Energy Optimization by Fan Speed Control for Data Centers

by Min Tian

B.S. in Communication Engineering, June 2016, Beijing University of Posts and Telecommunications

A Thesis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Master of Science

August 31, 2019

Thesis directed by

Suresh Subramaniam
Professor of Electrical and Computer Engineering

Acknowledgments

I would like to thank my advisor Professor Suresh Subramaniam in the first place. He is one of the best professors I have ever met. As my advisor, he always uses his patience and guidance to encourage and inspire me on this work. His great knowledge not only guides me on this work but also gives me courage to challenge and explore more. Additionally, I would like to thank Professor Guru V. Venkataramani and Dr. Arun Vishwanath. During this research I have met lots of challenges that I'm not familiar with, it's Professor Guru V. Venkataramani and Dr. Arun Vishwanath's knowledge help me to learn and solve these problems.

Abstract

Energy Optimization by Fan Speed Control for Data Centers

Currently, energy efficiency for data centers is an attractive problem. For data centers, good scheduling algorithms must not only satisfy the Quality of Service (QoS) and but also try to minimize the total energy. However, such algorithms do not always consider the effect of the cooling system, which consumes a significant amount of energy by itself. An algorithm that can balance the QoS and energy cost of the CPU and network components might cause an unexpected energy increase in the cooling system. According to [4], the energy of cooling system in data centers can be as large as 50 percent of the overall consumption. This thesis investigates algorithms that can balance the overall energy consumption with QoS by considering a cluster of compute servers, each of which is cooled by fans. The impact of CPU power on the cooling system can be predicted by a thermal model, and by relating to the power model of fan, we get the relation between CPU power and the fan power. Workload Adaptive Energy-Latency Optimization in Server Farms using Server Low-Power States (WASP) is a previously proposed algorithm that can adjust itself dynamically according to workload. This thesis exmplores the impact of WASP on the cooling system under different sizes of jobs and utilization levels, and proposes a pre-cooling based optimization for the fan speed to improve WASP.

Table of Contents

Acknowledgments . . . . . **ii**

Abstract . . . . . **iii**

List of Figures . . . . . **vi**

List of Tables . . . . . **vii**

1 Introduction **1**

1.1 Low-Power State Based Optimal Framework 1
1.2 WASP and A-I 2
1.3 Optimal Fan Speed for Servers 3
1.4 Contribution in this Thesis 4

2 Energy Saving Efficiency of WASP **6**

2.1 Power Data 6
2.2 Configuration for WASP 7
2.3 Energy Consumption Comparison between WASP and A-I 9
2.4 Conclusion 12

3 Thermal Model for Server and Optimal Fan Speed **13**

3.1 Stable Thermal Model 13
3.2 Transient Thermal Model 13
3.3 Realization of the Thermal Model 14

4 Fan Power Model and Optimal Fan Speed **16**

4.1 Fan Power Model 16
4.2 Optimal Fan Speed 16
4.2.1 Optimal Fan Speed for Small Latency Jobs 19
4.2.2 Optimal Fan Speed for Large Latency Jobs 20
4.2.3 Comparison of the CPU Power Trace for Different Job Sizes 21
4.3 Pre-cooling 21
4.4 The Energy Demand for WASP and A-I under Pre-cooling Based Optimal Fan Speed 29

5 Limiting Active Cores for Energy Optimization **32**

5.1 The Energy Saving Efficiency of Core Limitation 32
5.1.1 The Constraint on Core Limitation 32
5.1.2 The Energy Saving under Core Limitation 33
5.1.3 Best Core Limitation under Different Environments 35
5.1.4 The Essence of Core Limitation 36
5.1.5 Indirect Energy Saving on CRAC 38

6 Discussions **39**

7 Future Work **42**

Bibliography **44**

List of Figures

2.1 Energy of Each Server for different N_a (Joules), N = 10, u = 0.3 8
2.2 Total energy for different N_a 9
2.3 Execution time for different N_a 10
2.4 95th percentile normalized latency for different N_a 10
2.5 Power for WASP and A-I under different utilization 11

3.1 Comparison between Measurement and Thermal Model 15

4.1 Comparison between Fan Power Model and Measurement 17
4.2 CPU Power Trace of a Server 18
4.3 Comparison of Optimal and Stable Fan Speed 18
4.4 CPU Temperature Comparison of Optimal and Stable Fan Speed 19
4.5 Total Energy of Optimal and Stable Fan Speed 19
4.6 Total Optimal Fan Power for WASP and A-I (Small Latency Jobs) 20
4.7 Total Optimal Fan Power for WASP and A-I (Large Latency Jobs) 21
4.8 The CPU Power Trace of WASP under Different Size of Jobs and Low Utilization 22
4.9 A CPU Power Trace of WASP (Small Latency Jobs) 22
4.10 Demonstration for Cycle 23
4.11 Optimal Fan Speed during Active Period in a Cycle 23
4.12 Pre-cooling Based Optimization Fan Speed under Different T_{max} and $T_{initial}$. 25
4.13 The Relationship between $T_{initial}$ and $FS_{initial}$ 26
4.14 The Optimization Fan Speed with and without Pre-cooling Scheme 28
4.15 The CPU Temperature under Different Optimization Fan Speed 29
4.16 The Fan Power under Different Optimization Fan Speed 30
4.17 Total Fan Energy for different $T_{initial}$ 30
4.18 Fan Power Comparison for WASP and A-I under Different Optimization Fan Speed 31

5.1 Energy without Fan under Different Core Limitation of WASP and A-I . . . 34
5.2 Fan Energy under Different Core Limitation of WASP and A-I 35
5.3 Total Energy under Different Core Limitation of WASP and A-I 36
5.4 CPU Power Trace of 26-Core Limitation for Small Latency Jobs 37

List of Tables

1.1 State for server and CPU 2
1.2 Wake-up Latency for Different C-state and S-state [20] 2
1.3 Parameters for WASP power management algorithm [20] 3
1.4 Energy Consumption Breakdown of Data Center [4] 3

2.1 Power Parameters for Server with one 10-core CPU under different state . . . 6
2.2 Power Parameters for Server with one 28-core CPU under different state . . . 7
2.3 95th normalized latency for WASP 11

4.1 Increase Point for different T_{max} (Celsius) 25

5.1 Execution Time under Different Core Limitation of WASP and A-I 33
5.2 Best Core Limitation under Different Utilization ($T_{amb} = 25°C, T_{max} = 80°C$) 35
5.3 Best Core Limitation for Different Temperatures and Job Sizes 37

Chapter 1: Introduction

Large data centers nowadays is a significant consumer of energy; they constitute about 2% of the US energy consumption, and the proportion keeps growing year by year [1]. As the number and size of data centers increase, optimization of the overall system power and achieving a higher level of performance have been important concerns for data center operators [21]. Optimizing the energy inside the data center is a challenging problem, and the exploration for autonomous frameworks with system-wide energy management is essential [22].

1.1 Low-Power State Based Optimal Framework

Though several server-level frameworks have been suggested before, such as the low-power state-based frameworks studied in [9][11], they fall short of successfully handling dynamically arriving jobs while meeting the Quality of Service (QoS) demands. Frameworks such as leveraging temporal-spatial batching (TS-Bat) can judiciously integrate spatial and temporal batching of jobs to save energy while meeting the constraint of QoS [24][23].

To meet the QoS, the latency for jobs must be maintained under a certain level. Latency can be caused by queuing; for example, this will happen while there are not enough servers to handle a burst in the arriving jobs, and the upcoming tasks have to keep queuing. Latency can also be caused by wake-up time. As shown in Table 1.1, for system and CPU, there are different sleep states that can save energy by shutting down different components [9]. C_x sate is the low-power state for processor, while S_x is the low-power state for the system. While all the cores of a processor enter C6, this processor itself will enter into another state called package sleep state. However, there will be a wake-up latency for both system and CPU, and the duration of wake-up latency grows for a deeper sleep state, as shown in Table 1.2. Servers can save energy by entering different sleep states, but the wake-up latency will

Table 1.1: State for server and CPU

SYMBOL	DESCRIPTION
$C0_{(a)}S0$	There are tasks in process, both the system and CPU are active
$C0_{(i)}S0$	There are no tasks in process, both the system and CPU are active
$C1S0$	The CPU enters halt state (shallow package sleep state), while the rest of system remains active
$C6S0$	The CPU enters deep sleep state, while the rest of system remains active
$C6S3$	The system (includes CPU) enters deep sleep sate

Table 1.2: Wake-up Latency for Different C-state and S-state [20]

STATE	WAKE-UP LATENCY
Core Sleep C1	10 μs
Core Sleep C6	82 μs
Package Sleep C6	1 ms
System Sleep	5 s

increase the job latency, which may in turn cost some energy.

1.2 WASP and A-I

Compared to these low-power based frameworks, there is another framework called workload-adaptive server provisioning (WASP) [20]. One of the advantages of WASP is that by adjusting its unique parameters according to the feature of current workload, it can minimize the energy cost while meeting the QoS. The parameters of WASP, as shown in Table 1.3, define when a server or CPU will enter different sleep states and when it wakes up. The values of T_s, T_w, τ determine whether or when to enter/exit different sleep states, and they vary with different systems.

Compared to WASP, Active-Idle (A-I) only switches the server between active and shallow package sleep state (bewteen $C0_{(a)}S0/C0_{(i)}S0$ and $C1S0$). While such a simple scheme will lead to improved QoS because of its small wake-up latency, a significant amount of energy could be wasted when there are no jobs to process.

Table 1.3: Parameters for WASP power management algorithm [20]

SYMBOL	DESCRIPTION
V_{act}	Servers in active/package sleep mode
V_s	Servers in system sleep mode
T_s	Workload threshold to reduce active servers
T_w	Workload threshold to increase active servers
N_p	Servers in system sleep mode
τ	Delay time before entering system sleep

Table 1.4: Energy Consumption Breakdown of Data Center [4]

COMPONENTS	PROPORTION
Cooling	50%
Server & Storage	26%
Power Conversion	11%
Network Hardware	10%
Lightning	3%

1.3 Optimal Fan Speed for Servers

However, the above frameworks only consider the energy of the server itself, and ignore the impact of the cooling system inside the data center. The energy consumption of the cooling system can even reach 50% of the total energy of a modern data center [4], as shown in figure 1.4. Within the cooling system, the energy of the cooling fans in servers is almost the same as the energy of computer room air conditioning (CRAC) unit.

In this thesis, we investigate energy optimization by controlling the fans. Several fan speed optimizations have been suggested before [5][7]; these optimizations are based on the server temperature, such as using PID control (proportional–integral–derivative control) to decide fan speed according to current temperature. In [17], a thermal resistance-based thermal model is applied to optimize fan speed; this model can give precise prediction of temperature based on current temperature, ambient temperature, and CPU power.

1.4 Contribution in this Thesis

In this thesis, we discuss the influence of WASP on cooling system, and propose improvements for that. The thesis makes the following contributions:

1) We solve the relation between CPU power and the corresponding required fan power to find the influence of WASP/A-I on cooling system. CPU generates most of the heat inside a server. Most of the CPU power transforms to heat directly, so it can be seen as the major heat source inside the server. In [17], the authors suggested a thermal resistance-based thermal model. Thermal resistance is decided by the fan speed, and this relationship can be different for different servers, since the space of the server can be different. The cooling efficiency of a specific fan speed can be reflected by corresponding thermal resistance. Under a stable state (the ambient temperature, CPU temperature), there will be a required thermal resistance for a given CPU power. So for given CPU power and temperature requirement, the corresponding fan speed can be solved through thermal resistance. The relationship between fan power and the fan speed (in units of revolutions per minute (RPM)) can be approximately described as a cubic function. Combined with the thermal model, we can build a model between CPU power and the Fan Power.

Basing on the power model between CPU and Fan, the minimum required fan power under certain cases can be solved. In other words, according to current CPU temperature and current CPU power, the temperature in the next timeslot can be predicted. Then the purpose is to find the minimum fan power, under the constraint that the prediction of next slot does not exceed the prescribed CPU maximum temperature.

2) We present two improvements for the optimal fan speed and WASP. The first is a pre-cooling scheme for optimal fan speed. The feature of the fan power model makes the choice of fan speed more complex. Since it is a cubic function($P_{fan} \propto FS_{fan}^3$) [13], the fan will consume much more energy at higher speeds. According to the CPU power trace generated by WASP, there exists a power pattern, which makes the power predictable in

some cases. Waiting to speed up the fan until it is about to reach the maximum temperature may lead to vastly increased fan power and energy, due to the cubic relationship. Based on the power trace, it may be helpful to start speeding up the fan well before the CPU is predicted to reach the maximum allowed temperature. This may reduce the overall energy consumption.

3) The second improvement is by limiting the number of active cores within a CPU for WASP. We discuss the power, energy of cooling system and the whole system to find the influence of limitation of cores. Then we present the best core limitation under different environments for WASP and show its energy efficiency.

The thesis is organized as follows. In Chapter 2, we first discuss the energy saving efficiency of WASP, then show its contribution or impact on cooling system in Chapter 3. Then, to improve the energy saving of cooling system, we suggest two improvements for the optimization of fan speed in Chapter 4, and for the WASP algorithm itself in Chapter 5. Conclusions and future directions are presented in Chapter 6 and Chapter 7.

Chapter 2: Energy Saving Efficiency of WASP

In this chapter, we first measure the power data for a certain tower server Dell T610 (with one 10-core CPU and two controllable fans), then perform a simulation using the simulator in [20] to see the latency and energy of WASP and A-I under different workloads.

2.1 Power Data

The power of the server is first measured by IPMItool, then verified by a power meter. The purpose is to measure the CPU power and get the power for each core and different C-states. We disable C-state in the kernel of Linux, to make sure the core will not enter deeper low-power state while it is idle. To simulate an active CPU, we use `dd Linux` command to keep the CPU copying a file from `/dev/zero` to `/dev/null`; this process increases CPU utilization without actually doing any I/O operation. Then we use IPMItool to read the power data under different C-state limitation and workload. We use a linear function to represent the connection between the number of active cores and CPU power [20].

The power during wake-up period is assumed as half of the sum of peak power and the power of current low-power state, as shown in (2.1), where x represents the current low-power state, while P_{peak} equals the power when all the cores are working.

$$P_x^{wake-up} = \frac{1}{2}(P_x + P^{peak}) \tag{2.1}$$

Table 2.1: Power Parameters for Server with one 10-core CPU under different state

STATE	POWER(WATTS)
Core Sleep C1	41.5+4.089*(n_a^*-1) +81.5
Core Sleep C6	39.5+4.31*(n_a-1) +81.5
Package Sleep C6	7.0+74.5
System Sleep	13.8

* n_a is the number of cores in active state.

Table 2.2: Power Parameters for Server with one 28-core CPU under different state

STATE	POWER(WATTS)
Core Sleep C1	68.87+4.85*(n_a^*-1) +81.5
Core Sleep C6	61.79+5.11*(n_a-1) +81.5
Package Sleep C6	104.7456
System Sleep	13.8

* n_a is the number of cores in active state.

For a tower server with only one 10-core CPU, even if it is running at 100% utilization, the heat generated by CPU will not stress the cooling system. Furthermore, the huge enclosure of the tower server will also amplify the cooling efficiency of the ambient temperature. However, the servers used in a data center nowadays are mostly rack servers, and sometimes the energy of the cooling system can reach more than 20% of the total energy of a rack server. In order to simulate a similar case as a rack server, here we assume a CPU with 28 cores to increase the stress on the cooling system, Table 2.2 shows the simulated power parameters according to the measured data shown in Table 2.1.

2.2 Configuration for WASP

The values of T_s, T_w and τ depend on the system and the workload. For given server, shown in Table 2.1, since the base power of the system is huge, it seems to be more efficient to put the server into system sleep state here. But, the wake-up energy from system sleep state of this server will also be huge. The wake-up power calculated by (2.1) is almost as much as the power of Package Sleep State, not to mention that a wake-up latency of 5 s will also cause a potential QoS violation.

Under experiments, the best configurations of T_s, T_w and τ reveal a scheme for this server. Compared to keeping all the servers active, or putting them into system sleep state from time to time, the most energy-saving arrangement is to find N_a servers, that will always be awake (never enter system sleep state), while the rest of the servers stay in system sleep state. As shown in (2.2), N represents the total number of servers, u represents the utilization of N

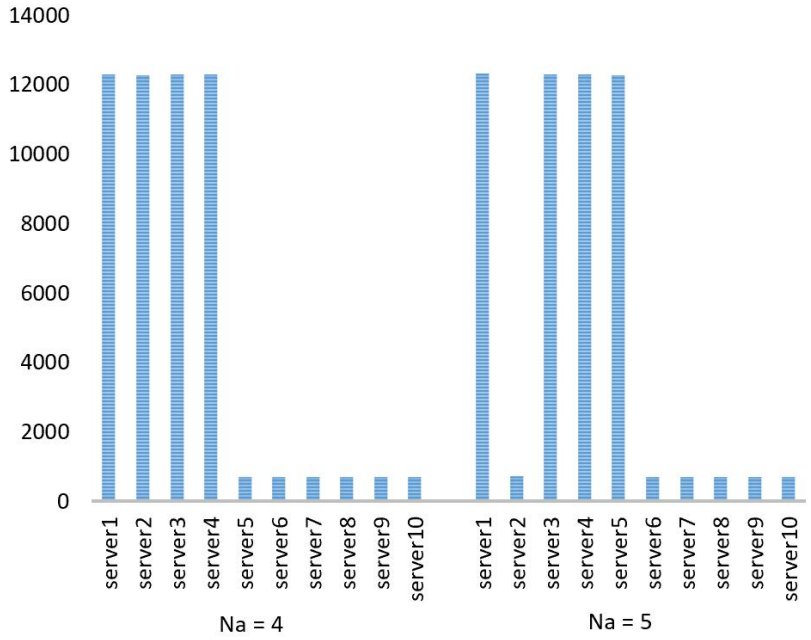

Figure 2.1: Energy of Each Server for different N_a (Joules), N = 10, u = 0.3

servers, u_a is the average utilization of N_a servers that are chosen to keep awake.

$$N_a = \min \left\lceil \frac{N * u}{u_a} \right\rceil, \; u_a < 1 \tag{2.2}$$

According to this formula, if $N = 10, u = 0.3$, then $N_a = 4$ and u_a will be 0.75. Here we run the experiment with 1000000 jobs, the service time per jobs is 4.2 ms. Figure 2.1 shows the energy of each active server under best configurations while N_a is 4 or 5. We can see that the total energy is the same, and even if N_a increases, as shown in the Figure, one of the 5 serves remains in system sleep all the time, and the actual number of awake servers is the same as $N_a = 4$.

In (2.2),the constraint of $u_a < 1$ makes sure that the utilization of each awake server will not reach or exceed 100 percent. Reducing the number of awake servers and keeping them working at full load though can save energy, but may cause a severe latency problem. To explain why the utilization per active server should not exceed 1, we give a comparison

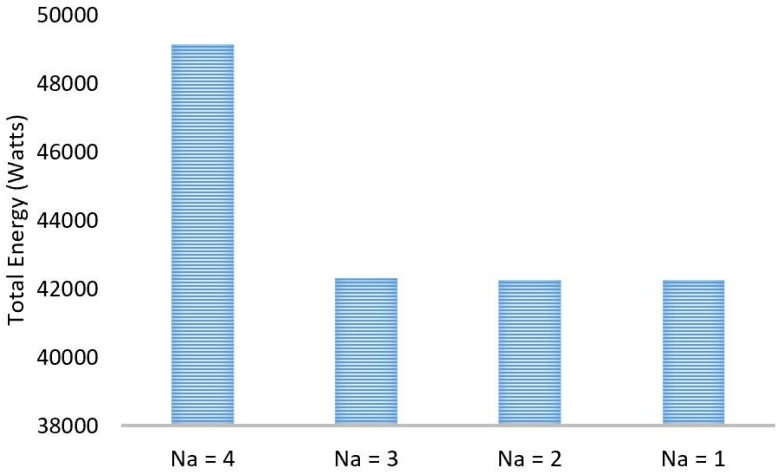

Figure 2.2: Total energy for diferent N_a , N = 10, u = 0.3

for different number of awake servers under 30 percent utilization. The number of servers is 10, which means that according to (2.2), N_a should at least be 4. Figure 2.2 gives the total energy from $N_a = 1$ to 4. As shown in this figure, energies for $N_a = 1$ to 3 are almost the same (because the utilizations are all 100%), and almost half of the energy of $N_a = 4$. Figure 2.3 shows the corresponding job latencies (or execution times) for different N_a. The increase of execution time means that there is also an increase in the queuing tasks. We can see that the execution time for $N_a = 3$ and 4 are almost the same, and for $N_a = 2$ and 1 the execution time grows rapidly.

Figure 2.4 shows the 95th percentile normalized latency. Reducing the number of awake servers can save energy, but the consequence is that tasks have to be queued, resulting in poor QoS.

2.3 Energy Consumption Comparison between WASP and A-I

Compared to WASP, A-I tends to keep all the servers awake all the time, and the processor will stay in C1 state while there is no job available to process. Though there will be much energy wasted during idle period, this algorithm reduces the latency greatly by minimizing the wake-up latency and offers very good QoS. Here we assume jobs will be assigned to

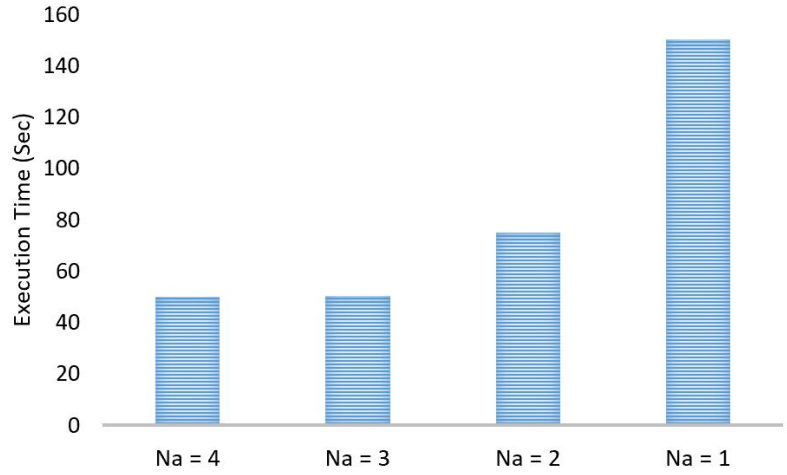

Figure 2.3: Execution time for different N_a , N = 10, u = 0.3

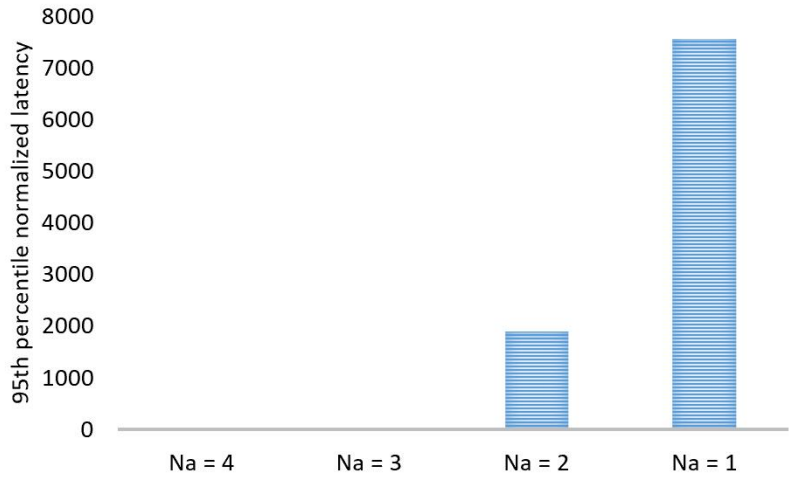

Figure 2.4: 95th percentile normalized latency for different N_a, N = 10, u = 0.3

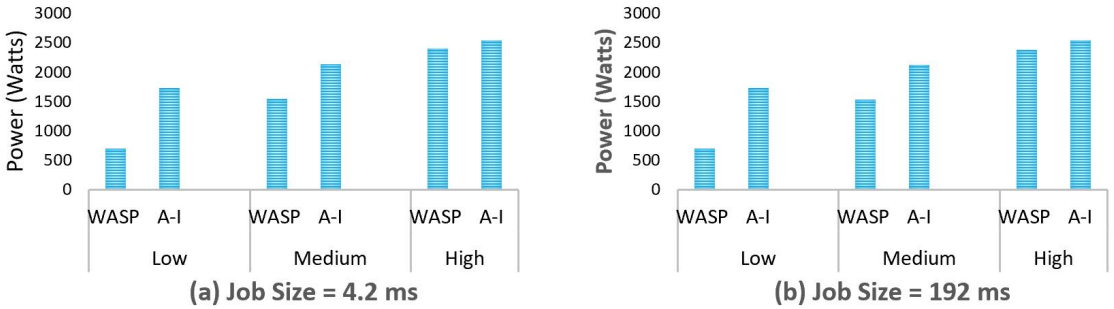

Figure 2.5: Power for WASP and A-I under different utilizations

Table 2.3: 95th normalized latency for WASP

Job Size (ms)	Utilization		
	Low	Medium	High
4.2	1.07879	1.053668	1.059061
193	1.078168	1.049651	1.055844

active CPUs in preference to idle CPUs. We give the power and latency comparison between WASP and A-I under different levels of utilization and different types of jobs. Here we use small latency jobs (Web service-like, service time = 4.2 ms) and large latency jobs (DNS service-like, service time = 194 ms) to run the simulation[10].

As shown in Figure 2.5, the energy saving efficiency of WASP, compared to A-I, is better when the utilization is low. When u = 0.3, the energy of WASP is merely about 40% of the energy of A-I for both small and large size of jobs. In this case, more than half of the servers enter system sleep state according to (2.2), and most of the power saving is caused by the system sleep state. As shown in Table2.1, it can save at least 67.6 Watts per server.

When utilization increases, the number of servers that can stay in system sleep state decreases. When u = 0.9, N_a = 10, WASP cannot save energy by system sleep state; all the

servers keep awake all the time to meet the QoS. However, we can see that the energy of WASP is still smaller than A-I. In this case, the energy saving is mostly caused by the C6 and the package sleep state compared to A-I. The consequence to put servers into sleep state is an increase of the latency; however, as explained in Section2.2, the constraint of N_a will avoid a huge latency.

2.4 Conclusion

This chapter discusses the configuration of WASP, and the comparison of energy between WASP and A-I. Here we use N_a as an important configuration, however this is not suitable for every case as the best configuration will vary for different servers. From Section2.3, we can see WASP can save much energy compared to A-I, but the average power per active server tends to be much more than that for A-I, which might cause the cooling system to be very active.

Chapter 3: Thermal Model for Server and Optimal Fan Speed

We use the thermal model suggested in [17] to find the relationship between CPU power and fan speed. In that paper, the authors use heat transfer theory to build a thermal model for a blade server. Since the server we are using is a tower server with single CPU, and the distances between CPU and fans are almost the same, some parameters in[17] can be ignored in this case (e.g., the parameter that represents the number of blade servers in the enclosure).

3.1 Stable Thermal Model

(3.1) is the thermal model under a stable environment.

$$T_{CPU} = Q\left(\frac{C_3}{(\sum_1^{n_f} FS)^{n_R}} + C_4\right) + T_{amb} \tag{3.1}$$

Here, T_{CPU} is the temperature of the CPU, T_{amb} is the temperature of the ambient environment. Q represents the heat generated by the server (most of it is generated by CPU), n_f is the number of fans, and FS is the fan speed (we assume that the speeds of both controllable fans in a server are the same). C_3, C_4 and n_R are the parameters that need to be identified through experiments.

3.2 Transient Thermal Model

Thermal model under stable state can solve the required fan speed for given CPU power, ambient temperature and CPU temperature. However, such stable model cannot give us a prediction of the CPU power while CPU power is changing. Therefore, we need a transient model (3.2) [2] to obtain the prediction.

$$\Delta T_{CPU}(k+1) = (1 - \frac{\Delta t C_2}{C1 R_j})\Delta T_{CPU}(k) - \frac{\Delta t}{C_1} Q \tag{3.2}$$

$$R_j = \frac{C_3}{(\sum_1^{n_f} FS)^{n_R}} + C_4 \tag{3.3}$$

The $\Delta T_{CPU}(k)$ here means the difference between T_{CPU} and T_{amb} at slot $time = k$ ($\Delta T_{CPU}(k) = T_{CPU}(k) - T_{amb}(k)$). C_1, C_2 can be identified through experiments. The Q represents the heat generated by server here, while R_j means the thermal resistance. Δt is the granularity of time slot.

R_j can be defined by 3.3, here C_3, C_4 and n_R are the same with 3.1 while in the same system. R is relative and only relative to fan speed FS.

3.3 Realization of the Thermal Model

We use the sensors of the server itself to obtain the CPU temperature, the fan speed, and its corresponding fan power. We first use Remote Access Controller Admin (RACADM) utility to set different fan speeds (720 RPM, 2160 RPM and 4120 RPM). Then we measure the corresponding CPU temperature under different CPU powers. (The method to change CPU power was described in Section 2.1.) These measurements are used to generate stable thermal model. CPU power represents Q in 3.1. For given Q, T_{amb} and T_{CPU}, we can obtain the thermal resistance for different fan speeds, and use that to define the value of C_3 and C_4 according to (3.3).

Then we change the CPU workload, and use IPMItool and RACADM to measure the CPU temperature and CPU power during this changing period. We also change the fan speed and measure the temperature trace under same CPU power. The measurement granularity for IMPItool is 1 sec, and for RACADM is 5 sec. This temperature trace under different

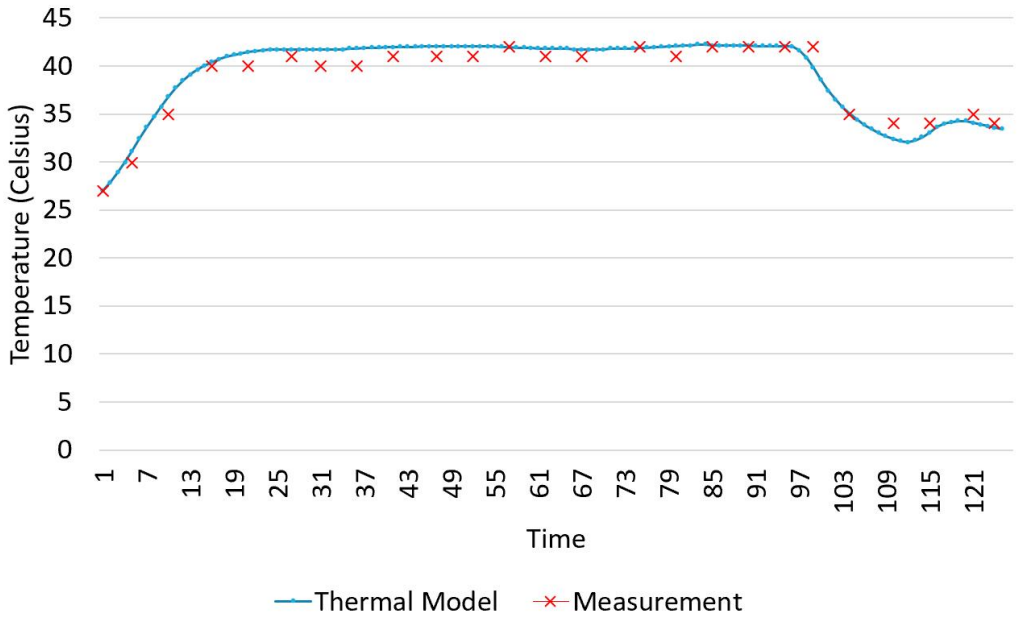

Figure 3.1: Comparison between Measurement and Thermal Model

changes can be used to identify the parameters C_1 and C_2 to build the transient model.

Since the thermal resistance R_j is only relative to the fan speed, by applying (3.3) to (3.2), we can predict the CPU temperature in the next slot ($T_{CPU}(k+1)$) according to current CPU temperature ($T_{CPU}(k)$), CPU power (represented by Q) and the fan speed (represented by R_j). Figure 3.1 shows the comparison between measurement and the prediction of CPU temperature.

With this transient thermal model, we can predict the temperature according to the simulated CPU power trace of WASP/A-I. However, the goal is to obtain the optimal fan speed to save energy. We will present a fan power model for this purpose in the next chapter.

Chapter 4: Fan Power Model and Optimal Fan Speed

In this chapter, we demonstrate the relationship between fan speed and its corresponding fan power. By combining this model with the thermal model we discussed in Chapter 3, we can obtain the optimal fan speed. Furthermore, we propose a pre-cooling scheme for this model based on the unique CPU power pattern of WASP.

4.1 Fan Power Model

As illustrated before, the relationship between fan power and its fan speed (rpm) is approximately a cubic function [6]. Here we use (4.1) to represent the function as below, where P_{FS} is the fan power and FS is the fan speed, and a_i are constants to be determined.

$$P_{FS} = a_0 * FS^3 + a_1 * FS^2 + a_2 * FS + a_3 \tag{4.1}$$

After gathering data, we use a polynomial function to identify the constants in this formula. Figure 4.1 shows the comparison between the model we build and the real measurement.

4.2 Optimal Fan Speed

Using (3.2), (3.3) and (4.1), the temperature of next slot can be predicted for a given CPU power, ambient and current CPU temperature, and fan speed. Our goal here is to find the minimum fan speed that can keep the CPU temperature under a prescribed maximum. The purpose is to minimize the sum of fan power over all time slots (4.2).

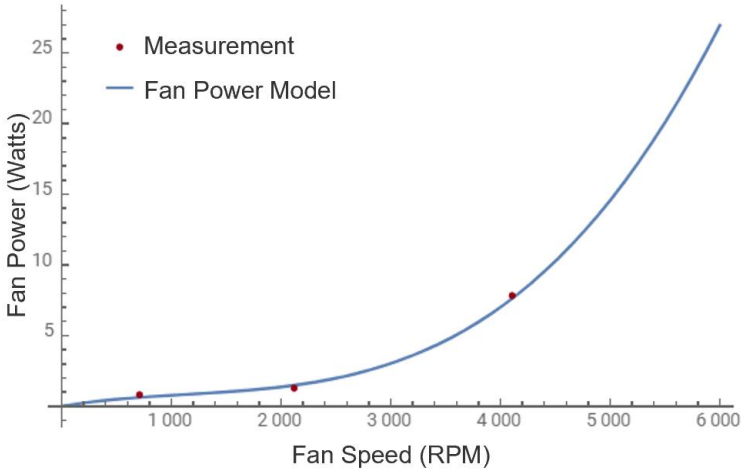

Figure 4.1: Comparison between Fan Power Model and Measurement

$$\min \sum_{n} P_{FS_n} \tag{4.2}$$

$$s.t \quad T_{cpu}(k+1) \leq T_{max} \tag{4.3}$$

$$T_{cpu}(k+1) = (1 - \frac{\Delta t C_2}{C1 R_j})\Delta T_{cpu}(k) - \frac{\Delta t}{C_1} Q \tag{4.4}$$

$$FS_{min} \leq FS \leq FS_{max} \tag{4.5}$$

Algorithm 1: Optimization for Fan Speed

Get current $T_{amb}(k)$, P_{cpu}(k), T_{cpu}(k);

Use transient thermal model and FS_{min} to predict the $T_{cpu}(k+1)$ of next slot;

if $T_{cpu}(k+1) > T_{max}$ **then**

  Use stable thermal model to find $FS(k+1)$ that can stablize T_{cpu} at T_{max};

else

  $FS(k+1) = FS_{min}$

end

Return FS;

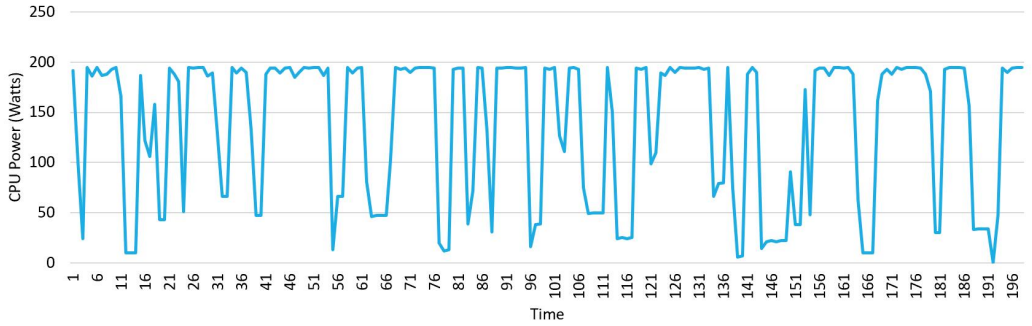

Figure 4.2: CPU Power Trace of a Server

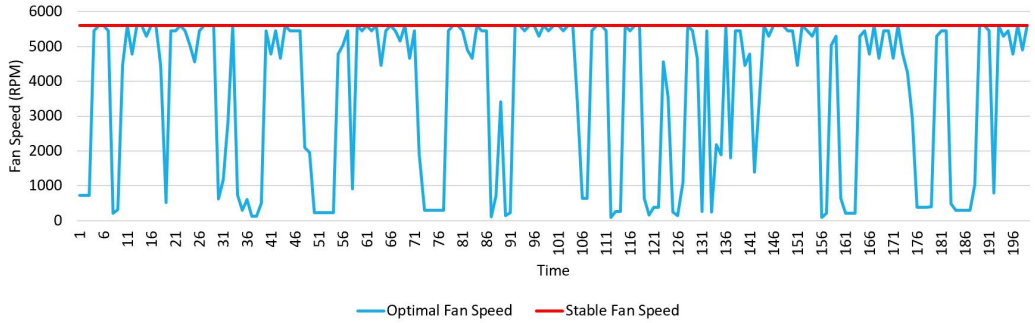

Figure 4.3: Comparison of Optimal and Stable Fan Speed ($T_{amb} = 25°C, T_{max} = 80°C$)

Here n represents the number of fans. The constraint is that the predicted temperature in the next slot does not exceed the given CPU maximum temperature, and the fan speed does not exceed its maximum speed. The Δt we use here is 1 sec granularity; this is based on the assumption that it is possible for a server fan to change its fan speed in 1 sec. While we are measuring the power for different fan speeds, it takes the system 2 to 3 seconds to deal with the racadm command line before changing the fan speed. Algorithm 1 presents the algorithm to find the optimal fan speed.

Figure 4.2 shows a CPU power trace of a server, and Figure 4.3 gives the comparison of the corresponding optimal fan speed and stable fan speed.

The stable fan speed is chosen as a fixed speed that can keep the CPU temperature just under the maximum. As shown in figure 4.3, the optimal fan speed varies with the CPU power. Then in figure 4.4 we can see the temperature of the optimal fan speed is much more stable and higher compared to stable fan speed. Figure 4.5 gives the comparison of the

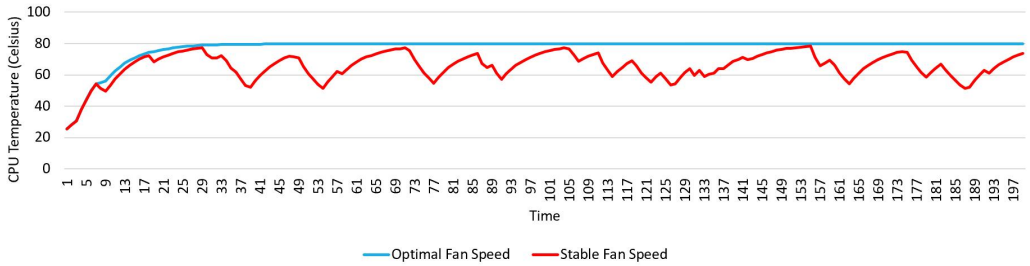

Figure 4.4: CPU Temperature Comparison of Optimal and Stable Fan Speed ($T_{amb} = 25°C, T_{max} = 80°C$)

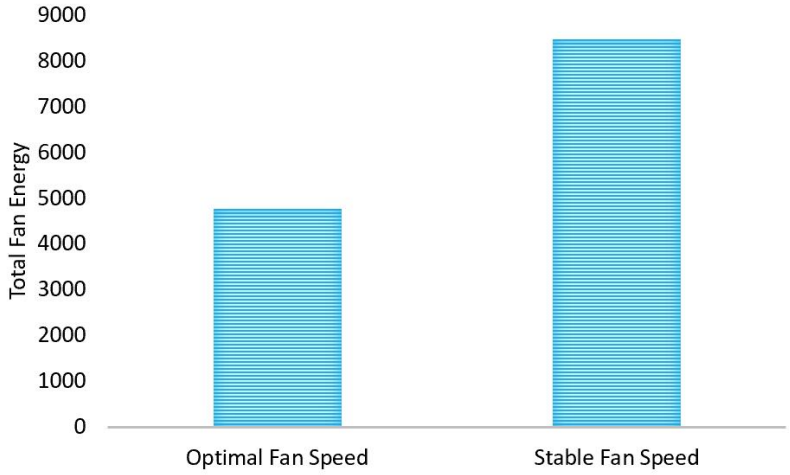

Figure 4.5: Total Energy of Optimal and Stable Fan Speed ($T_{amb} = 25°C, T_{max} = 80°C$)

energy cost. We can see that the optimal fan speed saves about 40% energy compared to stable fan speed.

Previously in Section 2.3, we demonstrated that WASP can have a better energy saving compared to A-I. We also mentioned that this is done by concentrating jobs into a few servers. However, since the power of fan is a cubic function, making a few CPUs run at full utilization might greatly increase the cooling energy.

4.2.1 Optimal Fan Speed for Small Latency Jobs

Figure 4.6 compares the fan power of 10 servers under 3 levels of utilization for small latency jobs. We can see that when the utilization is low, the fan power of A-I is larger than

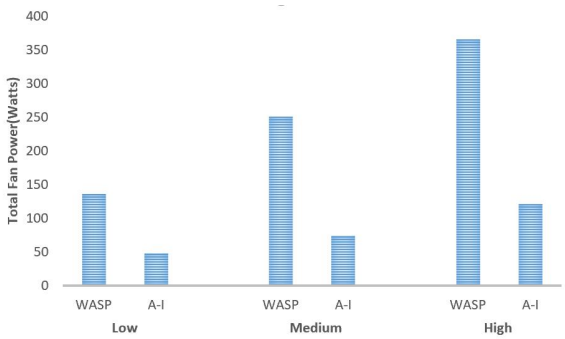

Figure 4.6: Total Optimal Fan Power for WASP and A-I (Small Latency Jobs, $T_{amb} = 25°C, T_{max} = 80°C$)

WASP, because in this case the demands of cooling system from both algorithms are low. WASP can save more power by putting the servers into system sleep (the fan will be shut down in this state). When the utilization grows higher, the average utilization of WASP's awake servers grows much faster than A-I, and the demands on the cooling system also grow rapidly.

However, in this system, even if the fan runs at its maximum speed, the cost is still much smaller than the power difference between system sleep state and package sleep state.This means that it is not advantageous to increase the number of awake servers just to save fan power. Though WASP will cause a higher demand on the cooling system, we can still use that as a foundation to improve the optimal fan speed algorithm to decrease the negative impact on cooling system caused by WASP.

4.2.2 Optimal Fan Speed for Large Latency Jobs

Similarly, Figure 4.7 shows the total power comparison between WASP and A-I under large latency jobs. Here, the behavior is opposite to that for low latency jobs. The fan power of WASP can be almost 30% of the power of A-I under low utilization, and it still can save about 40% of power under high utilization.

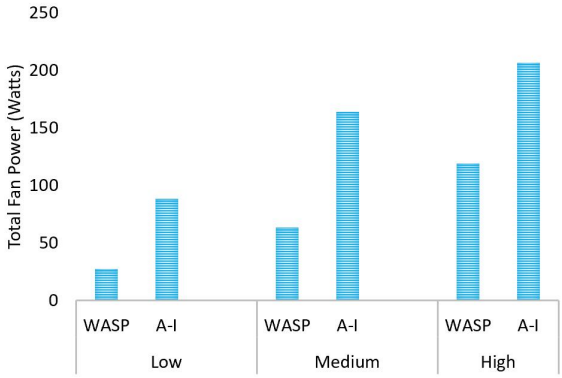

Figure 4.7: Total Optimal Fan Power for WASP and A-I (Large Latency Jobs, $T_{amb} = 25^{\circ}C, T_{max} = 80^{\circ}C$)

4.2.3 Comparison of the CPU Power Trace for Different Job Sizes

From Sections 4.2.1 and 4.2.2 we can see an obvious difference between the cooling energy of small and large latency jobs. However, from figure 2.5 we know that the powers under these two kinds of jobs are similar under same utilization. From the average or total system power it is hard to explain why there exists such a huge difference in cooling energy.

The only difference is the CPU power pattern. Figure 4.8 shows the CPU power trace of two different sizes of jobs under low utilization. The CPU power trace of large latency jobs tends have less variation, while the CPU power of small latency jobs tends to vary from its minimum to its peak power. The average power for these two kinds of jobs is almost the same. However, since the CPU power for small latency jobs stays at peak power once the CPU is in active period, it will cause a huge cooling pressure, compared to a lower but stable CPU pattern (large latency jobs). This is the reason why it will cost more cooling energy while having similar average power with large latency jobs.

4.3 Pre-cooling

The WASP algorithm has a unique power pattern for its awake servers under small latency jobs. For A-I, since there is no need for the server to enter deeper sleep state, the arriving

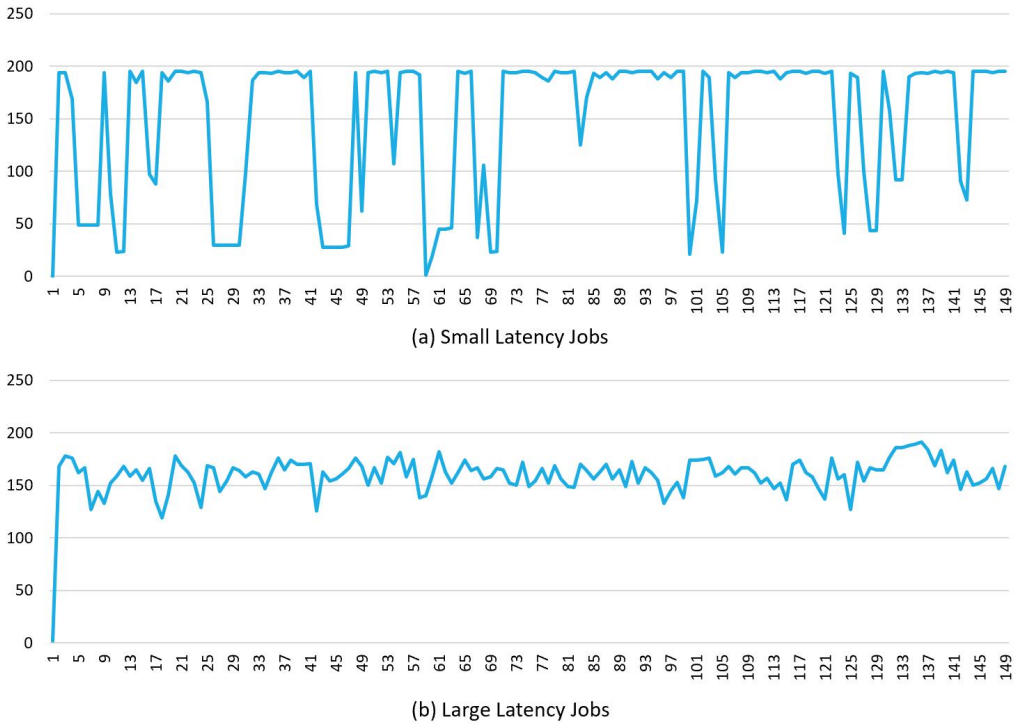

Figure 4.8: The CPU Power Trace of WASP under Different Size of Jobs and Low Utilization

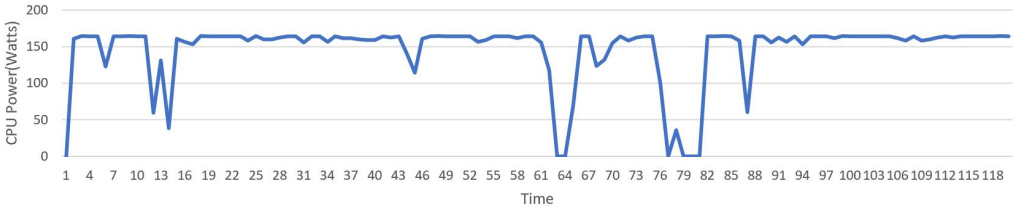

Figure 4.9: A CPU Power Trace of WASP

tasks will not be concentrated into a few servers. However, for WASP, in order to decrease the number of active servers/CPU, it will try to concentrate tasks to a few servers and allow the rest of them enter a deeper sleep state. Combined with what we discussed in Section 2.2, WASP will always tend to concentrate the tasks into $N_a - 1$ servers and let the remaining server enter package sleep state. The result is that these N_a servers will go into package sleep state in turn, and the utilization of active servers will be close to 100 percent.

Figure 4.9 shows this pattern of CPU power of a single server. We can divide this pattern into several cycles, as shown in figure 4.10. Then the CPU power trace of WASP can be

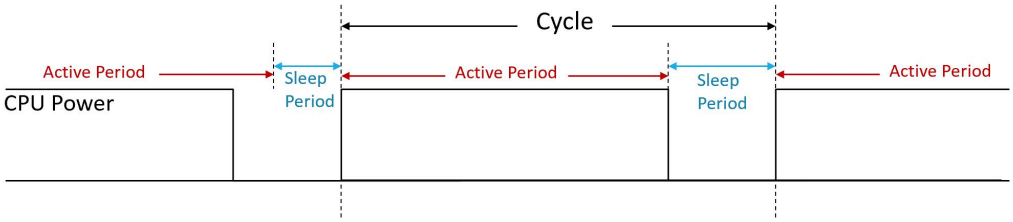

Figure 4.10: Active and Sleep Periods of CPU

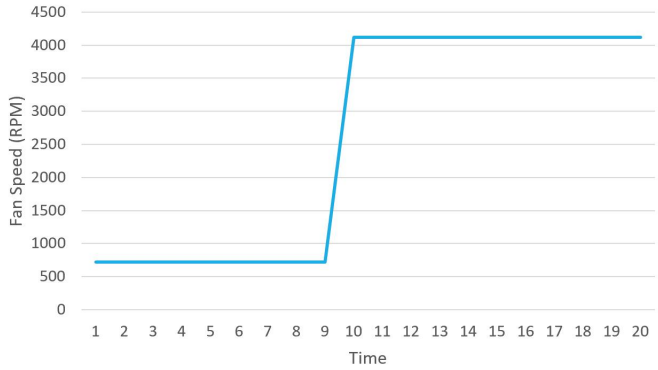

Figure 4.11: Optimal Fan Speed during Active Period in a Cycle

seen to be consisting of several cycles with different active period and package sleep state period.

If we are using optimal fan speed (Algorithm 1) in Section 4.2 for this cycle, the fan speed stays low until it is predicted that the CPU going to overheat. As shown in figure 4.11, the fan speed will then rise rapidly and stay at a high level for a long time.

Since the power has a cubic relationship with the fan speed, such a large increase may not be the most energy-saving fan speed for such a cycle. In the following, we optimize the overall fan power and then obtain the optimal fan speed by solving the optimization. using AMPL 4.6.

$$\min \sum_{k=0}^{T_c} P_{FS}(k) \tag{4.6}$$

$s.t$ for k in $[0, \lfloor \alpha T_c \rfloor]$: (4.7)

$$T_{cpu}(k+1) - T_{amb} = T_{cpu}(k) - T_{amb} - \frac{\Delta t C_2}{C1R_j(k)}(T_{cpu}(k) - T_{amb}) - \frac{\Delta t}{C_1}Q$$

for k in $[\lceil \alpha T_c \rceil, T_c - 1]$: (4.8)

$$T_{cpu}(k+1) - T_{amb} = T_{cpu}(k) - T_{amb} - \frac{\Delta t C_2}{C1R_j(k)}(T_{cpu}(k) - T_{amb})$$

for k in $[0, T_c - 1]$: (4.9)

$$FS_{min} \leq FS(k) \leq FS_{max}$$

for k in $[0, T_c - 1]$: (4.10)

$$P_{FS}(k) = a_0 * FS^3(k) + a_1 * FS^2(k) + a_2 * FS(k) + a_3$$

for k in $[0, T_c - 1]$: (4.11)

$$T_{cpu}(k) \leq T_{max}$$

$$T_{cpu}(0) = T_{initial} \tag{4.12}$$

In this optimization, T_c is the total time of a cycle, while α represents the utilization of this cycle. The constraints (4.7) and (4.11) require that every prediction of the CPU temperature during the active period ($[0, (1-\alpha)T_c]$) does not exceed the maximum temperature. Similarly, constraint (4.8) and (4.11) give the requirement during package sleep period.

The difference between this and the optimization in Section 4.2 is that, for this optimization, there exists a prediction of the upcoming active period. By analyzing the CPU power trace generated by WASP, we can know that for most of the cycle, the active period is at least $T_{a_{min}}$, and for package sleep state the duration should at least be $T_{i_{min}}$. Then we can perform the optimization for the cycle for $T_c = T_{a_{min}} + T_{i_{min}}$ and $\alpha = \frac{T_{a_{min}}}{T_c}$.

Figure 4.12 shows the optimization results under different T_{max} and $T_{initial}$. As we can

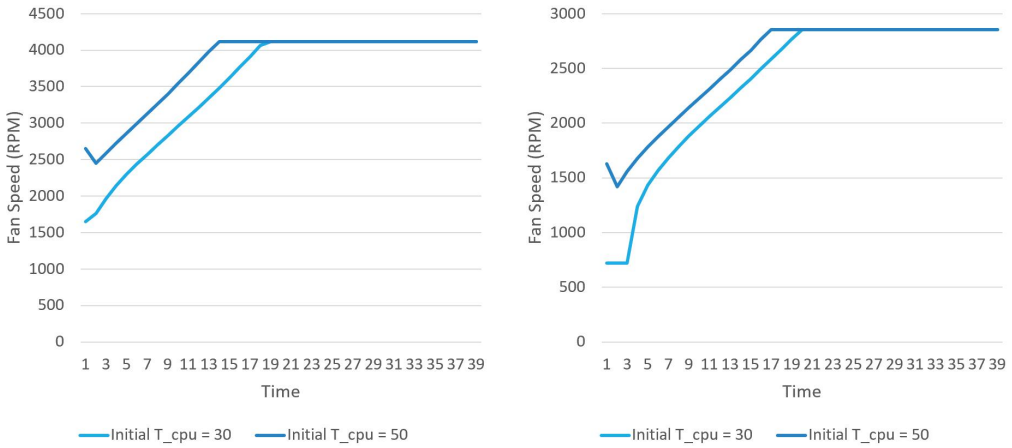

Figure 4.12: Pre-cooling Based Optimization Fan Speed under Different T_{max} and $T_{initial}$

Table 4.1: Increase Point for different T_{max} (Celsius)

T_{max}	85	83	80	78	75
increase point	67	60	51	41	34

see in this figure, the optimization for the fan speed can be described as a linear function. The slopes of these linear function are the same for the same T_{max}.

This optimization begins to increase the fan speed gradually long before the time when the temperature rises rapidly as in the optimization described in Section 4.2. This is because this optimization is based on a prediction that once the CPU wakes up, it will run at full utilization and last for at least $T_{a_{min}}$. Hence, we call this as a pre-cooling scheme.

However, this does not mean that once the CPU is awake, the fan will speed up immediately. We observe that there are different increase points for different T_{max}, as shown in Table 4.1.

Only if CPU is awake and the temperature grows larger than these increase points, the fan would begin pre-cooling. The initial fan speed for this also varies with different increase points, as shown in figure 4.13.

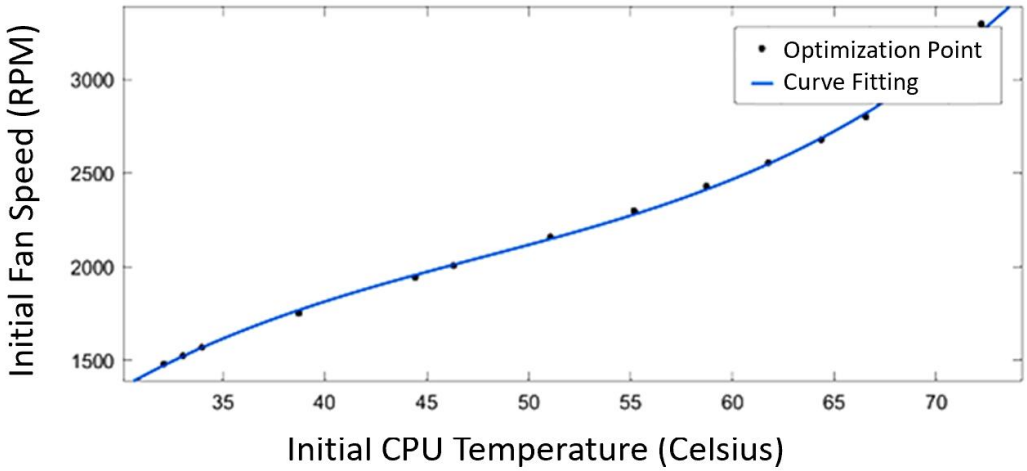

Figure 4.13: The Relationship between $T_{initial}$ and $FS_{initial}$

Algorithm 2: Pre-cooling Based Optimization for Fan Speed

Get current $T_{amb}(k)$, P_{cpu}(k), T_{cpu}(k);

Get current $T_{increase}$ and $F_{initial}$ according to T_{max}, generate linear function for pre-cooling fan speed;

Use transient thermal model and FS_{min} to predict the $T_{cpu}(k+1)$ of next slot;

if $T_{cpu}(k+1) > T_{max}$ **then**

    Use stable thermal model to find $FS(k+1)$ that can stablize T_{cpu} at T_{max};

else

    if $P_{cpu}(k) > P_{active}$ **then**

        if $T_{cpu}(k+1) > T_{increase}$ **then**

            Use linear function to generate $FS(k+1)$

        else

            $FS(k+1) = FS_{min}$

        end

    else

        $FS(k+1) = FS_{min}$

    end

end

Return FS_{min};

We now give a comparison of the optimization without (Optimization.1) and with pre-cooling (Optimization.2). As shown in Figure 4.14, this gives the optimal fan speed during a 20-sec active period under different initial points. We can see the optimization without pre-cooling tends to have a large increase, while optimization with pre-cooling will increase the fan speed gradually at the very beginning. Optimization.2 also makes the trend of CPU temperature through this pre-cooling (4.15).

Figure 4.16 gives the corresponding fan power. For Optimization.1, the power trace also shows a huge increase, and the power keeps at a high level for a long time. while the fan power for Optimization.2 tends to follow a smoother curve. Though it starts increasing

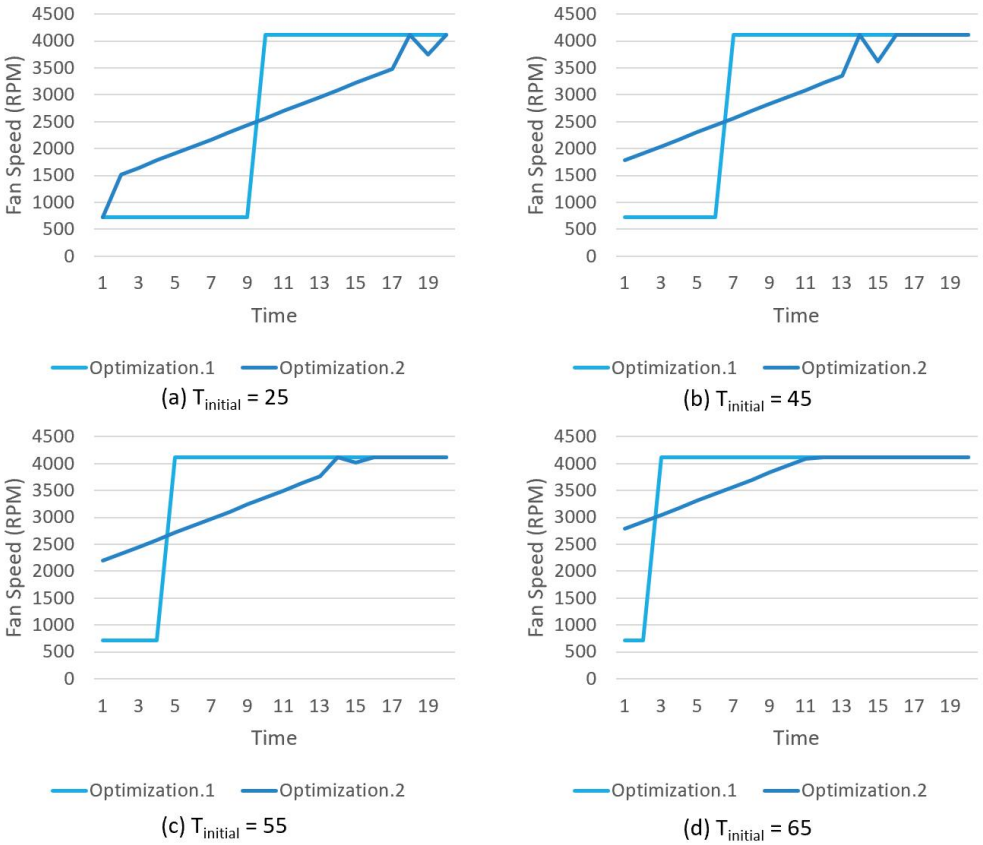

Figure 4.14: The Optimization Fan Speed with and without Pre-cooling Scheme

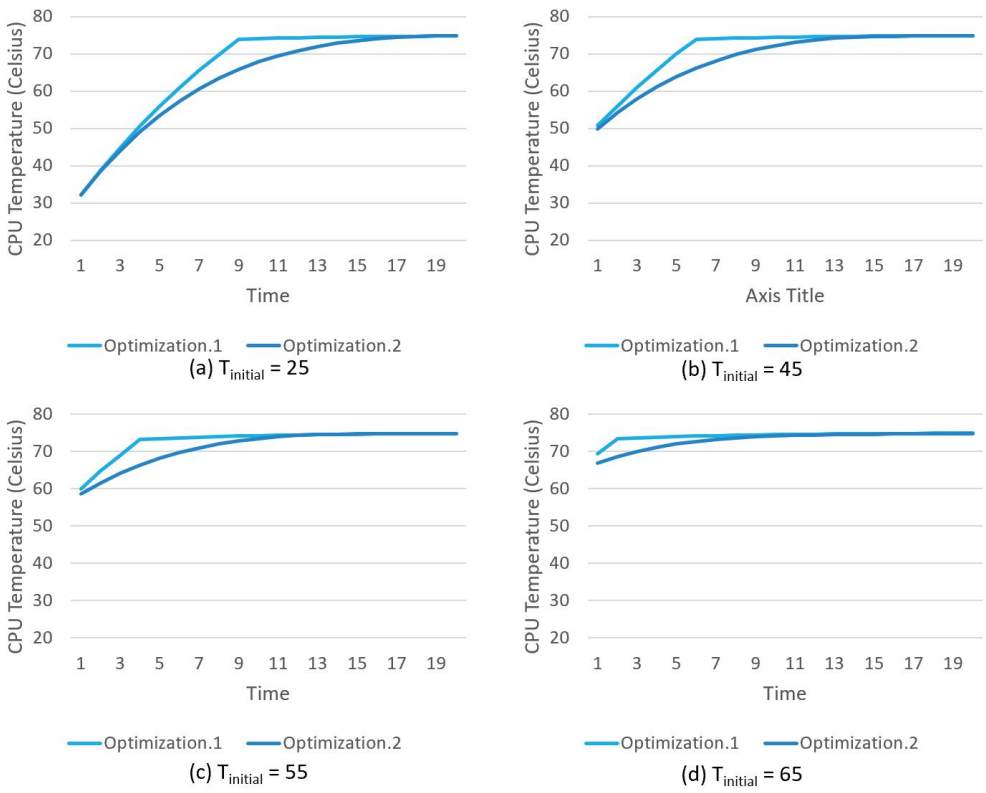

Figure 4.15: The CPU Temperature under Different Optimization Fan Speed

much earlier than Optimization.1, for most of the time it keeps running at a fairly low level, causing a large energy difference between the two optimizations.

As shown in figure 4.17, during this 20 sec active period, Optimization.2 can save about 20% of the energy compared to Optimization.1, and the energy-saving efficiency is better when the initial temperature is lower.

4.4 The Energy Demand for WASP and A-I under Pre-cooling Based Optimal Fan Speed

We now give energy comparison between A-I and WASP. Since there is no predictable pattern for A-I, we only use Optimization.1 to generate fan speed for A-I. As shown in figure 4.18, the energy saving efficiency of Optimization.2 is better for lower utilization; for 30% utilization it can almost eliminate the negative impact from WASP.

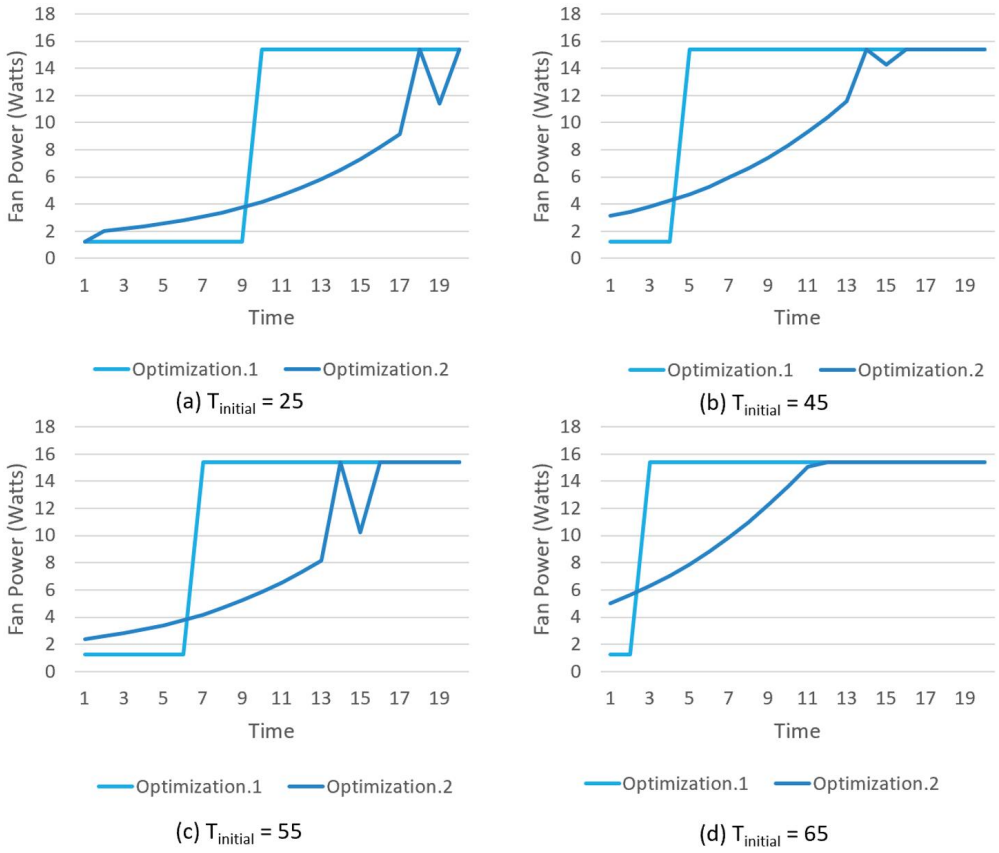

Figure 4.16: The Fan Power under Different Optimization Fan Speed

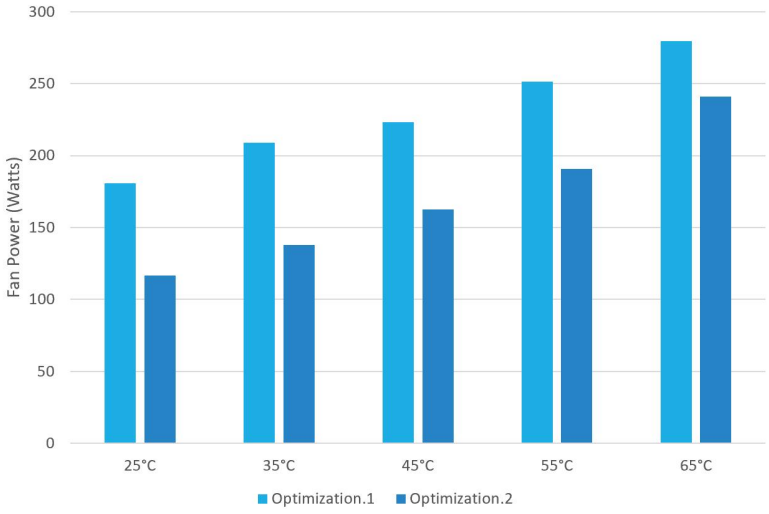

Figure 4.17: Total Fan Energys for different $T_{initial}$

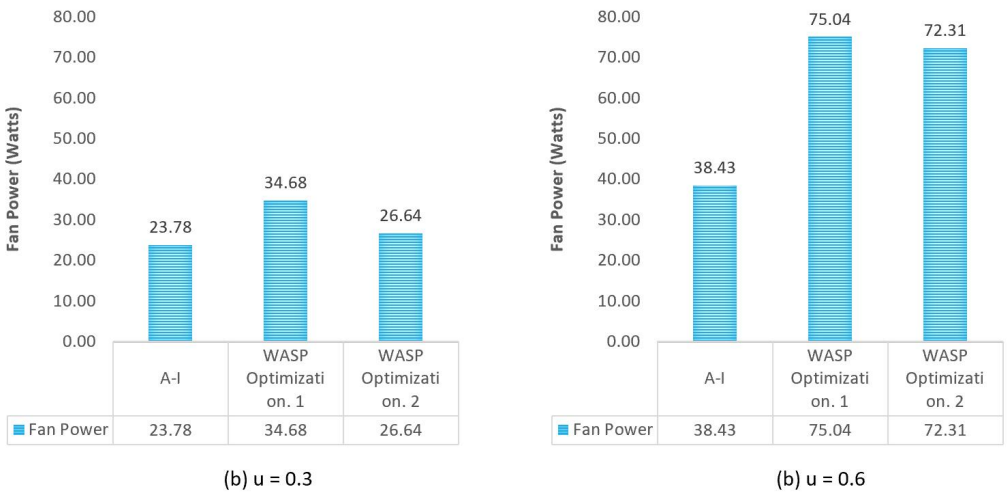

Figure 4.18: Fan Power Comparison for WASP and A-I under Different Optimization Fan Speed

In Section 4.3 we mentioned that pre-cooling works better when the initial temperature is lower. However, the initial temperature in a whole CPU power trace equals the temperature at the end of each package sleep state period. A lower initial temperature means a lower average temperature, which also means a lower CPU power. Though Optimization.2 can save more energy when the utilization of the system is low, it will not save much because the demand on the cooling system is also quite low in this case.

The pre-cooling based optimization can save significant energy compared to stable fan speed, and reduce the negative impact on cooling system caused by WASP, but it still cannot eliminate it. In the next chapter we will discuss how WASP can be further improved.

Chapter 5: Limiting Active Cores for Energy Optimization

To save energy, WASP tends to concentrate tasks to as few servers as possible, so that the other servers will have more opportunities to enter a deeper sleep state. The assumption in WASP is that all cores in an awake server are active. However, note that the server power (and hence the fan power) can be decreased by limiting the number of active cores in the server. This will of course mean that either more servers must be kept awake or the active servers take longer to process the jobs. The goal of this chapter is to explore the potential benefits that could be obtained by limiting the number of active cores per awake server.

The fan power rapidly increases with CPU power. For example, suppose the power of a 100% utilized core is about 4 Watts; if the CPU power rises from 100W to 104W, the corresponding fan power will rise from 1.51W to 1.57W according to the stable thermal model (3.1), but if the CPU power rises from 190W to 194W, the fan power will increase rapidly from 27.4W to 38.9W. This means that if we set a core limitation, moving tasks from a highly-utilized CPU to a less utilized CPU, might decrease the cooling power. However, to save energy, we should also consider the overall active time, which will increase under core limitation.

5.1 The Energy Saving Efficiency of Core Limitation

Core limitation decreases the peak power of a CPU, but its energy savings need to be investigated.

5.1.1 The Constraint on Core Limitation

The constraint on the core limitation is that the utilization of available cores must not exceed 100%, as explained in Chapter 2, because the QoS rapidly degrades as the utilization approaches 100%. If N is the number of servers, U is the utilization, C_{max} is the number of

Table 5.1: Execution Time under Different Core Limitation of WASP and A-I

-	Core Limitation	Utilization		
		Low	Medium	High
WASP	28	500.0661	300.0541	214.3376
	27	500.0688	300.0615	214.3465
	26	500.0542	300.0657	214.3371
	25	500.0556	300.0474	214.3472
	24	500.0575	300.0517	-
	23	500.0623	-	-
	22	500.0669	-	-
A-I		500.0476	300.0417	214.3254

available cores per server, and $C_{limitation}$ represents the core limitation (i.e., number of cores to which jobs are assigned), $u_{C_{limitation}}$ is the corresponding utilization for $C_{limitation}$ available cores.

$$u_{C_{limitation}} = \frac{C_{max}NU}{C_{limitation}N_a} \tag{5.1}$$

For example, if $N = 10$ servers, $U = 0.3$, N_a is 4 (according to (2.2)) and the utilization per awake server is 0.75, which is also the utilization for 28 cores ($u_{28} = 0.75$). $u_{C_{limitation}}$ increases when $C_{limitation}$ decreases. When $C_{limitation} = 21$, u_{21} = 1, in this case every awake server is fully occupied under core limitation, the job queues grow to infinity, and the latency increases rapidly.

5.1.2 The Energy Saving under Core Limitation

The energy effects of core limitation can be seen as two parts, the first part is the energy increase of the system energy, the second part is the energy saving for cooling system. This is an energy trade-off; here we give energy comparisons for different core limitation and utilization.

Figure 5.1 shows the system energy with different core limitations for WASP and A-I

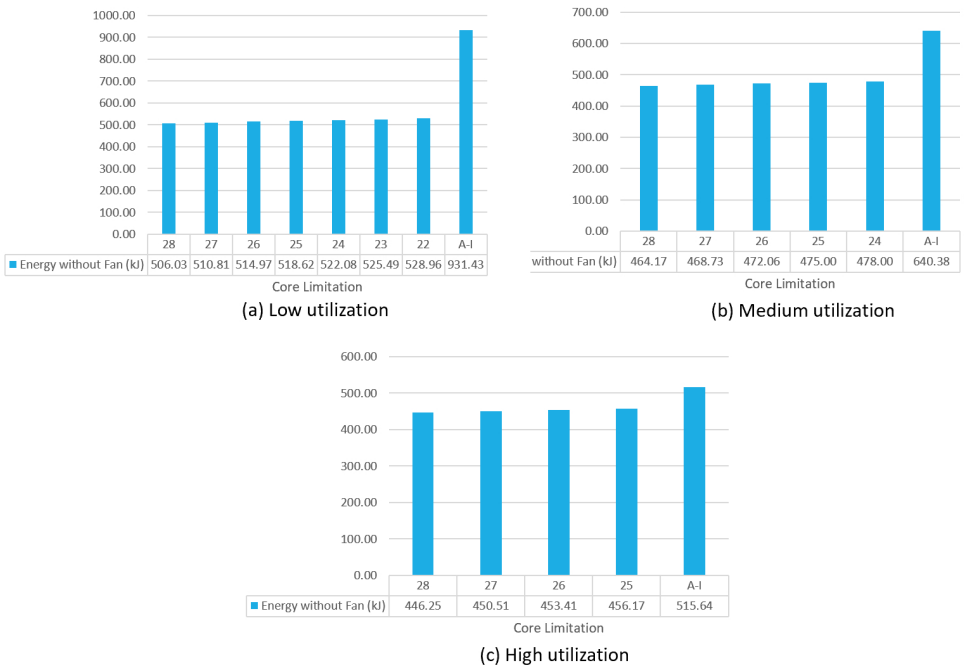

Figure 5.1: Energy without Fan under Different Core Limitation of WASP and A-I ($T_{amb} = 25°C, T_{max} = 80°C$)

($T_{amb} = 24$ and $T_{max} = 80$). The system energy increases as the core limitation decreases; nevertheless, the system energy of WASP is always lower than that of A-I. As shown in Table 5.1, the job execution times for different core limitations are almost the same, hence the system energy and system power have a similar trend.

The required fan energy for different core limitations is shown in figure 5.2. We can see the trend of fan energy is also like a cubic function with the core limitation; it increases rapidly as the core limitation is increased.

Figure 5.3 shows the total energy comparison for different core limitations for WASP and A-I. The best core limitation is the same for the three utilization levels. As shown in the figure, the core limitation can decrease, or even eliminate the negative impact on cooling system caused by WASP. Under a 26-core limitation, we can observe that such limitation can save more than 80% cooling energy under a low utilization, the energy saving efficiency is still good (70%) under high utilization. The increase of the system energy is small, only

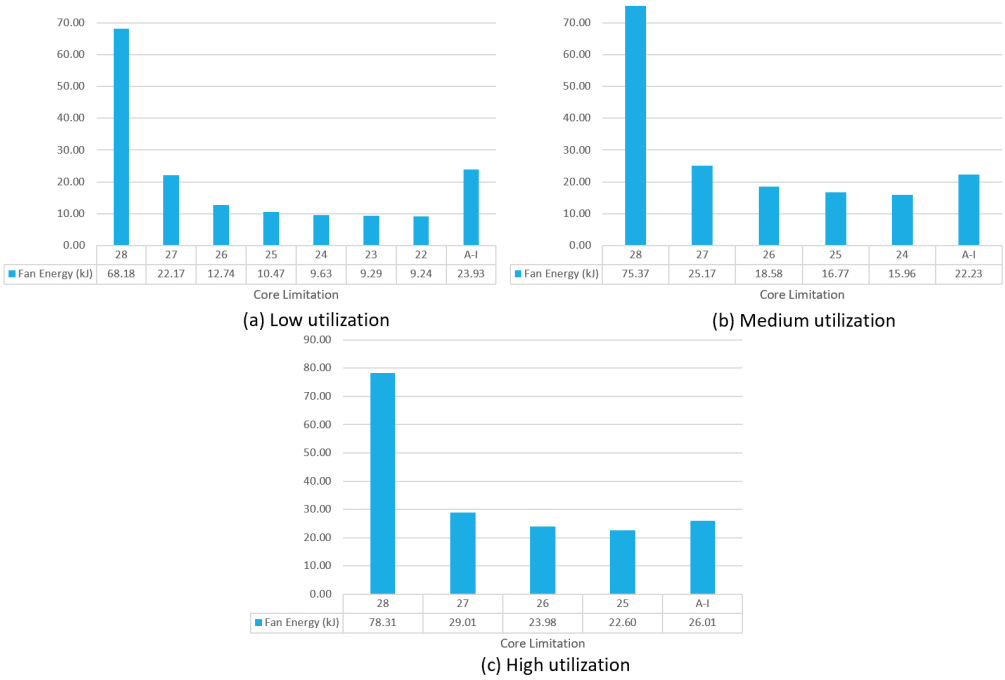

Figure 5.2: Fan Energy under Different Core Limitations for WASP and A-I

Table 5.2: Best Core Limitation under Different Utilization ($T_{amb} = 25°C, T_{max} = 80°C$)

UTILIZATION	0.2	0.3	0.4	0.5	0.6	0.7	0.8
BEST CORE LIMITATION	26	26	26	26	26	26	26

around 1% under this best core limitation.

The core limitation here helps to save around 10% total energy compared to the energy for 28 cores. It eliminates the impact of WASP on energy saving completely, and helps to save more cooling energy compared to A-I.

We also present the best core limitations under different utilizations in Table 5.2. The results show that the optimal limitation does not depend on the utilization.

5.1.3 Best Core Limitation under Different Environments

The best core limitation varies according to T_{amb} and T_{max}, as shown in Table 5.3. As shown in the table, the best core limitation increase as the difference between T_{amb} and T_{max} grows

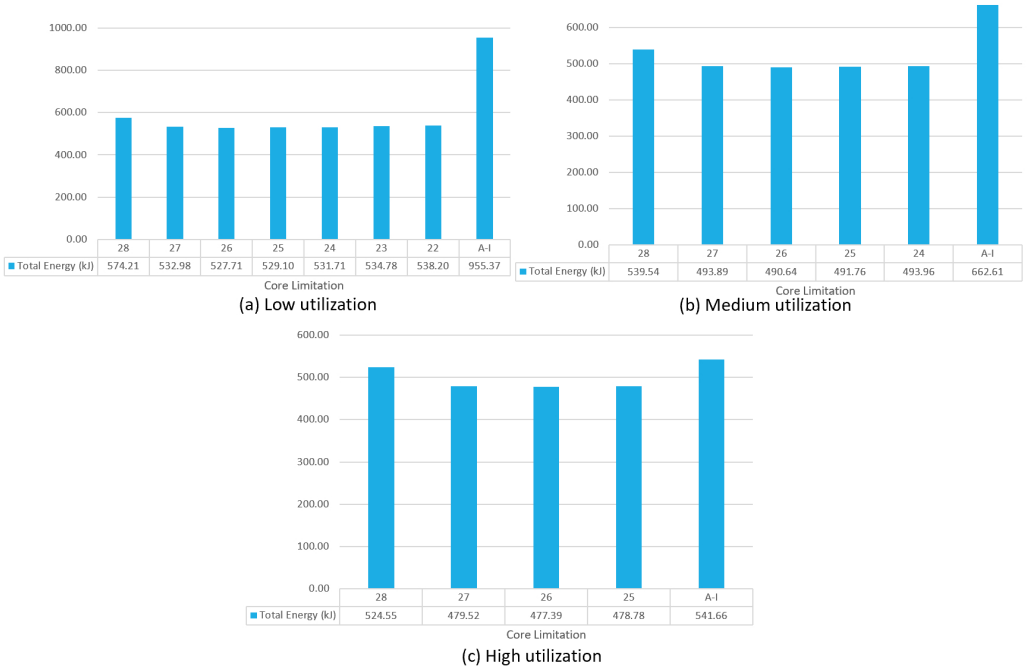

Figure 5.3: Total Energy under Different Core Limitation of WASP and A-I ($T_{amb} = 25^{\circ}C, T_{max} = 80^{\circ}C$)

larger.

From table 5.3 we see that the core limitation varies with job size.

5.1.4 The Essence of Core Limitation

We show a CPU power trace of 26-core limitation for small latency jobs in figure 5.4. Comparing this with the power trace of figure 4.8, we can observe that the core limitation actually make the power trace more stable (i.e., has less variation). Though the average power for figure 5.4 is even higher than that in figure 4.8(a), it avoids long period at peak power. This is also the reason large latency jobs perform better in terms of energy efficiency of cooling system. A stable power trace can reduce the cooling demand compared to an unstable one (with similar average power).

Table 5.3: Best Core Limitation for Different Temperatures and Job Sizes

Job Size (ms)		4.2			194		
$T_{max}(^{\circ}C)$		75	80	85	75	80	85
$T_{amb}(^{\circ}C)$	20	26	27	28	25	26	28
	25	24	26	27	23	25	26

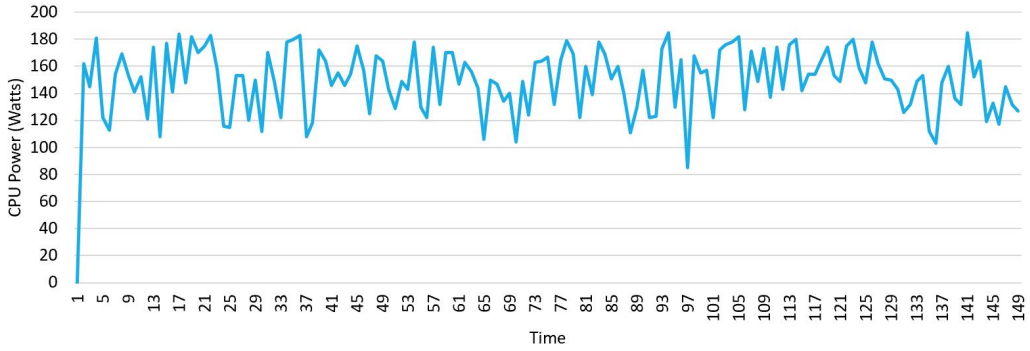

Figure 5.4: CPU Power Trace of 26-Core Limitation for Small Latency Jobs ($T_{amb} = 25^{\circ}C, T_{max} = 80^{\circ}C$)

5.1.5 Indirect Energy Saving on CRAC

As shown in figure 5.3, the core limitation for $T_{amb} = 25°C$ and $T_{max} = 75°C$ is 24 for small latency jobs. Taking medium utilization for example, the average cooling power per server under this limitation is around 15W, while the cooling power without limitation is 186W. For a 2-fan server, it is hard for a server fan to generate over 90W (the rated power for a server fan is about 36W). In this case, the only way to control the CPU temperature under T_{max} without core limitation is to lower the ambient temperature, which will stress the CRAC system and increase its power.

Chapter 6: Discussions

In Chapter 2, we introduce WASP and demonstrate its energy saving efficiency compared to A-I. In order to save more energy, WASP assigns jobs to as few servers as possible. Such a scheme forces those servers to run at maximum utilization while they are in active state. We then build a thermal model and fan power model in Chapter 3, and find the optimal fan speed for given T_{amb}, T_{max}, and P_{CPU}. However in Section 4.2, the scheme that WASP uses to save energy is then found to be disadvantageous for the cooling system (for small latency jobs). When the utilization is high, the cubic relationship between fan speed and fan power rapidly amplifies the impact of increase of CPU power.

It is possible to reduce the energy of the cooling system by adding more awake servers for WASP, which allows these servers to enter system sleep state. During this sleep state, since both CPU and fan are not working, the ambient environment can make a lrge impact in cooling down the system. However, as shown in Table 1.2, the wake-up latency for system sleep state can be 5 sec, and the power during this wake-up period is almost half of the peak power. If WASP adds more awake servers (N_a), the average utilization of these servers will be low. Putting them into system sleep state from time to time will waste power due to the large wake-up latency, not to mention this will also cause a latency problem. Keeping them awake will cause a large amount of base power (system power without CPU). Cooling fans do consume a significant part of the energy of a data center server, for example, the power of the cooling fans can be about 220W while the CPU power is around 380W [14]. However, even in this case the fan power only takes 18% of the total server power(1203 W). Through increasing N_a, WASP can save more fan power, but it will not be enough to balance the increase of the system power.

Based on this observation, we suggested two improvements to save cooling energy. Pre-cooling scheme enhances the energy saving efficiency of the optimal fan speed for WASP.

One of the features of WASP is that, once a server is active, its utilization is high (around 100%). Though such behavior puts a large demand on the cooling system, it makes the power variation of CPU predictable. It was then shown in Section 4.3 that, on the premise of a certain duration of high CPU power, it is more energy-efficient to increase the fan power gradually long before the CPU temperature rises to a high level. We then compared the optimization with and without pre-cooling scheme, and showed that pre-cooling can improve the energy saving efficiency. However, there exists a conflict. Pre-cooling works better for low utilization; in this case sometimes it can save about 30% of the optimal fan power (figure 4.18), but in this case it does not result is overall power savings as the fan power is small relative to the CPU power. When the utilization of CPU is high, it stresses the cooling system, but the energy-saving efficiency of pre-cooling scheme does not work as well.

Pre-cooling can be seen as a scheme to soften the impact caused by WASP. Though it is not efficient for WASP to increase N_a to save cooling energy, it might be another way to reduce the peak power of these N_a servers; in Chapter 5, we suggested using core limitation to save cooling energy.

The number of available cores of processors has increased rapidly in recent years [12]. There are some frameworks, such as Watts-inside [3], that study the application power and conduct optimization based on the power profiles for multi-core computing. Though the increase of cores offers a better performance, it might cause a huge demand on the cooling system. The essence of core limitation is to share jobs to other servers (among N_a awake servers) and stabilize the CPU power trace. By doing this, the peak power of these active servers decreases, while the total power increases because such behavior extends the total active period. But the energy increase caused by core limitation is much smaller than that of an increase of N_a, which makes it possible to make an energy trade off of system energy and cooling energy. We discuss the total energy under different environments and utilizations. The best core limitation varies with the environment; in some case the core limitation can

eliminate the impact caused by WASP, or even save more cooling energy compared to A-I.

Both pre-cooling and core limitation can save more energy – one is an improvement of the cooling strategy (change the algorithm to find the optimal fan speed), and the other is improvement of the WASP algorithm itself. The simulation results show that under these two improvements, WASP can save up to 45% energy at low utilization, 26% at medium utilization, and 12% at high utilization.

Chapter 7: Future Work

In this thesis, we demonstrated the energy saving efficiency of WASP, discussed its impact on the cooling system, and then suggested two improvements for WASP. We now present possible ways in which our work can be extended.

The pre-cooling scheme not only provides the optimal fan speed, but it can also be applied to the CRAC unit. In [16][15], pre-cooling scheme is used to enhance the optimal cooling strategy for smart buildings. Inside a smart building, the majority of the heat is generated by humans. The number of humans can be represented by the density of CO_2 in the air. A gray box model is used to identify the parameters for the thermal model inside the building. The foundation of the pre-cooling scheme is that the environment for future is predictable. Here the authors use Holt Winters triple exponential smoothing algorithm to predict the density of CO_2, and use weather forecast as the prediction of the ambient temperature. In our current work, the power and duration of active CPU under WASP algorithm are also predictable. However, if the workload for the data center can also be predictable, like in [15], once the heat trace inside the data center in a day can be predicted, we can apply the pre-cooling scheme both on CRAC and cooling fan of the servers.

The utilization was held constant in the simulations presented in this thesis. Though we have used a Poisson job arrival process to simulate the traffic in real data center, the arrival rate for traffic might change from time to time. This is also an important feature in data center and there are studies that perform optimization based on this feature. For example, the framework for data center [8][19][18] is designed to reconfigure the topology inside data center according to different traffic in order to minimize the average hops. WASP itself can also change its configuration according to different arrival rates. However, as we show in this paper, the best configuration for system energy might not be the best for the total energy.

The core limitation in this thesis is used as a hard limitation for WASP. For example, we

only compare the energy consumption of different core limitations from 28 to 22 in figure 5.1(a). If the core limitation drops down to 21, the average utilization for the awake servers will exceed 100%, and as discussed in Chapter 2, the latency of jobs will be very large and adversely impact QoS. In this case, to maintain the QoS, we can add more active servers or increase the core limitation. Exploring a soft core limitation provides more flexibility and could be another direction to pursue.

Bibliography

[1] Richard Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431. 2008.

[2] A Cengel. Heht transfer. 2007.

[3] Jie Chen, Fan Yao, and Guru Venkataramani. Watts-inside: A hardware-software cooperative approach for multicore power debugging. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 335–342. IEEE, 2013.

[4] Info-Tech Research Group et al. Top 10 energy-saving tips for a greener data center. *Info-Tech Research Group.*, 2007.

[5] Wei Huang, Malcolm Allen-Ware, John B Carter, Elmootazbellah Elnozahy, Hendrik Hamann, Tom Keller, Charles Lefurgy, Jian Li, Karthick Rajamani, and Juan Rubio. Tapo: Thermal-aware power optimization techniques for servers and data centers. In *2011 International Green Computing Conference and Workshops*, pages 1–8. IEEE, 2011.

[6] R Jorgenson. Fan engineering an engineer's handbook, 1961.

[7] Jungsoo Kim, Mohamed M Sabry, David Atienza, Kalyan Vaidyanathan, and Kenny Gross. Global fan speed control considering non-ideal temperature measurements in enterprise servers. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 276. European Design and Automation Association, 2014.

[8] Chong Liu, Maotong Xu, and Suresh Subramaniam. A reconfigurable high-performance optical data center architecture. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.

[9] Yanpei Liu, Stark C Draper, and Nam Sung Kim. Sleepscale: Runtime joint speed scaling and sleep states management for power efficient data centers. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 313–324. IEEE, 2014.

[10] David Meisner, Brian T Gold, and Thomas F Wenisch. Powernap: eliminating server idle power. In *ACM sigplan notices*, volume 44, pages 205–216. ACM, 2009.

[11] David Meisner and Thomas F Wenisch. Dreamweaver: architectural support for deep sleep. In *ACM SIGPLAN Notices*, volume 47, pages 313–324. ACM, 2012.

[12] Jungju Oh, Christopher J Hughes, Guru Venkataramani, and Milos Prvulovic. Lime: A framework for debugging load imbalance in multi-threaded execution. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 201–210. ACM, 2011.

[13] Donghwa Shin, Jihun Kim, Naehyuck Chang, Jinhang Choi, Sung Woo Chung, and Eui-Young Chung. Energy-optimal dynamic thermal management for green computing. In *2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers*, pages 652–657. IEEE, 2009.

[14] Arunchandar Vasan, Anand Sivasubramaniam, Vikrant Shimpi, T Sivabalan, and Rajesh Subbiah. Worth their watts?-an empirical study of datacenter servers. In *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–10. IEEE, 2010.

[15] Arun Vishwanath, Vikas Chandan, Cameron Mendoza, and Charles Blake. A data driven pre-cooling framework for energy cost optimization in commercial buildings. In *Proceedings of the Eighth International Conference on Future Energy Systems*, pages 157–167. ACM, 2017.

[16] Arun Vishwanath, Saverio Tripodi, Vikas Chandan, and Charles Blake. Enabling real-world deployment of data driven pre-cooling in smart buildings. In *2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, 2018.

[17] Zhikui Wang, Cullen Bash, Niraj Tolia, Manish Marwah, Xiaoyun Zhu, and Parthasarathy Ranganathan. Optimal fan speed control for thermal management of servers. In *ASME 2009 InterPACK Conference collocated with the ASME 2009 Summer Heat Transfer Conference and the ASME 2009 3rd International Conference on Energy Sustainability*, pages 709–719. American Society of Mechanical Engineers, 2009.

[18] Maotong Xu, Chong Liu, and Suresh Subramaniam. Podca: A passive optical data center architecture. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[19] Maotong Xu, Chong Liu, and Suresh Subramaniam. Podca: A passive optical data center network architecture. *Journal of Optical Communications and Networking*, 10(4):409–420, 2018.

[20] Fan Yao, Jingxin Wu, Suresh Subramaniam, and Guru Venkataramani. Wasp: Workload adaptive energy-latency optimization in server farms using server low-power states. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 171–178. IEEE, 2017.

[21] Fan Yao, Jingxin Wu, Guru Venkataramani, and Suresh Subramaniam. A comparative analysis of data center network architectures. In *2014 IEEE International Conference on Communications (ICC)*, pages 3106–3111. IEEE, 2014.

[22] Fan Yao, Jingxin Wu, Guru Venkataramani, and Suresh Subramaniam. A dual delay timer strategy for optimizing server farm energy. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 258–265. IEEE, 2015.

[23] Fan Yao, Jingxin Wu, Guru Venkataramani, and Suresh Subramaniam. Ts-bat: Leveraging temporal-spatial batching for data center energy optimization. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[24] Fan Yao, Jingxin Wu, Guru Venkataramani, and Suresh Subramaniam. Ts-batpro: Improving energy efficiency in data centers by leveraging temporal–spatial batching. *IEEE Transactions on Green Communications and Networking*, 3(1):236–249, 2018.