshold*, which combines different power management modes using simple threshold-based switching method (i.e., the average core power/performance level). We also compared *Chameleon* with the recent *energy-oriented* approach. We do not focus on *performance-oriented* design since its performance heavily depends on the actual installed battery capacity.

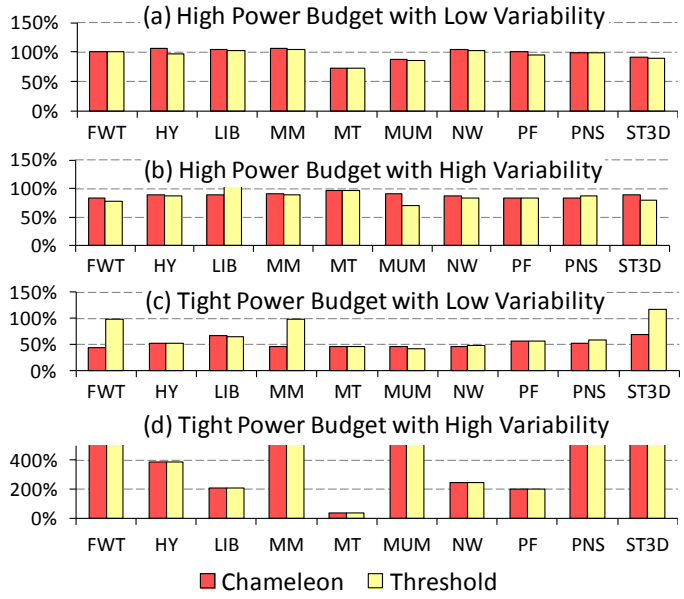

Figure 6: Renewable energy utilization for different workloads (Normalized to *energy-oriented* design)

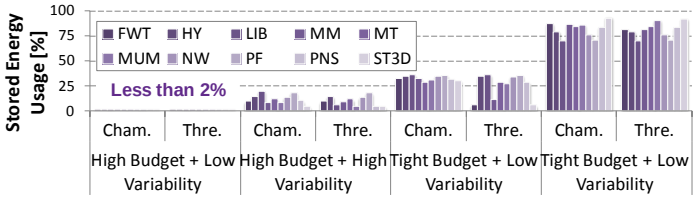

Figure 7: Dependence on energy storage (i.e., the cumulative amount of green energy that goes through battery divided by the overall green energy consumed by the system)

A. *Energy Utilization*

We first analyze the energy utilization profile of different power management schemes. As shown in Figure 6, when the green power budget is high, the efficiency of *Chameleon* and *Threshold* are both close to existing *energy-oriented* approach. On average, *Chameleon* could achieve 97% utilization rate compared to an aggressively supply-tracking based design and shows 95% of renewable energy utilization.

Second, if the green power budget is tight, the efficiency of different schemes can vary significantly when faced with different power budget variability. If the budget variation is low, mode-switching (i.e., *Chameleon* and *Threshold*) shows nearly 50% less renewable energy utilization since it tradeoffs energy efficiency for performance. However, when the power variability is high, mode-switching based design outperforms *energy-oriented* design significantly. This is because the later approach loses its advantages of directly utilizing the green power budget when the power budget frequently drops below a very low level.

In Figure 7 we show the energy utilization profile from the point of view of the stored energy. The system increasingly gives P-mode (i.e., using stored energy to boost system performance) high priority as the green power budget becomes tight and variable. Our results show that about 75% of the green power must be first stored into a battery then it can be used for the computing system to meet their performance/reliability goal.

B. *Performance Acceleration*

We further evaluate the benefits of mode switching in terms of average system throughput, as shown in Figure 8. Compared to *energy-oriented* design, *Chameleon* mode-switching power management could improve the performance by 1.5X when the power generation is high. When the green power budget is tight, our technique shows 16X increased performance compared to a rigid *energy-oriented* design. More importantly, *Chameleon* design outperforms simple threshold-based mode switching scheme due to its online learning capability. Although the actual effectiveness of our learning based mode-switching varies with different workloads, *Chameleon* shows 13% higher workload speed on average across all the workloads.

C. *Reliability Benefits*

The effect of aggressive load power adaption on processor is an open question. Prior study has shown concern for the lifetime issue of using power cycling on servers but does not dig into it. We argue that aggressive power tracking and load power tuning can result in excessive number of thermal cycles (TC), which will result in permanent failure of semiconductors [4].

Chameleon alleviates this problem as the mode tuning agent is aware of the load power variability and tries to avoid it. Figure 9 shows the mean time between failures (MTBF) of throughput processors in *Chameleon* and *Threshold* design. The results are normalized to that of the *energy-oriented* approach. As we can see, the second and third plots show less MTBF improvement compared to the first and forth plots. This is because *energy-oriented* design causes fewer thermal cycles when the green power supply is relatively stable and abundant. Overall, mode switching power management could improve the processor lifetime significantly, and *Chameleon* even shows 42% longer MTBF than *Threshold* due to its variability-aware light-weight mode switching activities.

V. Related Work

Sustainability has become an increasing concern in recent studies. In this section we discuss the state-of-the-art design in the emerging green energy-aware computing research.

A. *Focusing on server load matching*

There have been several pioneering studies on matching computer power demand with renewable power budget [15 - 22]. For example, Li et al. proposed load adaptation scheme for solar energy powered multi-core servers [16] and Sharma et al. discussed the impact of tracking intermittent power on system performance [17]. They both chose hardware tuning knobs for load matching. On the other hand, Goiri et al. leveraged workload scheduling based on the availability of renewable energy [18, 19]. Also, Zhang et al. explored workload routing algorithm for maximizing green energy usage [20] and Li et al. proposed energy-aware workload migration for improving load matching efficiency [21]. At HP, Arlitt et al. designed clusters that adjust its load based on the green power availability [22].

B. *Focusing on power source managment*

Another design approach is to fully leverage ancillary power systems or utility grid. For example, Govindan et al. evaluated the benefits of using onsite energy storage for managing power demand surge and power shortage problems [23, 24]. Deng et al. investigated distributed grid-tied inverter for integrating green energy at different design levels [25].

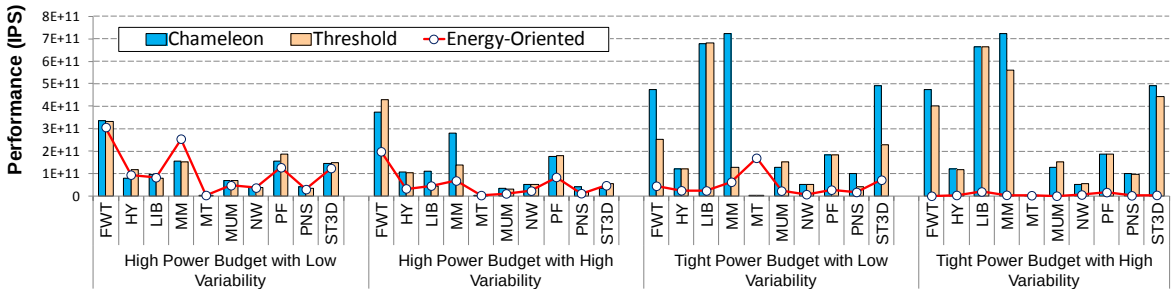

Figure 8: Comparison of average throughput across different power management schemes

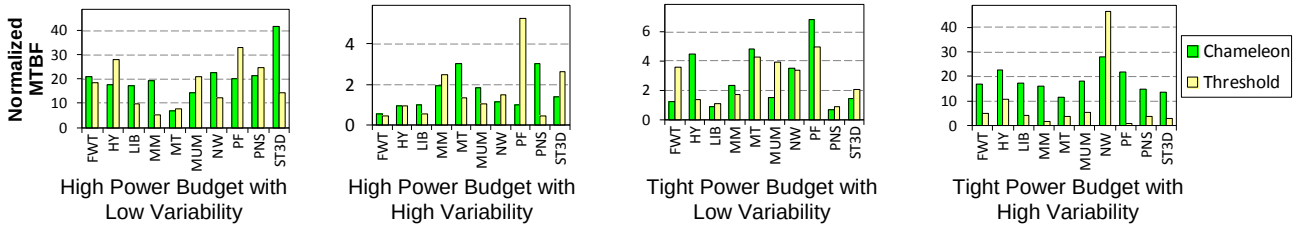

Figure 9: Comparison of processor lifetime across different power management schemes

C. Joint management of supply and load

We notice that recent designs have highlighted cooperative tuning of renewable energy sources and computing workload to achieve a better performance. For example, a joint control of computing power demand and onsite generator output is discussed in [26] for optimizing the green energy utilization and workload performance simultaneously. In [27], the system not only takes advantage of the self-tuning capabilities of computing load, but also intentionally switches the load between green energy and utility grid for better performance.

Different from existing works, this paper explores intelligent integration of different power management schemes for the optimal tradeoff between performance and efficiency. To our knowledge, this is the first paper that applies learning technique to throughput server power management to achieve reliability-aware adaptive green computing.

VI. CONCLUSIONS

Motivated by the energy crisis and environmental issue, this paper investigates green energy powered throughput server system. Existing *energy-oriented* and *performance-oriented* approaches lack a holistic view of managing the variable green power budget. We show that intelligent switching between different power management modes can boost the system performance by 12.8%, enhance reliability by 42%, and still maintain up to 95% green energy utilization.

VII. ACKNOWLEDGEMENT

This work is supported in part by NSF grants 1117261, 0845721(CAREER), by Microsoft Research Safe and Scalable Multi-core Computing Award, by NSFC grant 61128004, and by 863 Program of China grant 2012AA010902-3. Chao Li is also supported by a University of Florida Graduate Fellowship and by a Facebook Fellowship.

REFERENCES

[1] K. Brill, "The Economic Meltdown of Moore's Law and the Green Data Center," The Uptime Institute 2007
[2] http://www.apcmedia.com/salestools/WTOL-7DJLN9_R1_EN.swf
[3] IEEE Guide for Array and Battery Sizing in Stand-Alone Photovoltaic (PV) Systems, IEEE Std 1562™-2007
[4] A. Coskun, R. Strong, D. Tullsen and T. Rosing, "Evaluating the Impact of Job Scheduling and Power Management on Processor Lifetime for Chip Multiprocessors," *SIGMETRICS*, 2009
[5] V. Kontorinis, L. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. Pettis, T. Rosing and D. Tullsen, "Managing Distributed UPS Energy for Effective Power Capping in Data Centers," *ISCA*, 2012
[6] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," MIT Press, 1998
[7] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," *ISPASS*, 2009
[8] http://www.nvidia.com/object/product-geforce-gtx-580-us.html
[9] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," *MICRO*, 2009
[10] NVIDIA CUDA SDK, http://developer.nvidia.com/cuda-downloads
[11] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.Lee, and K. Skadron, "Rodinia: Accelerating Compute-Intensive Applications with Accelerators," *IISWC*, 2009
[12] Parboil Benchmark Suite, http://impact.crhc.illinois.edu/parboil.php
[13] Getting Started Guide for HOMER Version 2.1, www.homerenergy.com
[14] HotSpot 5.0 Temperature Modeling Tool, http://lava.cs.virginia.edu/HotSpot/
[15] C. Li, A. Qouneh, and T. Li, "Characterizing and Analyzing Renewable Energy Driven Data Centers", *SIGMETRICS*, 2011
[16] C. Li, W. Zhang, C. Cho, and T. Li, "SolarCore: Solar Energy Driven Multi-core Architecture Power Management," *HPCA*, 2011
[17] N. Sharma, S. Barker, D. Irwin, and P. Shenoy, "Blink: Managing Server Clusters on Intermittent Power," *ASPLOS*, 2011
[18] I. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenHadoop: Leveraging Green Energy in Data-Processing Frameworks," *EuroSys*, 2012
[19] I. Goiri, K. Le, Md. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: Scheduling Energy Consumption in Green Datacenters," *SC*, 2011
[20] Y. Zhang, Y. Wang, and X. Wang, "GreenWare: Greening Cloud-Scale Data Centers to Maximize the Use of Renewable Energy," *Middleware*, 2011
[21] C. Li, A. Qouneh, and T. Li, "iSwitch: Coordinating and Optimizing Renewable Energy Powered Server Clusters," *ISCA*, 2012
[22] M. Arlitt et.al., "Towards the Design and Operation of Net-zero Energy Data Centers," *ITherm*, 2011
[23] S Govindan, A. Sivasubramaniam, and B. Urgaonkar, "Benefits and Limitations of Tapping into Stored Energy for Datacenters," *ISCA*, 2011
[24] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar, "Leveraging Stored Energy for Handling Power Emergencies in Aggressively Provisioned Datacenters," *ASPLOS*, 2012
[25] N. Deng, and C. Stewart, "Concentrating Renewable Energy in Grid-tied Datacenters," *ISSST*, 2011
[26] C. Li, R. Zhou, and T. Li, "Enabling Distributed Generation Powered Sustainable High-Performance Data Center," *HPCA*, 2013
[27] I. Goiri, W. Katsak, K. Le, T.D. Nguyen, and R. Bianchini, "Parasol and GreenSwitch: Managing Datacenters Powered by Renewable Energy," *ASPLOS*, 2013