

Task Allocation for Minimum System Power in a Homogenous Multi-core Processor

Yang Ge, Qinru Qiu

Department of Electrical and Computer Engineering,
Binghamton University, State University of New York
{yge2, qqiu}@binghamton.edu

Abstract— In this paper we address the impact of task allocation to the system power consumption of a homogenous multi-core processor with a main focus on its impact on the leakage power and fan power. Although the leakage power is determined by the average die temperature and the fan power is determined by the peak temperature, our analysis shows that the overall power can be minimized if a task allocation with minimum peak temperature is adopted together with an intelligent fan speed adjustment technique that finds the optimal tradeoff between fan power and leakage power. We further propose a multi-agent distributed task migration technique that searches for the best task allocation during runtime. By choosing only those migration requests that will result chip maximum temperature reduction, the proposed framework achieves large fan power savings as well as overall power reduction. Experimental results show that, our agent-based distributed task migration policy can save up to 37.2% fan power and 17.9% system overall power compared to the random mapping policy when the temperature constraint is tight. When the temperature constraint is loose, the overall system power is insensitive to the task allocation.

Keywords—low power, task allocation, thermal aware, multi-agent distributed framework

I. INTRODUCTION

The ever-increasing power consumption of the computing system challenges the cooling system at all different levels. At data center level, the cooling infrastructure is becoming a limiting factor. The annual cooling cost for a large data center is considerably high and can reach up to tens of millions of dollars. At micro-architecture level, increased power density has set up a “Power Wall” which blocks the micro-processor’s performance improvement, and the clock frequency growth is restricted due to the thermal issue. To cool down the processor, a typical cooling fan can consume up to 51% power budget of a server [9].

Multi-core architecture has recently become the dominant design platform as it explores task and application parallelism in a power efficient way and hence relieves the power and thermal crisis. With the unprecedented number of transistors integrated on a single chip, the current multi-core trend may soon progress to hundreds or thousands of cores era. Examples of such system are the 80 tile network-on-chip that has been fabricated and tested by Intel [17] and Tiler’s 64 core TILE64 processor [1].

Even in a homogenous multi-core system, highly heterogeneous workload on different cores can produce local hotspot and create large thermal gradient. Elevated core temperature increases leakage current and stresses the cooling system. The cooling fan has to operate at a speed to accommodate the worst case power density and guarantee the chip temperature under a safe threshold anywhere and anytime. This would require the fan operating at higher speed to maintain fast air flow and strong heat dissipation ability. However, operating at high speed for long time consume more energy and reduce fan life time [3].

In this paper, we address the impact of task mapping on the overall power consumption of a homogenous multi-core system.

The system power consists of three components, dynamic power, static power, and fan power. Although different task mappings have little impact on the dynamic power in a homogeneous multi-core system, they do change the temperature distribution across the system and can potentially affect the leakage power and fan power. While the leakage power is determined by the average temperature, the fan power is determined by the peak power. Hence different optimization techniques are required. However, as we will show in Section IV that the impact on leakage power from task mapping is negligible if the fan convection speed is fixed. For a given workload, the chip leakage power can be approximated to a linear function of the convective resistance of the cooling system while the fan power is an inverse cubic function of the same parameter. Our analysis shows that the overall power can be minimized if a task allocation with minimum peak temperature is adopted together with an intelligent fan speed adjustment technique that finds the optimal tradeoff between fan power and leakage power. Furthermore, the impact of task allocation on the overall system power is significant when the temperature constraint is tight. When the temperature constraint is loose, the overall system power is insensitive to task allocation.

We further investigate the techniques to search for the task allocation for minimum peak temperature. We formulate the task allocation problem as a zero-one linear programming. Because solving the binary linear programming problem does not scale well for large system, we proposed an agent based distributed task migration approach for peak temperature reduction. Our agent based algorithm has good scalability as the number of processors increases. It achieves up to 18% power savings compare to a random mapping policy.

The rest of paper is organized as follows: Section II reviews the previous work. Section III introduces the many core system model, the system power and thermal model and cooling system model. We formulate the task allocation problem in section IV. In Section V we present the temperature aware distributed task migration framework. Experimental results are reported in Section VI. Finally, we conclude the paper in Section VII.

II. RELATED WORK

Various dynamic thermal management (DTM) techniques have been studied at different levels [8][6][11]. Most of these works rely on a widely used thermal modeling tool Hotspot [15] for fast thermal analysis. At micro-architecture level, DTM techniques such as clock gating, dynamic voltage and frequency scaling (DVFS), thread migration has been thoroughly explored [8].

At system level, different approaches have been taken to tackle the high processor operating temperature issues. Thermal aware task allocation and task migration has been studied in [6][12]. A multiple-input-multiple-output optimal control theory based power and thermal control algorithm has been proposed in [18]. The algorithm can control the power of the chip to a specific set point and maintain the chip temperature under a threshold. Recently, several proactive thermal management scheme has been proposed [7][19]. They utilized different temperature prediction model to

accurately estimate the future temperature in different scenarios and take actions in advance to prevent thermal emergencies.

At higher level, power and temperature management techniques for servers, for ensembles and even for data centers have been proposed in the previous works. In [11], a model for data center air conditioner cooling efficiency has been proposed. In [16], the heat transfer in data center has been studied thoroughly and a linear heat recirculation model has been introduced. Based on these works, an online workload allocation algorithm for data center has been proposed in [13]. They predict the future incoming requests and solve an integer linear programming problem online to allocate the workload and optimally turn on or turn off those servers in a data center.

Recently, the cooling fan power optimization has received noticeable attention. In [14], a joint fan power and processor leakage power consumption optimization has been considered. They formulate the problem as a convex optimization problem and solve it to obtain the optimum fan speed. In [3], fan cooling cost minimization for a multi-machine system has been studied. Their techniques intelligently adjust the workload at virtual machine level and CPU socket level and achieve large fan energy savings.

III. SYSTEM MODEL

A. Processor Model

In this paper, we consider a tile-based network-on-chip many-core architecture [17]. Each tile is a processor with dedicated memory and an embedded router. It will also be referred to as core or processing element (PE) in this paper. All the processors and routers are connected by an on-chip network where information is communicated via packet transmission. We refer to the cores that can reach to each other via one-hop communication as the *nearest neighbors*. Note that a pair of cores that are nearest neighbors sometimes does not have to be close to each other geometrically.

We assume the existence of a temperature sensor on each core. A temperature sensor can be a simple diode with reasonably fast and accurate response [8]. We also assume that a dedicated OS layer is running on each core that provides functions for scheduling, resource management as well as communication with other cores.

B. Processor Thermal Model

Due to the duality between heat transfer and RC circuits, we abstract the many-core system as an RC network. Let n denote the number of all thermal nodes in the system, including those in the heat sink layer and heat spread layer. Let N denote the number of processors in the system. The relation between n and N is determined by the equation $n = 4 \times N + 12$ [15]. Let TSS_i and P_i denote the steady state temperature and average power consumption of node i . P_i is 0 if node i belongs to the heat sink layer or heat spread layer because they does not consume any power. Let TSS and P denote vectors of TSS_i and P_i , $1 \leq i \leq n$. When the system reaches the steady state, for each thermal node, its temperature is a linear function of power consumptions P_1, P_2, \dots, P_n . The relation can be represented by the following equation

$$TSS = \mathbf{G}^{-1}P \quad (1)$$

where $\mathbf{G}^{-1} = [g_{ij}]$ is the inverse matrix of thermal conductance matrix \mathbf{G} . We simplify equation (1) by keeping only the thermal nodes related to the PEs:

$$\begin{pmatrix} T_1 \\ \vdots \\ T_N \end{pmatrix} = \begin{pmatrix} g_{11} & \cdots & g_{1N} \\ \vdots & \ddots & \vdots \\ g_{N1} & \cdots & g_{NN} \end{pmatrix} \begin{pmatrix} P_1 \\ \vdots \\ P_N \end{pmatrix} + \begin{pmatrix} D_1 \\ \vdots \\ D_N \end{pmatrix} \quad (2)$$

where N is the number of processors, and $D_i = \sum_{j=N+1}^n g_{ij} \cdot P_j$ is a set of constants, because the power consumption P_j of those thermal nodes related to the heat sink and heat spreader does not change. The coefficients g_{ij} and D_i , $1 \leq i, j \leq N$ can be obtained by offline analysis. Equation (2) shows that the steady state temperature of each PE is a linear function of average power consumptions on other PEs and increasing/decreasing the power consumption of one PE will have an impact on the steady state temperature of all other PEs.

C. Processor Cooling Model

In this paper, we assume one standard heat sink and fan cooling system for the entire many-core chip as the configuration of the TILER64 processor [1]. Our cooling system modeling follows the similar techniques described in the previous works [3] [14].

The heat generated in the die layer is transferred from the heat sink layer to the ambient environment and brought away by the cool air flow provided by the fan. The speed of the air flow determines how efficiently the heat can be dissipated and thus determines the temperature of the die. This heat dissipation efficiency is characterized by convective thermal resistance R_{conv} . The faster of the air flow speed, the more easily the heat can be dissipated, and the lower the convective resistance will be. Reference [3] pointed out that the convective resistance R_{conv} is proportional to $V^{-\alpha}$, where V is the air flow speed and α is a constant between 0.8 and 1.0. The air flow speed is determined by the fan speed, which in turn controls the fan power consumption. It has been pointed out in [14] that the fan power has cubic relation with the fan speed, i.e. $P_{fan} \propto V^3$. Finally, we obtained the relation between the fan power consumption P_{fan} and convective thermal resistance R_{conv} .

$$P_{fan} \propto R_{conv}^{-\frac{3}{\alpha}} \quad (3)$$

We next model the relation between the convective resistance and the die temperature. Although the Hotspot [15] provides a detailed and accurate thermal model at micro-architecture level, its complexity is too high to be used analytically. And it does not directly reveal the relation between convective resistance and the die temperature. Therefore, we adopted a simple yet accurate model as shown in Figure 1 [3]. In this model, P_i , C_i , and R_i are the power consumption, thermal conductance and die to package thermal resistance of processor i respectively. R_{conv} is the convective resistance.

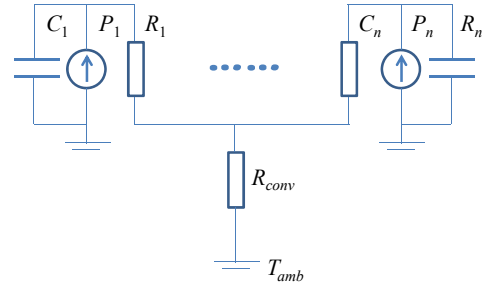


Figure 1. Simplified multiprocessor thermal model

Similar to the previous works [3][14] we are only interested in the temperature at steady state when the system reaches the equilibrium. This is because the time constant of heat sink is much larger than the time constant of the core. Therefore all the capacitors in the system are open circuit and only thermal

resistances will be considered. Then the die temperature T_i of core i can be computed as

$$T_i = P_i R_i + R_{conv} \sum_{i=1}^N P_i \quad (4)$$

where P_i is the power consumption of core i and R_i is the approximation thermal resistance from die to package. If the power consumption of core i does not change, the die temperature of core i is a linear function of R_{conv} . To verify the simple model, we run the simulation in Hotspot to obtain the die temperature of core by varying the convective thermal resistance. Figure 2 shows that the simulated core temperature and the core temperature predicted by the linear model matches very well.

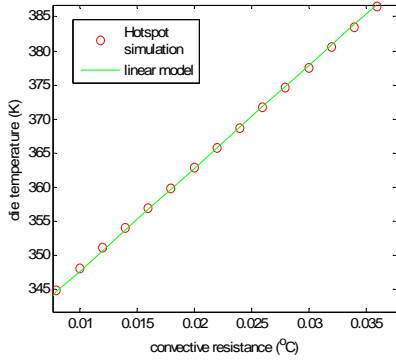


Figure 2 Linear approximation of relation between die temperature and convective resistance

D. Leakage power model

The leakage power consumption of a processor depends on the die temperature, supply voltage and a number of other factors. If the supply voltage is constant, the leakage power consumption can be expressed as follows:

$$P_{leak} = A_1 T_d^2 e^{\frac{A_2}{T_d}} + A_3 \quad (5)$$

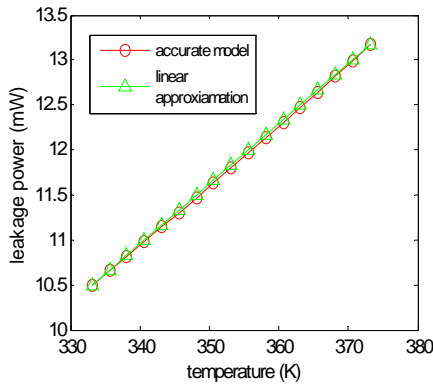


Figure 3 Linear approximation of leakage power model

Where A_1 , A_2 and A_3 are constants which are dependent on processing technology and supply voltage, T_d is the die temperature. It has been pointed out in [10] that the leakage power can be approximated using a linear model and the resulting error is expected to be less than 5% for a large temperature range from 20°C to 120°C. We approximate the leakage power using its first order Taylor expansion at 80°C and compare the linear approximation model with the original model in Figure 3. The green line is the linear approximation while the red line is the

original model given by equation (5). Figure 3 shows that the linear model has very small error compare to the original model in the normal operating range, which is between 60°C and 100°C.

Based on this observation, we approximate the leakage power of the i th core using a linear model $P_{leak,i} = aT_{d,i} + b$, where $T_{d,i}$ is the average die temperature of the core, a and b are two scalars. The total leakage power consumption can be simply calculated as $P_{leaktotal} = a \sum T_{d,i} + Nb = NaT_{d-avg} + Nb$, where $T_{d-avg} = \sum_{i=1}^N T_{d,i} / N$ is the average temperature of N cores. Thus the total chip leakage power can be approximated as a linear function of average die temperature. Because $T_{d,i}$ is linearly proportional to R_{conv} , the leakage power $P_{leaktotal}$ is also a linear function of the convective resistance.

IV. PROBLEM FORMULATION AND ANALYSIS

In this paper, we study the impact of task allocation on the overall system power. The overall power consumption is the sum of the CPU power consumption and the fan power consumption while the CPU power consumption consists of dynamic power and leakage power. Therefore the overall power consumption model can be written as follows.

$$P_{total} = P_{dyn} + P_{leak} + P_{fan} \quad (6)$$

In a homogenous multi-core system, task allocation has little impact on the dynamic power consumption because all cores are identical. However, because task allocation changes the temperature distribution across the system, it has the potential to change the leakage power and fan power which are temperature related power consumptions.

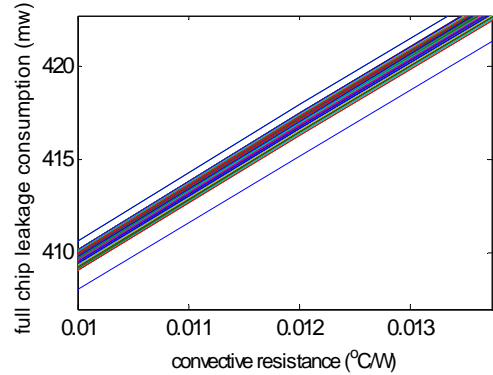


Figure 4 The relation between full chip leakage power consumption and different task allocations

To show the relation between task allocation and leakage power consumption, we randomly generated 100 groups of task allocation for a given workload and compare their leakage power consumption on a 36 core chip multiprocessor. The workload consists of 36 tasks whose power consumption varies from 10mW to 20mW (details about workload generation are described in section VI.) Figure 4 shows the leakage power for all 100 groups as the convective resistance increases. The leakage power consumption for the worst mapping and the best mapping differs only by less than 1% for a given convective resistance. This is intuitively correct. The leakage power is linearly proportional to the average die temperature which is determined by the average power density across the chip. Since the task allocation has little impact on the processor dynamic power consumption, which is still the dominant part of the CPU power consumption when it is actively running, it does not significantly change the average chip

temperature either. Consequently, the leakage power remains stable. Figure 5 (a) shows the average die temperature for those 100 different random mappings as the convective resistance increases. (The blue line that lies at the bottom corresponds to the task allocation scheme that is found by our multi-agent distributed task migration framework that will be introduced in the next section.) As we can see, the maximum difference in average die temperature is less than 1°C .

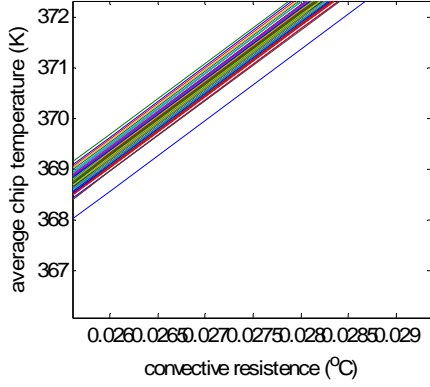


Figure 5 (a)

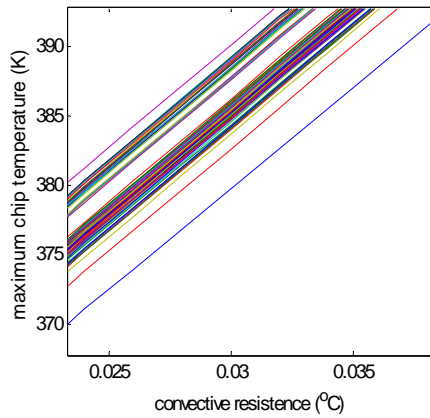


Figure 5 (b)

Figure 5 Comparison of average and maximum temperature of 100 different task allocations

Based on the experimental results we have two observations, (1) for a given R_{conv} , the leakage power can be considered to be independent to the task allocation, (2) when the workload is given, the only parameter that controls the leakage power is the fan speed which is reflected by R_{conv} . Their relation can be represented by a linear function: $P_{leak} = c_1 R_{conv} + c_2$.

On the other hand, different task allocation significantly affects the peak temperature. In order to bring the peak temperature below the constraint, the fan speed needs to be adjusted accordingly, which in turn leads to different R_{conv} . For example, Figure 5 (b) shows the maximum chip temperature of 100 different mappings as the convective resistance increases. (Again, the blue line that lies at the bottom corresponds to the task allocation that is found by our multi-agent distributed task migration framework.) We can see that the difference in peak temperature is more than 10°C . Note that because the average temperatures for different allocations are almost the same, the task allocation that gives the lowest peak temperature is the one that generates the most balanced

temperature distribution. A task allocation that generates highly unbalanced temperature distribution will force the cooling fan to work harder (and consumes more power) to keep the peak temperature under the constraint. However, as the speed of cooling fan increases, the average chip temperature will decrease and therefore bring down the leakage power. When searching for the optimal task mapping, we need to consider the tradeoff between fan power and leakage power.

Because P_{dyn} is independent to thermal convective resistance, P_{leak} is linearly proportional to convective resistance and P_{fan} is an inverse cubic function of the convective resistance, the overall power consumption is a convex function on the convective resistance. There will be an optimal convective resistance R_{conv}^* (corresponding to the optimal fan speed) which minimizes the overall system power. Furthermore, for a given workload, task allocation cannot change the relation between the overall power consumption and the convective resistance.

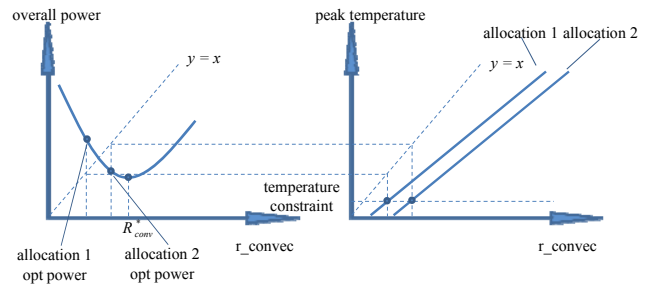


Figure 6 (a)

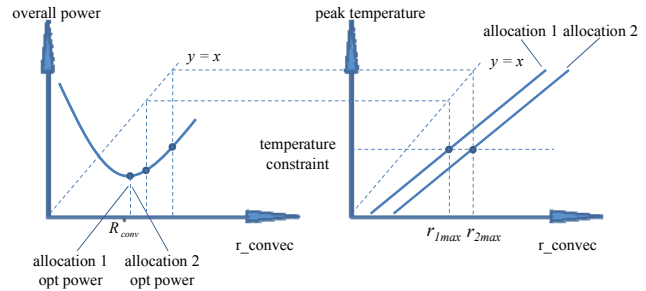


Figure 6 (b)

Figure 6 The overall power consumption depend on convective resistance

Figure 6 shows the overall power consumption and the peak temperature under different task allocations as functions of the convective resistance. Figure 6 (a) shows the scenario when the temperature constraint is strict and the convective resistance (i.e. r_{1max} and r_{2max}) that could bring the peak temperature to the constraint are located to the left of R_{conv}^* . In this case the overall power is dominated by the fan power. Increasing the fan speed can only increase the overall power consumption. The best task allocation that minimizes the overall system power is allocation 2 which minimizes the peak temperature. Figure 6 (b) shows the scenario when the temperature constraint is loose and the convective resistance that could bring the peak temperature to the constraint are located to the right of R_{conv}^* . In this case the leakage power dominates the overall power. If the fan speed is set to exactly satisfy the temperature constraint (i.e. the convective resistance is set to r_{1max} or r_{2max}) then the best allocation is scheme 1 which has higher peak temperature. We denote the maximum R_{conv} that keeps the peak temperature under constraint as r_{max} . Obviously, any convective resistance that is less than r_{max} can keep the system in safe temperature zone. This includes R_{conv}^* .

For both allocation 1 and 2, setting the convective resistance to R_{conv}^* can minimize the overall power while satisfying the temperature constraint. Because with a loose temperature constraint, the fan power does not dominate the overall power alone, leakage power plays an important role as well. Increase the fan speed would increase the fan power but could reduce the temperature and leakage power. In this case, power consumption is not sensitive to task allocation. Any allocation scheme whose r_{max} is greater than R_{conv}^* could be used to find the optimal tradeoff point between the fan power and the leakage power. Obviously, among all possible task allocations, the allocation that minimizes the peak temperature is most likely to satisfy this property.

Based on these observations, we concluded that, to optimize the overall power consumption, there are two steps. First is to find the task allocation that minimizes the peak temperature. Second is to adjust the fan speed to find the optimal tradeoff point between fan power and leakage power such that the overall power consumption is minimized and the temperature constraint is satisfied. The latter step could be achieved by using feedback control while former step will be discussed in detail in the following sections.

V. POWER OPTIMAL TASK ALLOCATION

Based on the analysis in the previous section, we will focus on searching for the optimal task allocation that minimizes the peak temperature among all cores.

In the following subsections, we first present an exact formulation of this problem which is a zero-one min-max problem. Because the exact formulation has extremely high complexity, we then propose a multi-agent task migration framework that searches for the best task allocation during runtime.

A. An exact formulation

Given a floorplan of a multi-processor system with n cores integrated on a chip, we assume that the thermal conductance matrix can be characterized by offline training. We further assume that the given workload consist of n different tasks $\{\tau_1, \tau_2, \dots, \tau_n\}$ whose power consumption $\{P_1, P_2, \dots, P_n\}$ can be obtained through offline training or online estimation by observing the event counter. We assume the power consumption is a constant for each task, because we are only concerned about the steady state temperature. Here we assume that the core does not support multitasking and the number of tasks is equal to the number of cores. If the number of tasks is less than the number of cores, we can simply add some dummy tasks with θ -zero power consumption.

Our goal is to obtain a mapping between the n tasks and the processors such that the resulting maximum temperature among all the cores is minimized. For each task k and processor j , there is a variable x_{jk} . Variable x_{jk} is 1 when task k is mapped to processor j , otherwise it is 0. We formulate the problem as a zero-one min-max linear programming as follows:

$$\min_i \max_j \left(\sum_{j=1}^n \sum_{k=1}^n g_{ij} x_{jk} P_k \right) + D_i \quad (7)$$

Subject to:

$$\sum_{j=1}^n x_{jk} = 1, \forall k = 1, \dots, n \quad (8)$$

$$\sum_{k=1}^n x_{jk} = 1, \forall j = 1, \dots, n \quad (9)$$

$$x_{jk} \in \{0, 1\} \quad (10)$$

Constraint (8) guarantees that a processor is only occupied by one task and constraint (9) ensures that a task can only be mapped to one processor. The item within the min-max operator in the objective function is the temperature of the i th core. To see this, we rewrite the equation (2) as follows:

$$\begin{pmatrix} T_1 \\ \vdots \\ T_N \end{pmatrix} = \begin{pmatrix} g_{11} & \dots & g_{1N} \\ \vdots & \ddots & \vdots \\ g_{N1} & \dots & g_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{NN} \end{pmatrix} \begin{pmatrix} P_1 \\ \vdots \\ P_N \end{pmatrix} + \begin{pmatrix} D_1 \\ \vdots \\ D_N \end{pmatrix} \quad (11)$$

where $X = [x_{ij}]$ is a permutation matrix which assigns the n tasks to the processors. Expand the right hand side the equation would give the equation $T_i = a_{ij} x_{jk} P_k$. Then the objective function $\min\text{-max}(T_i)$ is to minimize the maximum temperature among all n processors.

By some simple transformation, the min-max problem can be converted to traditional linear programming:

$$\min u \quad (12)$$

Subject to:

$$\left(\sum_{j=1}^n \sum_{k=1}^n g_{ij} x_{jk} P_k \right) + D_i \leq u, \forall i = 1, \dots, n \quad (13)$$

$$\sum_{j=1}^n x_{jk} = 1, \forall k = 1, \dots, n \quad (14)$$

$$\sum_{k=1}^n x_{jk} = 1, \forall j = 1, \dots, n \quad (15)$$

$$x_{jk} \in \{0, 1\} \quad (16)$$

The above zero-one min-max linear programming is an exact formulation of the task allocation problem. However, it is extremely difficult to solve. For example, for a problem with 36 cores there will be 1296 binary variables, it would take more than two days to solve this problem using the open source linear programming solver `lp_solver` [2] on a 3.2GHz Quad core Xeon processor. Therefore, it is infeasible to use the solution online for power and thermal optimization in the future many-core platform as the core counter could go up to hundreds and thousands [4]. Instead of solving this min-max problem directly, we propose the following online heuristic.

B. Distributed task migration

In this section, we presented our distributed task migration framework that searches the optimal task allocation during runtime.

We denote our multi-agent task migration algorithm as MATM. The framework has a low cost agent residing in each core. It is implemented as part of the OS based resource management program which performs thermal-aware task migration. The agent observes the workload and temperature of local processor while communicating and exchanging tasks with its nearest neighbors. The agent based distributed framework has better scalability compared to the centralized method as the communication cost and migration overhead for each core does not increase when the number of cores in the system increases.

The proposed MATM adopts a task exchange based migration scheme. By exchanging tasks, the processors can maintain a balanced temperature distribution and hence reduces the peak temperature.

Communication protocol

Each core running a MATM agent can be in two phases: execution phase and communication phase. These two phases are interleaved. During the execution phase the core executes the

current computing task while during the communication phase it initiates task migration request to its nearest neighbor or respond to the task migration request from its nearest neighbor. The communication phase can further be divided into four sub-phases: broadcasting self workload to neighboring cores, receiving workload information from neighbors, sending migration requests to neighbors, exchanging tasks with neighbors. Figure 7 shows the diagram of the communication protocol. We assume that a MPI (Message Passing Interface) based communication is adopted. Therefore two cores do not have to enter the communication phase synchronously in order to communicate to each other. We also refer the communication phase as scheduling interval.

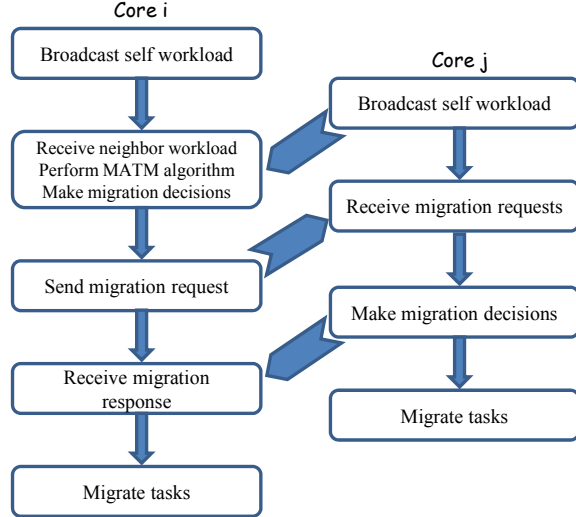


Figure 7 Diagram of communication protocol

At the beginning of each scheduling interval, an agent on a processor would broadcast its own workload to neighbors and request them sending back their workload. Because the scheduling intervals in all processors are not synchronized, the request is not likely to be checked and responded by neighbor agents right away. On the other hand, because all processors adopt the same execution and scheduling interval, it is guaranteed that all neighboring agents will respond before the next scheduling interval after the request is issued.

After receiving the response of neighbor workloads, the agent performs the MATM algorithm to decide whether to exchange task with neighbors and select which neighbor to exchange task with. Then it will send a migration request to the selected processor. For all other neighboring processors, the agent will also send an acknowledgement to them which indicates no task exchange. After that, the agent waits for the migration response from the selected processor. Note that this communication protocol will only be performed in a neighborhood when there are workload changes in a core. Therefore the communication overhead is relatively low.

MATM distributed migration algorithm

The FDTM algorithm distributes the tasks among processors based on their heat dissipation. It moves high power tasks to processors with strong heat dissipation capability and moves low power tasks to processors with weak heat dissipation capability. By distributing tasks in this way, local hotspots can be mitigated and thus peak temperature of the chip can be reduced.

To determine if an exchange of tasks between two processors is beneficial to the whole system, we consider equation (2) again.

Assume that PE_i and PE_j exchange tasks, and their average power consumptions are altered by ΔP_i and ΔP_j respectively. Using equation (2), the total die temperature change of all processors in the system after task migration can be calculated as:

$$\sum_{k=1}^N \Delta T_k = G_i \cdot \Delta P_i + G_j \cdot \Delta P_j \quad (17)$$

where G_i (or G_j) is a parameter that characterizes the heat dissipation ability of processor i (or j), $G_i = \sum_{m=1}^N g_{mi}$, $G_j = \sum_{n=1}^N g_{nj}$. The temperature contributed by processor i running task k can be calculated as $G_i P_k$. If $G_i < G_j$, then the temperature contribution made by processor i will be less than the contribution made by processor j when running the same task, which means processor i can dissipate heat better. Thus running high power task on processor i have smaller chance to produce high peak temperature than running high power task on processor j . Therefore, if $G_i < G_j$ and $P_i < P_j$, it is reasonable to switch the tasks on the two processors. This leads to $\sum_{k=1}^N \Delta T_k < 0$ in (18). In conclusion, if a task exchange between two neighbor processors leads to $\sum_{k=1}^N \Delta T_k < 0$, then this task exchange is beneficial for the system and the task exchange should be carried out.

If an agent found that it is beneficial to exchange task with several neighbor agent, the agent will select a neighbor that lead to maximum temperature reduction, i.e. the minimum $\sum_{k=1}^N \Delta T_k$ (because it is negative), and send migration request to the selected neighbor. If an agent received several migration requests from neighbors, it will follow the same criterion to select a neighbor to exchange tasks. We summarized the MATM in Figure 8.

Algorithm 1 MATM

1. **for** each neighbor processor j , compute
2. $\Delta T_{ij} = G_i \cdot \Delta P_i + G_j \cdot \Delta P_j$
3. $\Delta T_{min} = \min(\Delta T_{ij})$
4. Select processor j , and send migration request to it

Figure 8 MATM algorithm

VI. EXPERIMENTAL RESULTS

We implemented a multicore system simulator using C++. Hotspot [15] is integrated to the simulator to analyze the system thermal behavior. Though the model is scalable for any number of cores, a 36 core system with 6x6 grids is chosen for our experiments due to the limitation of simulation time. Each core has a size of 4mm x 4mm with silicon layer of 24mm x 24mm.

We carried out experiments using power sequences collected from real applications. We used 9 different CPU benchmarks comprising of 3 SPEC 2000 benchmarks (bzip2, applu and mesa), 4 Mediabench applications (mpeg2enc, mpeg2dec, jpegdec, jpegenc) and 2 telecom applications (crc32 and fft) from MiBench benchmark suite. We collected cycle level power trace by modifying the Watch power analysis tool [5]. The average dynamic power consumptions and steady state temperatures of each task are summarized in Table I. The workloads of the following experiments are random combinations of multiple copies of these 9 benchmarks. All experiment results reported below are the average of 10 runs.

Table I. Average Power and Steady State Temperature of CPU Benchmarks

Bench marks	crc32	mp2 enc	mp2 dec	fft	applu	mesa	bzip2	jpeg dec	jpeg enc
Avg. Power (mW)	24.4	19.4	19	18.5	17.4	17.3	13.3	10.7	10.4

Steady Temp. (°C)	99.42	84.17	82.95	81.42	78.07	77.76	65.56	57.63	56.72
-------------------	-------	-------	-------	-------	-------	-------	-------	-------	-------

The experiment is performed on 5 different task sets. Each task set consists of 36 tasks. Each task is random selected from the 9 benchmarks listed in Table I. We control the selection probability of a benchmark based on its average power consumption so that the average power consumption of the 36 tasks can follow a desired distribution. Uniform distribution evenly generates tasks with different average power consumptions. Triangular (cool) distribution generates more low power tasks than high power tasks, whereas triangular (hot) distribution generates more high power tasks. Normal distribution generates a set of tasks whose power consumption is mostly clustered around the medium power. On the other hand, inverse normal distribution generates more high power tasks and low power tasks than the medium power tasks.

A. Fan power savings

Figure 9 compares the r_{max} (i.e. the maximum thermal convective resistance that is required to exactly meet the temperature constraint) between the random allocation and MATM based allocation with the temperature constraint setting to 85°C. The results show that, to maintain the whole system under the temperature constraint, the minimum fan speed required by MATM based allocation is 14.5% less than that is required by the random task allocation. The reduced fan speed could bring cubic savings in fan power for the system. And Table II shows the fan power savings of our proposed MATM policy compared to the random allocation policy. The MATM can achieve an average of 37.2% fan power savings over random allocation while maintains the maximum chip temperature under the thermal constraint.

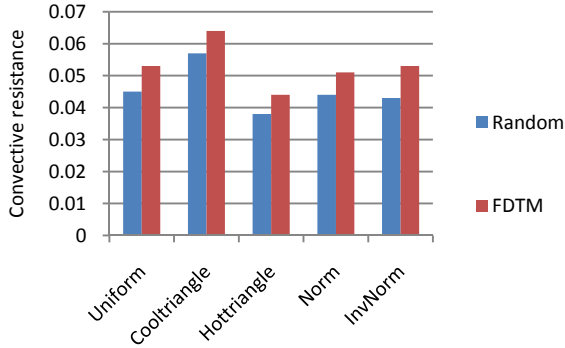


Figure 9. Convective resistance comparison between Random allocation and FDTM allocation

Table II. Fan power savings of FDTM compare to the random task allocation

Workload	Uniform	Cool triangle	Hot triangle	Norm	Inv Norm
Fan power savings	38.79%	29.35%	35.58%	35.78%	46.60%

The reason that the MATM policy can achieve power savings is that through agent negotiation and task migration, tasks can be distributed among processors evenly according to a processor's heat dissipation ability, i.e. high power tasks are moved to cores with stronger heat dissipation ability while lower power tasks are moved to cores with weaker heat dissipation. Therefore the processors' temperatures are distributed more evenly across the chip and the maximum temperature is reduced. The fan can run at

a relatively lower speed to guarantee the temperature constraint. Therefore the fan power savings is achieved.

B. Overall system power consumption

In the second experiment, we examine the effect of temperature constraint and task allocation on the overall system power consumption, i.e. the power consumption summation of dynamic power, leakage power and fan power. We select the uniform workload distribution in this experiment. We vary the temperature constraint for 80°C, 85°C and 90°C and compare the power consumption between MATM based task allocation and random allocation. For both systems, optimal tradeoff point between fan power and leakage power will be searched after the system reaches stable state. As shown in Table III, MATM based allocation policy could achieve 17.9% overall power savings when the temperature constraint is 80°C. When the temperature constraint increases to 85°C and 90°C, the power saving reduces to 5.1% and 1.2% respectively.

The experimental results show a diminishing power savings as the constraint temperature increases from the Table III, and task allocation gives large power savings especially when temperature constraint is strict. To understand this, we draw the overall power consumption and fan power consumption against the convective resistance curve in Figure 10. When temperature constraint is strict, the convective resistance has to be small to satisfy the constraint. When fan working in this area, the curve slope is sharp and a little decrease in convective resistance would increase the fan power as well as the overall system power significantly; therefore a better task allocation which reduces maximum chip temperature can achieve large power savings. On the other hand, when temperature constraint is loose, the convective resistance does not have to be small to satisfy the constraint. In this case, the curve slope is flat and the difference in convective resistance does not affect the fan power and overall power consumption significantly. Therefore, different task allocation achieves similar overall system power consumptions. If we further relax the temperature constraint so that the r_{max} of both random and MATM allocations are located to the right side of R_{conv}^* , the MATM allocation will not give any power saving over the random allocation as both of them can work at the optimal tradeoff point.

Table III. Overall system power consumption comparison under different temperature constraints

Overall System Power Consumption (mW)			
Temp. Constraint	80°C	85°C	90°C
Random Mapping	1268.9	1110.8	1067.3
FDTM	1076.6	1057	1055.1

Figure 11 shows each component in the overall system power consumption. The fan power consumption plays an important part in the random allocation when temperature constraint is strict. It accounts for 21.1% of total consumption. When the constraint is relaxed, the share of fan power decreases. The MATM allocation reduces maximum chip temperature and the fan can be maintained in a low speed. Therefore the fan power consumption is small, less than 6% for all temperature constraint. We also notice that the dynamic power stays the same for all constraint while the leakage power increase as the constraint is relaxed. This is because allowing higher maximum chip temperature will also increase the average chip temperature, therefore the leakage power increases. We also notice the MATM based task allocation has higher

leakage power consumption compare to the random allocation. This is because in order to maintain the same maximum chip temperature, the higher fan speed needed for random allocation makes its average temperature lower and hence it consumes less leakage power. However, after combining the fan power, the MATM based allocation still has lower total power consumption.

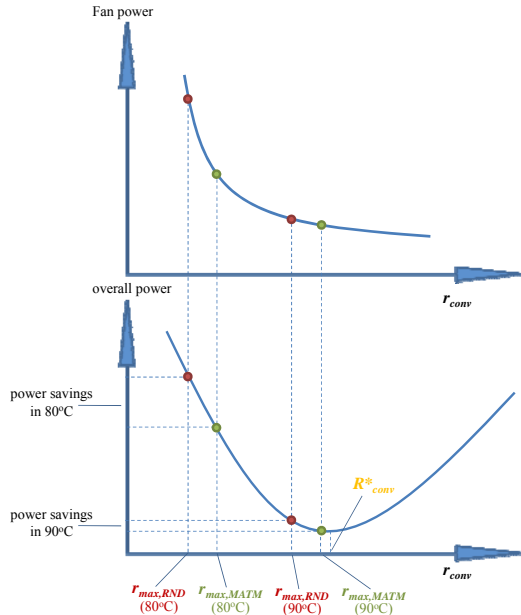


Figure 10. Power consumption against convective thermal resistance curve

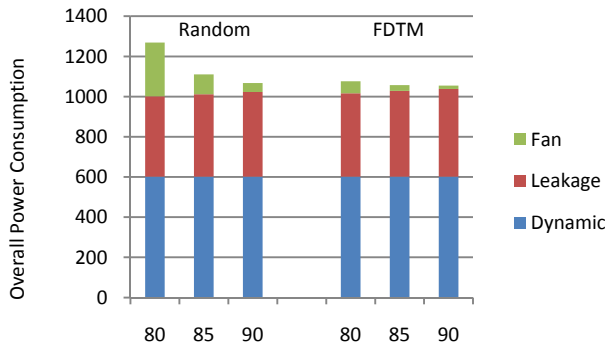


Figure 11. The overall power consumption break down

VII. CONCLUSION

In this paper, we studied the impact of task mapping on the overall power consumption of a homogenous multi-core system. We formulated the task mapping problem as a zero-one linear programming problem and proposed an agent based distributed task migration approach to solve this problem. Our agent based algorithm has good scalability as the number of processors increases. Experimental results show that our policy achieves large power savings compare to a random mapping policy.

REFERENCES

- [1] *Tile Processor Architecture: Technology Brief*. [Online]. Available: http://www.tilera.com/pdf/ProductBrief_TileArchitecture_Web_v4.pdf
- [2] <http://lpsolve.sourceforge.net/5.5/>
- [3] R. Ayoub, S. Sharifi, T. Rosing, "GentleCool: Cooling Aware Proactive Workload Scheduling in Multi-Machine Systems," *In Proc. of Design Automation and Test in Europe*, pages 295-298, Mar. 2010.
- [4] S. Borkar, "Thousand Core Chips – A Technology Perspective," *In Proc. of Design Automation Conference*, pages 746 – 749, Jun. 2007.
- [5] D. Brooks, V. Tiwari and M. Martonosi, "Wattch: A Framework for Architectural Level Power Analysis and Optimizations," *In Proc. Int. Symp. Computer Architecture*, pages 83-94, June 2000.
- [6] A. Coskun, T. Rosing, K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Proc. of Design Automation and Test in Europe*, pages 1659-1664, Apr. 2007.
- [7] A. Coskun, T. Rosing and K. Gross, "Utilizing predictors for efficient thermal management in multiprocessor SoCs," *In IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pages 1503-1516, Oct. 2009.
- [8] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration," *In Proc. Int. Symp. Computer Architecture*, pages 78-88, Jun. 2006.
- [9] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler and T. Keller, "Energy Management for Commercial Servers", *IEEE Computer*, Vol. 36, Issue 12, pages 39-48, Dec. 2003.
- [10] Y. Liu, R. Dick, L. Shang and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *Proc. of Design Automation and Test in Europe*, pages 1526-1531, Apr. 2007.
- [11] J. Moorey, J. Chasey, P. Ranganathan and R. Sharma, "Making Scheduling Cool: Temperature-Aware Workload Placement in Data Centers," in *Proc. of the annual conference on USENIX Annual Technical Conference*, pages 5-18, Apr. 2005.
- [12] F. Mulas, M. Pittau, M. Buttu, S. Carta, A. Acquaviva, L. Benini, and D. Atienza "Thermal Balancing Policy for Streaming Computing on Multiprocessor Architectures," *In Proc. of Design Automation and Test in Europe*, pages 734-739, Mar. 2008.
- [13] E. Pakbaznia, M. Ghasemazar, and M. Pedram, "Temperature-Aware Dynamic Resource Provisioning in a Power-Optimized Datacenter," *In Proc. of Design Automation and Test in Europe*, pages 124-129, Mar. 2010.
- [14] D. Shin, N. Chang, J. Choi, S. Chung and E. Chung, "Energy-Optimal Thermal Management for Green Computing," *In Proc. Int. Conf. on Computer-Aided Design*, pages 652-657, Nov. 2009.
- [15] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy and D. Tarjan, "Temperature-Aware Microarchitecture: Modeling and Implementation," *ACM Trans. on Architecture and Code Optimization*, Vol. 1 Issue 1, pages 94-125, Mar. 2004.
- [16] Q. Tang, S. Gupta and G. Varsamopoulos, "Energy-Efficient, Thermal-Aware Task Scheduling for Homogeneous, High Performance Computing Data Centers: A Cyber-Physical Approach," in *IEEE Trans. Parallel and Distributed Syst.*, vol. 19, issue 11, pages 1458-1472, Nov. 2008.
- [17] S. Vangal, J. Howard, G. Ruhl, S. Dige, H. Wilson, J. Tschanz, D. Finan, P. Lyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote and N. Borkar "An 80-Tile 1.28 TFLOPS Network-on-Chip in 65nm CMOS," *In Proc. Int. Solid-State Circuits Conf.*, pages 98-589, Feb. 2007
- [18] Y. Wang, K. Ma and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," *In Proc. Int. Symp. Computer Architecture*, pages 314-324, Jun. 2009.
- [19] Y. Ge, P. Malani, Q. Qiu, "Distributed Task Migration for Thermal Management in Many-core Systems," *In Proc. of Design Automation Conference*, Jun. 2010.