GentleCool: Cooling Aware Proactive Workload Scheduling in Multi-Machine Systems

Raid Ayoub, Shervin Sharifi, and Tajana Simunic Rosing
Department of Computer Science and Engineering
University of California, San Diego
Email: {rayoub, shervin, tajana}@cs.ucsd.edu

***Abstract*—In state of the art systems, workload scheduling and server fan speed operate independently leading to cooling inefficiencies. In this work we propose *GentleCool*, a proactive multi-tier approach for significantly lowering the fan cooling costs without compromising the performance. Our technique manages the fan speed through intelligently allocating the workload across different machines. The experimental results show our approach delivers average cooling energy savings of 72% and improves the mean time between failures (MTBF) of the fans by 2.3X compared to the state of the art.**

I. Introduction

Effective management of cooling is critical in modern data centers and enterprise servers. High temperature has a significant negative impact on reliability and leakage power as both depend exponentially on temperature [3]. Cooling costs of data centers are in the range of millions of dollars [4]. A forced convection method is typically used in high-end-servers to remove the excess heat. According to [5], fan power can reach up to 51% of the overall server power budget. As the CPU power density escalates, the air flow rate must increase by raising the fan speed. Unfortunately, maintaining high rate of air flow requires dissipating a substantial amount of power due to the cubic relationship between fan power and its speed [6]. Lowering fan speed has additional advantage of improving the fan lifetime. The results in [8] show that a 35% reduction in fan speed improves the fan lifetime by a factor of two.

Current data centers leverage virtualization to enhance fault and performance isolation, improve system manageability and reduce infrastructure cost [16]. Virtualization provides an opportunity to significantly improve utilization and efficiency of the resources by clustering the workload into as few servers as possible and shutting down the rest [20]. This leads to both significant energy savings due to powering less servers and to a large utilization increase on the remaining servers. However, this also dramatically increases chances of creating thermal hotspots in the system.

Modern job schedulers do not consider either thermal/cooling characteristics of the machines, or the inherent power characteristics of the workload running on the system. We propose a scheduling framework called *GentleCool*, a proactive multi-tier approach for significantly lowering the fan cooling costs in highly utilized systems through intelligently allocating the workload across different physical machines (PMs). *GentleCool* reassigns the workload at the virtual machine (VM) level as well as the CPU socket level. Our scheduling algorithm has two phases, *spreading* followed by *refinement*. The spreading phase lowers speeds of the fans through balancing the power density among the PMs/CPU, leading to cubic savings in power needed to spin up the fans. The refinement phase tries to maximize the use of fast fans by placing additional hot VMs/threads into those sockets and unloading from them less active threads while maintaining a similar total power, thus further reducing the cooling costs. To estimate the benefits of workload reassignment, we developed a novel cooling cost predictor that requires no runtime adaptations and can be employed at negligible cost. Our results show that *GentleCool* is able to achieve significant reductions in cooling power (72%) and MTBF (2.3X) compared to the state of the art scheduling policies.

II. Related work

Recently, a few fan control algorithms are suggested which operate based on closed loop fan control algorithms that change the fan speed based on the current reading of the thermal sensors [12]. The research in [10] suggests an optimal fan speed mechanism for the blade servers. The optimal speed is determined through convex optimization. A class of techniques has been suggested to improve the cooling efficiency in the data centers by minimizing airflow recirculation through the use of workload scheduling [13], [14]. However, these techniques did not consider the effect of scheduling on the server cooling costs. In [6], [15] methodologies are proposed for modeling the convection thermal resistance between the heat sink and ambient temperature as a function of the air flow rate, which we use here. In [16], an approach is presented for lowering the system energy costs in virtualized environments through focusing on the CPU power by running the system as fast as possible and then placing it in an idle mode. However, it does not consider the cooling costs. An interesting aspect of this work is a framework for characterizing the VMs at run time, which we leverage in this work.

To the best of our knowledge, there is no prior work on cooling aware dynamic workload scheduling at both the machine level as well as the socket level. The primary contributions of our work can be summarized as: 1) We propose a new multi-tier cooling aware dynamic workload scheduling algorithm which significantly reduces the cooling costs by controlling workload scheduling. 2) We develop a predictor for the fan power costs which doesn't need runtime adaptation and can be derived only based on the system characteristics.

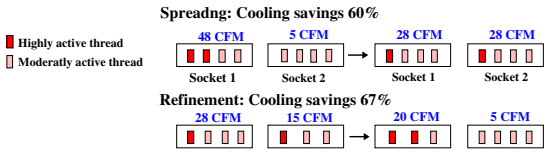

Fig. 1. Cooling aware scheduling at the socket level

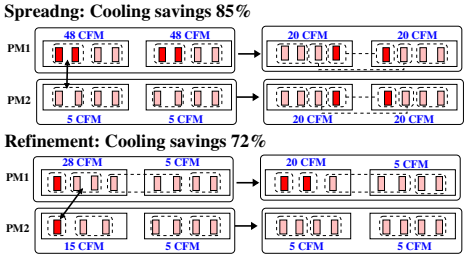

Fig. 2. Cooling aware scheduling at the VM level

III. Cooling Aware Hierarchical Workload Scheduling

A. Motivation

Energy of cooling subsystems could be reduced by intelligently distributing the workload across the physical machines as well as among each machine's sockets to manage the fan speeds. To illustrate this, we use two machines, each with two quad core sockets with two fans per socket. Two types of threads are executing, one highly active that consumes 14 Watts, and the other moderately active with 9.5 Watts. Temperature threshold is 85 ^{o}C, and ambient temperature is set to 42^{o}C. For thermal simulation, we use HotSpot [17], which we have extended to include the cooling model as described shortly in subsection C. Figure 1 shows the impact of workload assignment on cooling cost savings at the socket level. The left part of the figure shows the thread assignments by state of the art schedulers while the right part shows their assignments by *GentleCool*. At the top of this figure we show how to save the cooling cost when there is a big imbalance in the total power across the sockets. For such cases, the efficient solution is to balance power across the sockets. The savings are substantial and reach 60%. We call this *spreading*. In the second scenario, the air flow rate of socket 1 in the original assignment is about twice of that of socket 2. To minimize the cooling costs we can migrate the hot thread from socket 2 to 1 and migrate the two moderate threads from socket 1 to socket 2. The new assignment lowers the heat in socket 1 since the total socket power is reduced by 5 Watts while maintaining the maximum power at the same level. The savings are significant and reach 67%. We denote this class of reassignments as *refinement*. We further extend these concepts to the machine level through VM migration to enhance the cooling savings particularly in the cases where the socket level rescheduling is not beneficial. Figure 2 shows two cases, one for employing spreading and the other for refinement. In summary, scheduling the workload is an effective way to minimize the cooling energy.

B. Scheduling framework

The proposed framework is based on a multi-tier design where the top level reschedules the VMs across the available PMs while the lower level reschedules the threads among the CPU sockets. The objective is to reduce the total cooling cost of the system. The framework in [16] is used to characterize the VMs and their threads at runtime. The period for VM level rescheduling is in the order of minutes to incur minimal overhead on the system network while thread rescheduling at socket level is done at periods in the order of few seconds.

The cooling cost of the system is equal to the sum of cooling costs of its PMs, where the cost of each PM is proportional to the sum of the cubes of its fans' speeds. The fan speed of a socket depends on both the total socket power and the maximum power consumed by a core, as will be explained later. Therefore, to lower the heat we can minimize the maximum power, reduce the total power or both. To minimize cooling energy it is desirable to keep the fan speeds as low as possible. In order to do so, first the VMs are *spread* across different PMs in order to balance the power density and reduce the socket temperatures and their fan speeds. At this point, refinement is performed to further reduce the fans speed. The refinement phase lowers the cooling cost by focusing the workload on a smaller set of the fans (higher speed ones) while keeping their speed in a similar range. This is achieved by consolidating more hot workload into the sockets that are associated with those fans and unloading from them multiple moderately active workload to maintain similar total power.

The evaluation of potential cooling savings is performed by a predictor whose inputs are the power characteristics of the workload and the ambient temperature map of the PMs. The information required for predictions is periodically collected from the PMs and aggregated at the VM manager for VM rescheduling and at the hypervisor for the socket rescheduling. Algorithm 1 illustrates spreading at the VM level. The refinement phase operates in a similar manner, with the only difference that the $destPM$ is the PM with the highest $CoolingCost$ and $srcPM$ is the PM with the second highest $CoolingCost$. The spreading and refinement at the thread level follow the same algorithms, with the difference that PMs and VMs are replaced by sockets and threads in the socket level algorithm respectively.

Algorithm 1 VM level spreading

```
1:  while not all VMs are marked do
2:      srcPM ⇐ pick the PM with the highest CoolingCost
3:      destPM ⇐ pick the PM with the lowest CoolingCost
4:      while (not all VMs of srcPM are marked) do
5:          VM1 ⇐ the highest power unmarked VM of srcPM
6:          VM2 ⇐ the highest power unmarked VM of destPM
7:          VM3 ⇐ the second highest power unmarked VM of destPM
8:          evaluate the CoolingCosts when these migrations are done:
9:              migrating VM1 to destPM
10:             migrating VM1 to destPM and VM2 to srcPM
11:             migrating VM1 to destPM and {VM2 + VM3} to srcPM
12:         if (any new CoolingCost is lower than current CoolingCost) then
13:             do the migration resulting in the lowest CoolingCost
14:             mark the migrated VMs
15:         end if
16:     end while
17: end while
```

C. Cooling cost predictor

Figure 3 shows the thermal model of a single socket incorporating both the ambient temperature, T_{amb}, and the convective resistance, R_{conv}. The lateral thermal flow between the socket's cores is ignored due to the high ratio of the core area to the die thickness [11]. R_v is the die component's

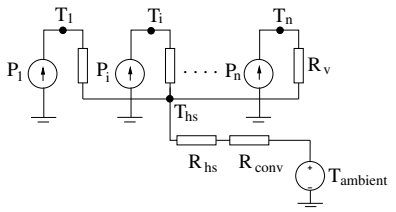

Fig. 3. Steady-state single socket thermal modeling

vertical thermal resistance including the thermal interface resistance. R_{hs} represents the sum of heat spreader and the heat sink resistances. To simplify the modeling of multiple sockets, the heat path between the individual sockets can be safely neglected since there is no effective conductive or convective heat path between them as they are typically placed apart in the motherboard. The convective resistance can be computed as $R_{conv} = \frac{1}{hA}$ where A is the heat sink area and h is the convective heat transfer coefficient. The value of h can be computed as $h(V) \propto V^{\alpha}$ where V is the air flow rate and α is constant in the range of (0.8-1.0). The cost of changing the convective resistance can be estimated based on the cubic relation between fan speed and power as $\frac{PF_2}{PF_1} = (\frac{R_{conv_1}}{R_{conv_2}})^{\frac{3}{\alpha}}$ where PF_1 and PF_2 represent the fan power dissipation at R_{conv_1} and R_{conv_2} respectively.

To design a low cost runtime predictor we utilize the fact that the heat sink represents a low pass filter with a very narrow bandwidth (range of 0.01 Hz since the heat sink time constant is in the order of tens of seconds) which allows only the average power to pass that is typically highly stable over a period of seconds. As a result, we can do steady-state analysis to allow for far prediction of the heat sink temperature while not compromising accuracy, an issue that is essential particularly for the machine level rescheduling events since they happen over periods of minutes. The predicted value of the convective resistance, R^{new}_{conv}, that is required to calculate the new air flow can be computed as follows:

$$R^{new}_{conv} = \frac{T_c - T_{amb} - P_{max}R_{core}}{\sum_{j=1}^{m} P_j} - R_{hs} \quad (1)$$

where T_c is the critical temperature threshold. P is the power dissipation of the individual components. A new method is developed to estimate the core power based on the temperature readings of their thermal sensors. The temperature is used since it reflects the density of both dynamic and leakage power. The temperature is averaged over the socket rescheduling period. The power of the individual cores, P_i, can be estimated as:

$$P_i = \frac{\beta_i T_i + (\sum_{j=1}^{N} \frac{\Delta_{ij}}{R_{core}}(R_{hs} + R^{cur}_{conv})) - \Theta}{R_{core} + N(R_{hs} + R^{cur}_{conv})} \quad (2)$$

where T^l_i is core thermal sensor reading, $\Delta_{ij} = (\beta_i T_i - \beta_j T_j)$, R^{cur}_{conv} is the current convective resistance, β is a factor to convert the thermal sensor readings of the core to its average temperature (can be estimated at the design time), and N is the number of cores in the die. The value of $\Theta = P_{extra}(R_{hs} + R^{cur}_{conv})$ corresponds to the contribution of L2 cache and the interconnect between the cores to the heat sink temperature. The power of these components, P_{extra}, can be estimated using the access frequency (estimated using processor performance counters) multiplied by the power of each access since they are highly regular. The power of each access can be estimated during the design time. The predicted cooling power, P^{new}_{cp}, can be estimated as $P^{new}_{cp} = (\frac{R^{cur}_{conv}}{R^{new}_{conv}})^{\frac{3}{\alpha}} P^{cur}_{cp}$, where P^{new}_{cp} is the current cooling power. The average prediction accuracy across the set of the benchmarks used in our results section is in the range of 5% which is good enough for our approach.

TABLE I
BENCHMARKS CHARACTERISTICS

Benchmark	Avg. power (W)	Standard deviation (W)
mcf	6.7	0.7
gcc	9.0	2.1
bzip2	13.4	0.3
perl	14.5	0.2

TABLE II
WORKLOAD AND AMBIENT TEMPERATURE ASSIGNMENT

VM assignment: PM1 to 4	Ambient temp. ^{o}C PM1 to 4
1: PM1(3*perl* + 2*bzip2*), PM2=PM1, PM3(3*gcc* + 2*mcf*), PM4 = PM3	41 each
2: PM1(3*perl* + 2*bzip2*), PM2=PM1, PM3(3*gcc* + 2*mcf*), PM4 = PM3	43 each
3: PM1(4*perl* + 4*bzip2*), PM2=PM1, PM3(4*gcc* + 4*mcf*), PM4=PM3	43 each
4: PM1(4*perl* + 4*bzip2*), PM2=PM1, PM3(4*gcc* + 4*mcf*), PM4=PM3	45,41,38,41
5: PM1(2*perl* + 4*gcc* + 2mcf), PM2(2*perl* + 4*gcc*), PM3 = PM1, PM4 = PM2	43 each
6: PM1(2*perl* + 6gcc), PM2=PM3=PM4=PM1	43 each

IV. RESULTS

To evaluate our algorithm we use four 45nm Intel Quad Core dual socket Xeon E5440 servers. These machines run Xen3.3.1 and use Xeno-Linux 2.6.18 for Domain 0. To estimate the core power we inserted current sensors in the power connector of each socket and collected the data using data acquisition system. These power values were then used to estimate the temperature using the HotSpot simulator [17] and the layout in [18].

Our algorithm, *GentleCool*, initiates VM rescheduling every 10 minutes and socket-level thread rescheduling every 5 seconds. A live migration scheme is used to migrate the VMs at a minimal cost [9]. *GentleCool* is compared to a state of the art *baseline policy* that implements VM and socket level scheduling. At the VM level Eucalyptus VM scheduler is emulated that assigns VMs using a greedy algorithm. At the socket level scheduling we emulate typical dynamic load balancing policy that balances the running threads across the CPU sockets as well as the socket cores as part of the base line policy. We use a closed loop feedback controller for the fan control that adjusts the fan speed based on the thermal sensors readings. The critical temperature is set to 85 ^{o}C. Quad Core Xeon E5440 package characteristics have been used in thermal simulation [15]. Two fans are associated with each socket that can provide max air flow of 53.4 CFM at max power of 29.4 Watts [19].

Benchmarks from the SPEC2000 suite have been used as workloads (see Table I). A set of benchmarks are selected that exhibit various levels of power intensity to represent real life applications. Each benchmark is run till the end and then repeated to construct a half-hour worth of execution. To construct the workload, we select VMs (see table II) that exhibit different power intensities and are combined in various ways.

Cooling savings: Figure 4 shows the cooling power savings of applying *GentleCool* over the baseline policy. As the results clearly show, applying *GentleCool* results in significant energy savings; the average improvement is **72%**. The savings in the case of *workloads* 1, 2, 3, 4 come from swapping the hot

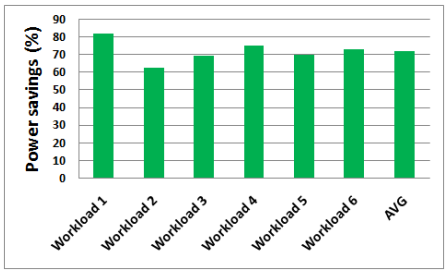

Fig. 4. Fan power savings

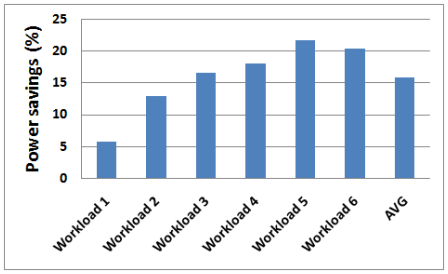

Fig. 5. Total savings including the CPU power cost

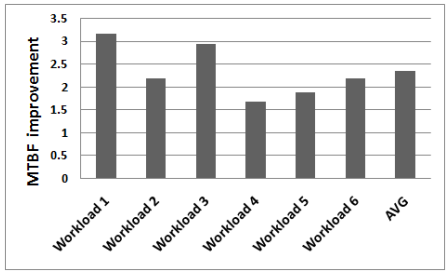

Fig. 6. Fan MTBF improvement

VMs with colder ones, spreading the hot VMs, followed by socket level spreading. For example, the savings in the case of *workload* 3 comes from swapping {4*perl*} VM/PM1 with {4*gcc*}VM/PM3 and also swapping {4*perl*} VM/PM2 with {4*gcc*}VM/PM4 since the fans in machine 1 and 2 run at the highest speed while the fans in machine 3, 4 run at very low speed due to the difference in the VMs induced power on the CPU sockets. The socket level rescheduling would follow to ensure a balance between the high and low activity threads with lower activity in each socket. The rescheduling in the case of *workload* 4 is different form the case of workload 3 due to the variations in the ambient temperature across the machines, even though the machines run the same workload in both cases. In this case, our *GentleCool* algorithm swaps the workload between the first and third PM since machine 3 has a low ambient temperature. It also swaps {4*perl*} VM/PM 2 with {4*gcc*}VM/PM4 as before. The savings in the case of *workload* 5 come from consolidating the VMs with high power density. *GentleCool* swaps the {2*perl*}/PM2 with {4*gcc*}/PM1 and then distributes one instance of *perl* to each socket. Similar rescheduling occurs between PM3 and PM4. For the case of *workload* 6, the savings are achieved by consolidating the hot threads within the sockets.

System power savings: To quantify the significance of our approach we evaluate the total savings including the CPU sockets power since the CPU is the primary component in the active power since the two sockets can draw up to 120+ Watts under intensive workload. Figure 5 shows the total savings with accounting the CPU power cost. The results show that the savings are appreciable; the average improvement is 16%. The savings in the case of *workload* 5 is the highest due to the large reduction in the fan power and the moderate CPU power in this case. The savings in the case of *workload* 1 is in the lower side since the fans original power before improvement was in the moderate range.

Fan reliability: The fan reliability results are obtained based on the model suggested in [8] that uses the fan speed reduction as a metric for extracting the MTBF. The results in Figure 6 show the MTBF improvement; the average enhancement is 235%. The improvements in the cases of *workload* 1, 3 are high since the drop in the fan speed.

Overhead: The overhead of running the *GentleCool* at the VM level is in the range of few tenths of milliseconds depending on the number of the machines in the system. This cost is negligible since this algorithm runs once every 10 minutes. The runtime of the socket level algorithm is in the range of few milliseconds, which allows it to run at a finer granularity (once every few seconds) on the same machine.

V. Conclusion

In this work, we have proposed a new workload scheduling technique to minimize the overall energy costs in multi-machine/multi-socket CPU platforms. Our algorithm incorporates continuous temperature measurements, workload characterization and cooling cost estimation to guide allocation of the workload in a thermally aware manner. GentleCool achieves cooling savings through spreading and refinement of the hot VMs/threads depending on the system's current thermal state. Our results show that GentleCool algorithm achieves cooling savings of 72% concurrently with improving the MTBF of fans by a factor of 2.35.

Acknowledgements

This work has been funded by NSF Project GreenLight grant 0821155, NSF SHF grant 0916127, MuSyC, UC Micro grant 08-039, Sun Microsystems, and Cisco.

References

[1] www.sun.com/servers/x64/x4270/

[2] www-03.ibm.com/systems/x/hardware/rack/x3550m2/

[3] M. Pedram et al. Thermal modeling, analysis, and management in VLSI circuits: principles and methods. *Proc. of the IEEE*, pages 1487-1501, 2006.

[4] C. Patel et al. Smart cooling of data centers. *Proc. IPACK*, 2003.

[5] C. Lefurgy, et al. Energy management for commercial servers *IEEE Computer*, pages 39-48, 2003.

[6] M. Patterson. The effect of data center temperature on energy efficiency. *Proc. ITHERM*, pages 1167-1174, 2008.

[7] R. Lyon, et al. Noise and cooling in electronics packages. *IEEE Trans on Components and Packaging Tech.*, Vol. 29, No.3, pages 535-542, 2006.

[8] G. Paparrizos. An Integrated fan speed control solution can lower system costs, reduce acoustic noise, power consumption and enhance system reliability. Technical report, *Microchip Technology Inc*, 2003.

[9] C. Clark, et al. Live migration of virtual machines. *Proc. NSDI*, pages 273-286, 2005.

[10] Z. Wang, et al. Optimal fan speed control for thermal management of servers. *Proc. IPAC*, pages 1-10, 2009.

[11] S.Heo, et al. Reducing power density through activity migration. *Proc. ISLPED*, pages 217-222, 2003.

[12] H. Chiueh, et al. A Novel Fully Integrated Fan Controller for Advanced Computer Systems. *SSMSD*, 2000.

[13] J. Moore, et al. Making scheduling "cool": temperature-aware workload placement in data centers. *Proc. USENIX*, pages 61-75, 2005.

[14] Q. Tang, et al. Thermal-aware task scheduling for data centers through minimizing heat recirculation. *Proc. ICCC*, pages 129-138, 2007.

[15] Quad-Core Intel Xeon Processor 5300 Series: Thermal/Mechanical Design Guidelines

[16] G. Dhiman, et al. vGreen: A System for Energy Efficient Computing in Virtualized Environments. *Proc. ISLPED*, pages 243-248, 2009.

[17] K. Skadron, et al. Temperature-Aware Microarchitecture: Modeling and Implementation. *Proc. TACO*, pages 94-125, 2004.

[18] http://www.digitalbattle.com/2007/11/12/intel-launches-45nm-cpus/

[19] http://www.sunon.com.tw/products/pdf/DCFAN/PMD4056.pdf

[20] A. Verma, et al. pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems. *Proc. Middleware Conference*, pages 243-264, 2008.