An Energy-Aware Multi-Core Scheduler based on Generalized Tit-For-Tat Cooperative Game

Guowei Wu
School of Software, Dalian University of Technology, Dalian, China
E-mail: wgwdut@dlut.edu.cn

Zichuan Xu, Qiufen Xia, Jiankang Ren
School of Software, Dalian University of Technology, Dalian, China
E-mail: {eggerxu, xiaqiufen, rjk.dlut}@gmail.com

***Abstract*—Energy-constrained computing environments are emerging those years, especially in embedding computing. A game theoretic energy-aware scheduling algorithm for multi-core systems is proposed in this paper, namely, GTFTES (Generalized Tit-For-Tat Energy-aware Scheduling). GTFTES is designed to work in a resource-rich environment where resources always compete for tasks. A generalized Tit-for-Tat based method, where whether a core will cooperate or not is decided by a hardness factor, is considered in this paper. The algorithm is implemented in our EASS simulator. Simulations results show that the proposed game can reduce the temperature difference between different groups of cores which effectively avoids the local hotspot of a processor.**

***Index Terms*—energy-aware scheduling, multi-core, game theory, generalized tit-for-tat**

I. INTRODUCTION

Resources in embedded systems such as power and computation capacity are often limited in both industrial and mobile applications. It is deserved attention that there exists an increasing trend in the use of multi-core processor in energy-constrained embedded and mobile systems, where real-time guarantee must be met without exceeding safe temperature levels within the processor. However, multi-cores increase power consumption which in turn creates temperature problems and non uniform power density map. As high-performance embedded systems become increasingly energy-constrained, the question of how to balance the energy distribution against real-time guarantee must be addressed.

Dynamic Thermal Management (DTM) techniques are the primary solutions to lower energy consumption in hardware level. Clock gating and Dynamic Voltage Frequency Scaling are two reactive DTM mechanisms. When the core power rises to a critical level, clock gating technique switches off the core clock for a certain period of time and then switches it on. During this time the core is slowing down the total processing time and in turn decreases power consumption. Dynamic Voltage Frequency Scaling (DVFS) decreases the frequency and voltage of processors at run time to achieve lower power consumption. Hardware level thermal management can respond rapidly when the processor is overheated. However, it has no information on the behavior of the applications when the processor is running, i.e., hardware level thermal management always responds in the same way independent from the workload, moreover, hardware level thermal management mechanism increases chip cost.

Recently, software level energy management technique has been studied because of its low cost. The primary advantage of software technique is that it can obtain the information of running threads; hence it can reduce the processor energy or temperature according to the characteristics of the thread. One primary approach adopted in operating system level for energy management is energy aware scheduling, which takes processor power into account to avoid thermal violation, and keeps the temperature below a certain limitation by throttling the "hot" tasks. Authors in [1-3] present algorithms for thread migration. However, they do not take real-time characteristics into consideration. Paper [4] proposes a thread scheduling scheme for a class-based 2D Mesh interconnected multi-core system which considers inter-core distances in its scheduling decisions. Without considering the temperature property, their algorithm is hard to fulfill the thermal constraints. Authors [5] only analyze the relationship between activity migration and power density. Some thermal-aware scheduling algorithms are only suitable for CMP architecture [5-7]. Algorithms in [8] aims to minimize the energy consumption and the makespan of complex computationally intensive scientific problems, subject to energy constraints. Moreover, authors in [6] also did some research the problem of power-aware scheduling/mapping of tasks by applying game theory in [9]. However, their algorithms only consider energies

Corresponding author: Zichuan Xu
Email:eggerxu@gmail.com

doi:10.4304/jcp.7.1.106-115

rather than temperatures. In contrast, we propose a more detailed and energy-aware scheduling environments where players (cores) are competing for the tasks, and from a more realistic perspective, temperatures are used to evaluate the algorithm's performance.

The problem we consider falls under the class of resource allocation problems. Although these problems have been exploded theoretically and practically in areas such as gird computing, scheduling and wireless network [10-12], it's still a novel approach to apply market models into the constraint-aware scheduling, such as power-aware, temperature-aware scheduling. In the most common form of auctions, the highest bidder wins the resource and pays as much as the bid. However, the users may have an incentive to lie about their true value of the resource. A Vickrey auction [13] is a type of sealed-bid auction, where bidders submit written bids without knowing the bid of the other people in the auction. The highest bidder wins, but the price paid is the second-highest bid. This type of auction gives bidders an incentive to bid their true value.

Above lists common game theoretic methods which works in a non-corporative and Tit-for-Tat manner. Yet, Tit-for-Tat, as traditionally defined, makes sense only for a two-player game. For a many core processor, the cores in it are always highly interacted especially in many core systems. For example, if one core is thermally saturated, and its neighbor cores are often influenced for the heat transferring. Thus, it is reasonable to model the scheduling by utilizing a corporative game model. In this paper, we adopt the generalized Tit-for-Tat mechanism to model the scheduling problem. The objective is to minimize power density of the processor and temperature of each core.

This paper is an extension in terms of both theoretic and simulation results of our conference paper [14]. The main contributions of this paper are as follows.

- A generalized tit-for-tat based corporative scheduling game is proposed in the energy-aware scheduling algorithm.
- A simple but feasible and efficient temperature calculation method is presented.
- Extensive simulations on our energy-aware scheduling simulator, namely EASS, are given.

The rest of the paper is organized as follows. In section II, we present some notations and the problem definition. In section III, the GTFTES algorithm and temperature calculation algorithm is stated in this section. In section IV, the GTFTES algorithm is implemented in EASS and compared with other related algorithms. The experimental results and performance analysis is listed in this section. In section V, we summarize related work. We conclude the paper in Section VI.

II. PRELIMINARIES

In this section, we give some notations and the definition of the problem. We introduce the scheduling environment considered by this paper and the concepts of game theory.

A. A Scheduling Environment

This paper adopts a resource-rich scheduling environment in which numerous execution cores complete the tasks. The reason that we adopt such a circumstance is that, currently, the execution cores of commercial CPU, especially GPU are soaring in an astonishing rate. It is common to see hundreds of cores in a GPU. However, since not all users' daily tasks can be related to GPU, there is circumstances that majority of the execution units of GPU are idle, which also means number of tasks is less than the cores .

A multi-core scheduling environment (shown in Figure 1) in which cores are competing for tasks is proposed in this section. In such environment each core wants to maximize its income through executing tasks. Tit-for-Tat is a highly effective strategy in game theory for the iterated prisoner's dilemma. In this paper, we consider a generalized tit-for-tat method where each player decide to cooperate or retaliate according to the hardness factor h. Specifically, after each round of the game, each player will calculate the proportion that the players who cooperate (this proportion is specified as hardness h). If h is over a predefined value, the player will decide to cooperate in the next game round. Otherwise, he will choose to retaliate.

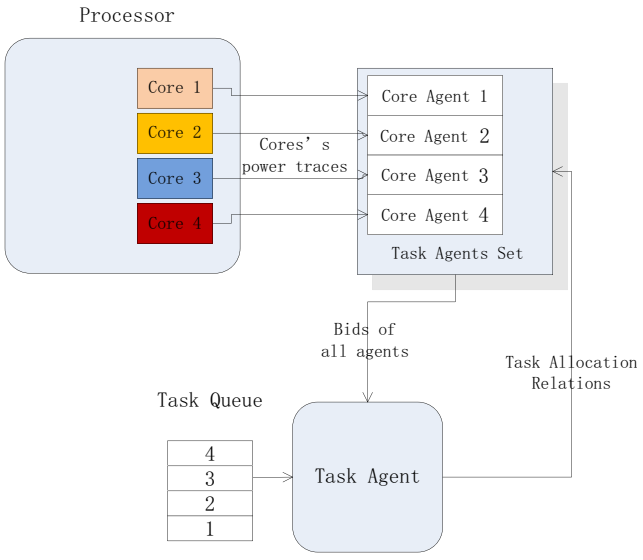


Figure 1. Scheduling Environment of GTFETS

Auctioning is a common way of allocating tasks to cores. In an auction each core bids a certain amount to buy a task. As the cores are selling its services, we use a reverse second price auction.

We consider a more general and realistic scenario where cores may have potential to cooperate to avoid the hotspot and reduce the energy consumption of a processor. And also, they are rational which means they will choose to retaliate

without considering the power status of the processor. Consider a energy-constrained embedded system that consists of a set of tasks, $T = \{\tau_1, \tau_2, \ldots, \tau_i\}$ $(0 < i < N)$ and a set of execution cores $EC = \{ec_1, ec_2, \ldots, ec_j\}$.

To state and analysis the algorithm, we need some assumptions which are stated next.

A.1 No task is starved of wealth.

A.2 The scheduling environment is resource-rich. As in market, this is a model of supply exceeds demand. Thus, with this assumption, the cores compete for the tasks sounds reasonable.

Then, the symbols used in this paper are listed in Table I.

TABLE I.
SYMBOLS AND NOTATIONS

Symbol	Notation
ec_i	The execution core on which the current thread is running
w_i	The overall wealth that a core bids at each time slice.
b_i	The core ec_i 's bid
c_j	The jth execution core where $0 < j \le M$
C_c	Currently available cores.
T_c	Temperature set of the currently available cores $T_c = \{t_j\}$
p_j	The price that core ec_j has to pay for the winning of an auction
g	The biding function of each task.
h_{th}^i	The core's hardness threshold which is actually a percentage of plays who choose to cooperate in a new round of game.
S_{th}	The temperature threshold of a execution core.
T	The set of tasks that are ready to be executed.
γ	The weighting coefficient that a agent to evaluate the value of a core
L_{avg}	The average execution time of a task on a core
φ	The weighting coefficient on L_{avg} when task agents are calculating the bids

B. Game Theory and Generalized Tit-for-Tat

Game theory is a multi-player decision theory. The players of a Game are subjects that make decisions. The players participate in a Game in order to get maximum benefit by selecting a reasonable action. Thus the elements of a Game include players, information, strategy spaces and payoff functions. In our scheduling environment, the competition of task j is a Game among N cores. Game theory involves three basic elements, which is defined as follows: Player set N : The set of all execution cores. $EC = \{1, 2, \ldots n\}$. Strategic set S_i : For each core $i \in N$ owns a strategic set, which comprises several strategies $S_i = \{s_i\} = \{s_1^{(i)}, s_2^{(i)}, \ldots, s_n^{(i)}\}, i = 1, 2, \ldots, n$. After each core determines its strategy, then $s = \{s_1, s_2, \ldots, s_n\}$ expresses action of the N cores at one time. We utilize s_i to represent the ith core's strategy, whereas s_{-i} is the strategy set of all the other cores. Payoff function $u_i : S \to R$, representing the benefits of cores in different strategies. Hence each cores wishes to seek for the maximal profit; therefore it intends to enlarge the benefit of its payoff function.

Generally, in game theory, effectiveness of a strategy is measured under the assumption that each player cares only about him or herself. By this game-theory definition of effectiveness tit for tat was superior to a variety of alternative strategies. However, in reality, tit-for-tat is only applicable to two player game where non-corporative actions are common. Generalized tit-for-tat is an extended version of tit-for-tat which could be used in the multi-player game. In this mechanism, in each auction, players can decide to cooperate or retaliate according to their observations.

III. GAME THEORETIC TEMPERATURE-AWARE SCHEDULING ALGORITHM

In this section, a game theoretic thread allocation in the energy-constrained systems and a temperature calculation method which is used in the simulator to evaluate the effectiveness of the GTFETS algorithm are proposed. Moreover, the theoretic results of the task allocation game are analyzed.

A. Problem Definition

We consider a scheduling environment with an energy-limited processor executing tasks whose number is always lower than cores (as assumption **A.2** shows). Thus, a scenario of cores competing for different tasks is considered in this paper.

In reality, the cores favor to get a task to execute for the sake of getting as much personal profit as possible. Actually, they only consider their own payoffs rather than the power status of a processor. However, for multi-player games in economic environments, there do exist some corporations where a group of players have a social goal. The 'HotSpot' of a processor may cause instability of the processor and even hardware damage [15], which is a reason that the processor performs to achieve a more uniform thermal status of itself for the sake of longer servicing time. In our previous work [16], we proposed temperature-aware scheduling algorithm which the temperature is calculated in a pre-set interval. Though, by adopting this method, the hotspots of processors can be efficiently eliminated, it is more expensive to do so than just using the power which can be calculated from the voltage. Thus, we consider energy factor as the main social goal of cores, and treat the temperature as a measurement of the algorithm.

One reverse VCG auction is actually one type of the sealed bid auction where the resource could be divided to be allocated to the tasks. In our scheduling scenario, each core ec_i submits a price that a task should pay for the executing service. Naturally, the core biding the lowest bids will win the auction. The auctioneer (the scheduler proposed in this paper) firstly decides the winning cores, and then assigns the tasks to the cores. As what it does in VCG, the auctioneer charges each core the harm they cause to other cores, and ensures that the optimal strategy for a task is to bid the value of the cores.

Formally, this N-player game can be written as $G=\left[N,\{S_i\},\{u_i(\cdot)\}\right]$ which specifies for each player i a set of strategies, or bidding functions, S_i (with $s_i \in S_i$) and a payoff function $u_i(s_1,\cdots,s_n)$ giving the computational capability associated with outcome of the auction.

B. Task Allocation Algorithm

As described in section 3.1, we consider the auction happens at the beginning at each time slice. However, in practical applications, not all the cores are willing to choose to cooperate in an auction for the sake of social goal. Also, the core affinity of each task is an important issue that a task allocation algorithm should consider. In the GTFTES algorithm, the task preferentially choose its previous execution if it is in the winners of each auction. The task agent's algorithm is listed in Table II.

TABLE II.
TASK AGENT'S ALGORITHM

Algorithm Name	Task's Algorithm
Input:	Bids set $B_i=\{b_i\}$, ready set of tasks T
Output:	The task allocation relation
1.	Rank the bids in non-increasing order. The winners are the top $\lvert T\rvert$ cores.
2.	Calculate the winners' payment. A winner's payment is the least bid of the winner set without its participation.
3.	If the previous execution core of task is in the winner set and it is currently not ocupied. Allocate itself to its previous core.
4.	Else, allocate the tasks whose previous execution core is not in the set of winners according to the tasks contribution. Specifically, the tasks with lower contribution will be allocated to a core with lower bid.

TABLE III.
CORE AGENT'S ALGORITHM IN ROUND K

Algorithm Name	Cores' Algorithm
Input:	The core's hardness threshold h_{th}^i and temperature threshold S_{th}, the set of tasks T
Output:	The core's bid vector b_i.
1.	Carry out the valuation of itself and calculate its thermal status S_i.
2.	if $S_i > S_{th}$, the core is forced to cooperate in this round of auction. And, calculate its bid by its thermal status, that is, $b_i=\gamma\cdot P_i$. Then jump to step 6.
3.	Calculate the hardness h_k in the previous round of the auction, which aims to decide whether to cooperate or not.
4.	if $h_k > h_{th}^i$, the core will choose to cooperate in this round of auction. And, calculate its bid by its power status, that is, $b_i=\gamma\cdot P_i$.
5.	if $h_k < h_{th}^i$, the core will choose to retaliate. And, calculate its bid by the average execution time of the tasks in T, that is, $b_i=\varphi\cdot L_{avg}$.
6.	Send the bid to auctioneer (Task Agent).

The core agent's algorithm in the auction of each time slice is to carry out a proper bid. Firstly, the cores should decide whether to cooperate or not. If corporation is selected, the core has to carry out for a social goal—more uniform power status of the processor. Specifically, if the core has a higher power consumption in the passing ten time slices, it should carry out a much higher bid to avoid being selected to execute a task. The algorithm is listed in Table III.

C. Temperature Calculation Method

Real temperature measure method, like integrating advanced thermometers, can undoubted get very precise temperature dynamics. However, that kind of equipments are not integrated in all systems, or the number of them is limited. Some models calculate the temperatures by completely theoretic methods, which may be so unrealistic to be utilized to real systems. In this section we use an integrated method of calculating temperatures which utilized the empirical temperature property of an instruction-level event.

The GTFETS algorithm is evaluated by the temperatures traces of the processor in the simulation, and the calculation method is presented in this subsection. The symbols used in the temperature calculation method are listed as follows.

- T_p: the temperature of the processor (°C).
- $q_j(x,y)$: the power density of the jth core (W/m^2).
- $u_j(x,y,t)$: the temperature of the jth core on plane z at time t
- S_j: the area of core j
- γ_j: the constant values that used in the superposition of different cores' temperature.
- h: the conductance of processor's interface material, $h=k/d$.
- a_v: event power, which is counted by performance counter v.
- t_j: the number of times the event j happens.

We adopt the similar thermal model used in [17] and its physical architecture is depicted in Figure 2.

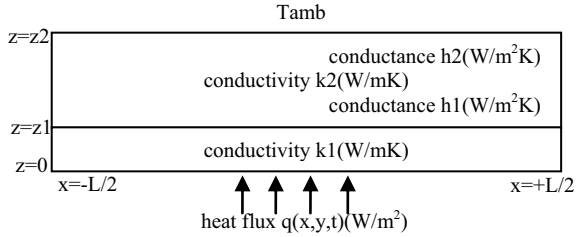


Figure 2. ATMI physical model

Let $u_{(0,0,0)}(x,y,t)$ to denote the temperature generated on the plane $z=0$ when a core located at point $(0,0,0)$. Then the temperature of the *jth* core is:

$$u_j(x_j,y_j,t)=\iint q_j(x,y)u_{(0,0,0)}(x_j-x,y_j-y,t)dxdy \quad (1)$$

Then, by adopting the concept of superposition of ATMI model, the temperature of processor can be calculate by

$$T_p=\sum_{j\in N}\gamma_j u_j(x_j,y_j,t) \quad (2)$$

We give a simpler and faster core power estimation method. Generally, most of the processors include some dedicated hardware performance counters for debugging and measurement. Hardware performance counters includes event signals generated by processor functional units, event detectors detecting these signals and triggering the counters, and configured counters incrementing according to the triggers. As the read operation of performance counters can be finished by only several assembly instructions, we can read one or more performance registers to calculate each core power in every or several time slots which ensures no influence on the timing requirements of real-time systems. The performance counter used here includes cycle counts, the number of floating point register accesses, and instructions executed. A single core power density is:

$$q_j(x,y)=\sum_{v=1}^{m}a_v t_v / S_j \qquad (v=1,2,\cdots,m) \quad (3)$$

We adopt event power a_v as in [18], which is depicted in Table IV.

TABLE IV.
EVENT POWER VALUE a_v

Event	Event Power(W)
time stamp counter	11.23
unhalted cycles	14.53
retired branches	26.79

Substituting Eq.3 in Eq.2, we can get Eq.4.

$$T_p=\sum_{j\in N}\gamma_j\iint\frac{\sum_{v=1}^{m}a_v t_v}{S_j}u_{(0,0,0)}(x_j-x,y_j-y,t)dxdy \quad (4)$$

IV. SIMULATION

The GTFTES scheduling algorithm for multi-core systems is implemented in our EASS (Energy-Aware Scheduling Simulator). In this section, the experimental setup, algorithm evaluation methodologies and simulation results is presented.

A. Experimental Setup

*1) Platform:*To verify the proposed algorithm, the ATMI[17] thermal model is adopted. The ATMI thermal model takes power traces as input and outputs the steady or transient temperatures. The ATMI parameters in our experiment are listed in Table V.

TABLE V.
ATMI PARAMETERS

Parameters	Unit	Value
Heat Sink width	Meter	0.10
Heat sink thermal resistance	Meter	0.3
Copper thickness	Meter	5e-3
Silicon thickness	Meter	5e-4
Interface thickness	Meter	1e-4
Interface thermal conductivity	W/mK	4

Artificial Thread : In our simulator, we use artificial thread which is defined as a C struct (Listed as follows) for simulation.

```
struct task { /* task id */
        int tid ;
        /* thread ' s power density contribute*/
        float power_density_contribution;
        /* task' s running core */
        int execusive_core;
        /* deadline of a task*/
        int deadline ;
        /* arrivial time of a task*/
        int arrivial_time ;
        /* execution time of a task*/
        int execution_time ;

        /* worst case execution time of a task*/
        int worst_case_execution_time ;

        /* thread state
         RUNNINGSTATE: the task is running now.
        EADYSTATE: the task is ready for running
        */
        int state ;

        /*The simulation step that a processes may occupy*/
        int duration_step ;

        /*The range that a task's simulation step number*/
        int duration_step_range ;

        /*The deadline penalty that a task has to pay*/
        float deadline_penalty;

        /*The thermal penalty that a task has to pay*/
```

```
    float thermal penalty ;
};
```

Tasks are classified into Cool, Warm and Hot three types, as listed in TABLE VI. A sample task set is given in Table VI.

TABLE VI.
A SAMPLE TASK SET.

Task	Arrival Time	Deadline	Power Contribution	Execution Core	Thermal Type
τ_1	31	37	$13.131*10^5$	1	Hot
τ_2	7	37	$11.382*10^5$	1	Hot
τ_3	3	42	$14.308*10^5$	1	Hot
τ_4	4	30	$10.224*10^5$	1	Hot
τ_5	4	40	$6.538*10^5$	2	Warm
τ_6	22	40	$8.257*10^5$	2	Warm
τ_7	16	18	$0.641*10^5$	2	Cold
τ_8	10	20	$4.814*10^5$	2	Cold
τ_9	27	30	$8.126*10^5$	2	Warm
τ_{10}	9	14	$4.972*10^5$	2	Cold
τ_{11}	30	36	$11.221*10^5$	3	Hot
τ_{12}	9	35	$11.333*10^5$	3	Hot
τ_{13}	5	41	$13.314*10^5$	3	Hot
τ_{14}	3	29	$10.226*10^5$	3	Hot
τ_{15}	2	36	$7.548*10^5$	3	Warm
τ_{16}	20	39	$6.153*10^5$	4	Warm
τ_{17}	17	20	$0.289*10^5$	4	Cold
τ_{18}	11	21	$5.113*10^5$	4	Cold
τ_{19}	26	31	$8.143*10^5$	4	Warm
τ_{20}	10	13	$4.953*10^5$	4	Cold

B. Methodology

ATMI can model different parts of processors, such as Floating Point Unit (FPU), and consider their influences on the processor energy. We simulate cores of a multi-core processor as the ATMI parts of a processor. Since our algorithm focuses on the core energy consumption, we only simulate the cores without other parts of processor (FPU, Instruction Cache, Data Cache, L2 Cache and so on). Of course, these omitted parts have influences on the processor's temperature, for simplicity, we suppose that the influence is zero. We rewrite ATMI to implement our core temperature calculation method and thread thermal contribution prediction. Every core corresponds to an ATMI image. The ATMI outputs processor temperature $T_{processor}$ and $T_{core,i}$, where $1 \leqslant i \leqslant n$ is the core index, i.e., we first consider a 4-core processor.

The ultimate goal of the GTFTES is to get uniform temperature and power distribution. So, in the simulation we have to decide the factor (power or temperature) to evaluate the efficiency of the algorithm. As we all know, excessive power consumption leads to short battery life, higher utility costs, and large currents in interconnect, and elevated temperatures. Also, the power can fluctuate in a large range in a short time, while the temperature may not. All those come to our decision to use temperature as a measure of the GTFTES algorithm.

We simulate the NBS-EATA algorithms in [9], a pure Tit-for-Tat algorithm where cores bid without consideration of corporation and our GTFTES algorithm. Firstly, we simulate the temperature differences irrespective of the task types (aperiodic tasks, periodic tasks and sporadic tasks). After that, task types are involved in the simulation, and in this part, the temperature distributions are evaluated. Finally, the simulation is extended to an eight-core processor.

C. Simulation Results of Temperature Difference

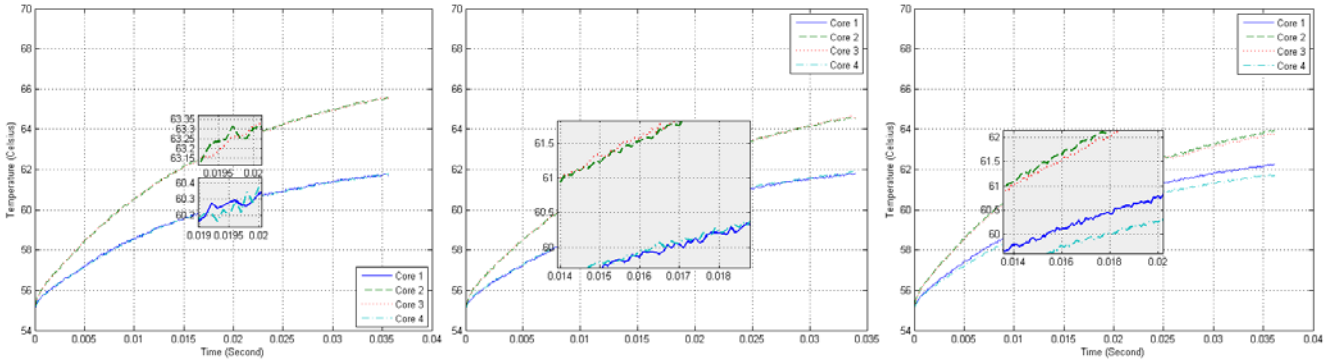
The temperature difference between cores is an important evaluation factor for the algorithms, since the smaller the difference the more uniform power density map will the algorithms provide.

We simulate different task sets with 50, 100 and 200 tasks respectively. The GTFTES algorithm performs better than NBS-EATA and a pure Tit-for-Tat algorithm. For the sake of easy reference, we call the pure Tit-for-Tat algorithm PTFT algorithm.

Our processor floorspan contains four cores sitting in a line. Thus, the two middle cores in the processor get higher temperatures than the other two cores. That is why, in our results, the four cores' temperatures fell into two categories: side cores and middle cores (As illustrated in Figure 3-5, the upper two lines are the temperature of middle cores, and the lowers are the side cores'.). For the sake of clearness, we set one or two amplifiers (grey blocks in each subfigures of Figure 3-5.) to get a close look of the data.

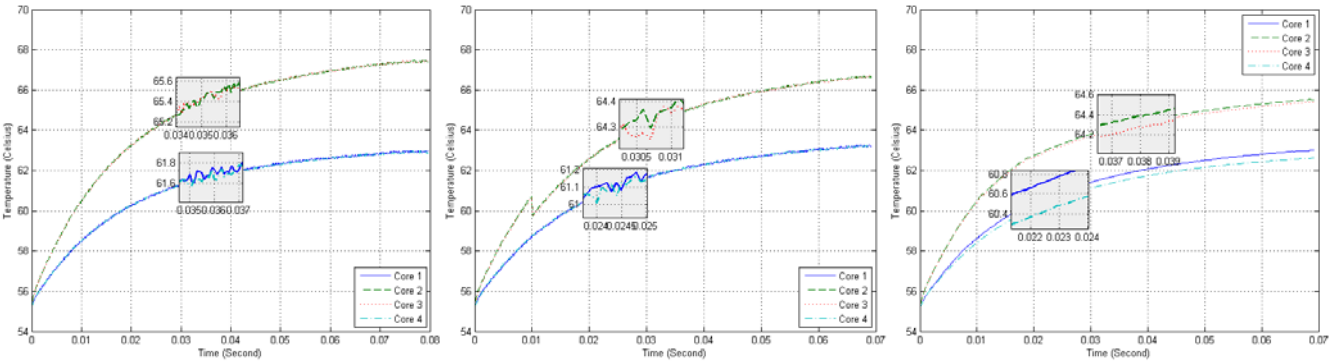
Take Figure 4 as an example, during the time period from 0 to 0.08, the temperature difference of the two groups of cores grows steadily from nearly 0 to almost 5 centigrade. In the GTFTES algorithm, the cores sometimes are forced to cooperate or they may choose to cooperate in a round of auction. It means cores that choose to cooperate may lose the auction because of its high thermal status, consequently leading to a lower temperature. Therefore, as subfigures c in Figure 3-5 show, the GTFTES algorithm finally gets a lowest temperature difference among the three algorithms.

Also, in Figure 5 which shows a temperature variation of a run of 200 tasks, GTFTES also shows a smaller difference between the two sets of core (around one centigrade lower than the NBS-EATA algorithm and two centigrade lower than the PTFT algorithm). Thus, the temperature difference between the central and side part of the processor is smaller.



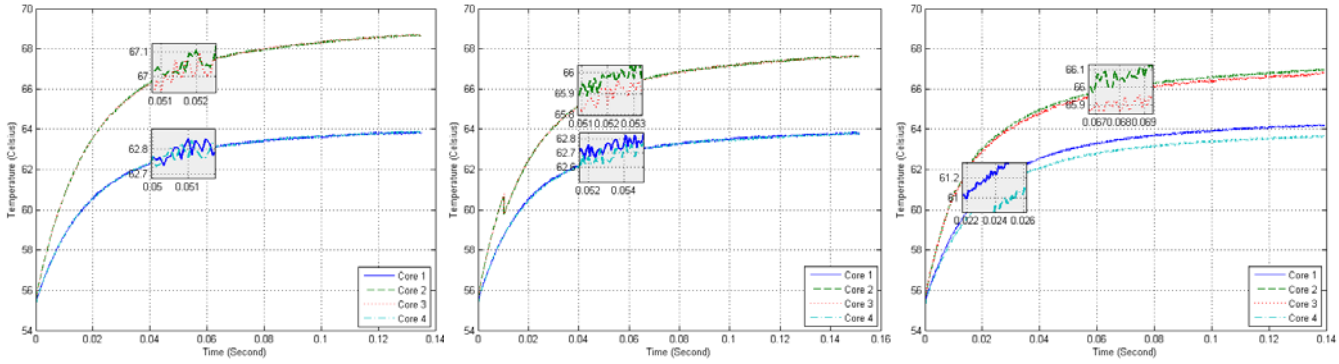
(a) PTFT algorithm (b) NBS-EATA (c) GTFTES

Figure 3. Temperature variations with 50 tasks.



(a) PTFT algorithm (b) NBS-EATA (c) GTFTES

Figure 4. Temperature variations with 100 tasks.



(a) PTFT algorithm (b) NBS-EATA (c) GTFTES

Figure 5. Temperature variations with 200 tasks.

D. Simulation Results of Different Types of Tasks

We generate three task sets with 250 aperiodic, periodic and sporadic tasks. Then, we simulate the GTFTES, NBS-EATA and PTFT algorithms separately. After the simulation we get numerous discrete time points and the temperature at those points. The temperature distribution on those discrete time points has to be listed to show the competence of the GTFTES algorithm. Instead to show a temperature curve, we use TABLE VII to list the statistical results in detail for a clearer illustration.

From the shaded grids of TABLE VII, we can see that the temperature range with majority tasks is decreasing gradually from ⩾68 ºC in PTFT to 66-68 ºC in NBS-EATA

and finally 64-66 ºC in GTFTES. Thus, we can say GTFTES can deal with aperiodic, periodic and sporadic tasks all with a better effect to lower the temperature of processors.

TABLE VII.
TEMPERATURE DISTRIBUTION OF PTFT, NBS-EATA AND GTFTES WITH 250 TASKS

	Percentages of the corresponding temperatures(%)								
	PTFT			NBS-EATA			GTFTES		
	Aperiodic	Periodic	Sproadic	Aperiodic	Periodic	Sproadic	Aperiodic	Periodic	Sproadic
⩽56ºC	4.22	4.51	3.52	3.35	4.16	6.17	4.23	4.16	5.16
56-58 ºC	8.45	7.16	9.16	12.54	13.17	12.46	11.66	13.27	13.89
58-60 ºC	11.46	12.16	12.45	15.32	15.33	14.33	14.52	15.13	16.11
62-64 ºC	13.3	14.33	14.33	17.97	16.38	16.85	18.87	18.48	20.83
64-66 ºC	14.98	13.88	15.75	21.21	19.72	23.31	30.24	29.51	28.44
66-68 ºC	19.1	18.11	19.21	29.24	30.51	26.42	20.21	18.72	15.31
⩾68 ºC	28.12	29.12	25.12	0	0	0	0	0	0

Note: Aperio: Aperiodic tasks; Perio: Periodic tasks; Sproa: Sporadic tasks. Gray grids represent the highest temperature range in a column.

TABLE VIII.
TEMPERATURE DISTRIBUTION OF PTFT, NBS-EATA AND GTFTES WITH 250 TASKS IN A EIGHT-CORE PROCESSOR

	Percentages of the corresponding temperatures(%)								
	PTFT			NBS-EATA			GTFTES		
	Aperiodic	Periodic	Sproadic	Aperiodic	Periodic	Sproadic	Aperiodic	Periodic	Sproadic
⩽56ºC	3.22	4.61	5.42	2.15	4.15	4.17	5.16	4.2	6.06
56-58 ºC	7.45	7.06	7.28	13.73	13.28	13.46	10.73	12.18	12.98
58-60 ºC	9.46	11.25	12.33	17.36	15.25	15.31	14.59	16.18	16.12
62-64 ºC	12.3	12.25	13.43	19.94	16.46	16.77	21.8	18.48	21.93
64-66 ºC	13.98	16.87	16.66	26.23	24.63	24.4	36.14	26.61	25.34
66-68 ºC	22.1	19.12	22.32	20.21	25.5	25.43	18.31	21.62	17.31
⩾68 ºC	31.12	28.11	22.1		0	0	0	0	0

Note: Aperio: Aperiodic tasks; Perio: Periodic tasks; Sproa: Sporadic tasks. Gray grids represent the highest temperature range in a column.

D. Simulation Results of Different Number of Cores

In order to show the core adaptability, we simulate the GTFTES algorithms under an eight-core processor. By listing the temperature distribution of this round of simulation (TABLE VIII), the similar conclusion that the GTFTES gets lower temperatures than PTFT and NBS-EATA algorithms can be reached.

V. RELATED WORKS

The majority of the power management research focuses on power management for the purpose of saving energy, not for balancing power consumptions and maintaining safe temperature levels [19-22]. While energy and temperature are closely related, power control mechanisms for energy and temperature are quite different. Although it has been shown that many energy-saving techniques do not work well in reducing peak temperature [23-25], there is still some change to design energy-balancing techniques in multi-core systems.

In this section, we generally look into the power management techniques from the perspective of temperature management, power management, and finally the game theory methods used in energy-related scheduling.

Firstly, from the perspective of temperature management, the authors [26] study the thread migration in temperature constrained multi-core processors. They show that the thermal benefit of thread migration depends on the number of threads, characteristics and ambient temperature. The thread migration algorithm makes thread exchange when a cold and a hot core are detected. They demonstrate that their method yields the same throughput with HRTM (Heat-and-Run Thread Migration) [6], but requires fewer migrations. Authors in [2] a temperature-aware task scheduling algorithm in microprocessor systems.

Secondly, several power management techniques have been proposed and applied in modern processors via either hardware or software mechanisms [3, 17, 18]. Hardware level DTM mechanisms, such as Dynamic Frequency Scaling(DFS) and Dynamic Voltage Scaling(DVS) as well as clock gating, are able to reduce processors temperature effectively and guarantee thermal safety, but with high performance loss. So some literature adapts to the software based mechanisms. As to the multi-core processors power management, there are some software-based mechanisms, such as an power density management approach based on operating system is studied in [6]. The proposed method has two key components: SMT thread assignment and CMP thermal migration. Within the heat-and-run the SMT thread assignment attempts to increase processor-resource utilization by co-scheduling threads using complementary resources; the CMP thread migration cools overheat by migrating threads. They show that their mechanism achieves 9% higher average throughout than stop-go and 6% higher average throughout than DVS. The authors [27] propose a

method to balance power consumption in multiprocessor systems. They use performance counters to create energy profiles which is used to describe the energy characteristics of individual tasks, and distribute energy consumption of all CPUs of a system. Their results show a 5% increase in throughput for a system with all CPUs busy by avoiding the throttling of processors. As stated above, these methods pay their attentions on the power management rather than the temperature management. The authors [28] present a method to minimize the total energy consumption and guarantee deadline of each task. They present a mixed-integer linear programming model for the NP-complete scheduling problem and solve it for moderate sized problem instances using a public-domain solver. For larger task sets, they present a novel low-energy earliest deadline first (LEDF) scheduling algorithm and apply it to two real-life task sets. However, they do not pay much attention to multi-core systems and energy balancing management.

Finally, authors in [8, 9] propose a corporative game theoretic method to address the problem of power-aware scheduling/mapping of tasks onto heterogeneous and homogeneous multi-core processor architectures. They consider the problem as a multi-objective optimization problem. However, they did not pay much attention to the temperature problems which is a key factor in the energy consumption[25].

VI. Conclusion

In this article, we propose a generalized tit-for-tat based energy-aware scheduling algorithm, namely GTFTES, for multi-core systems. Right before each auction, each core will decide to cooperate or not by the hardness factor. If it decides to cooperate, it will bid according to its power status. Otherwise, it will bid according to the average execution time of the tasks. Moreover, we have given core temperature calculation method which is used in the result analysis stage. In GTFTES, the scheduler forces the cores to cooperate if some core is thermal saturated. Simulation results show that the proposed game can surely reduce the temperature between different groups of cores and avoid the hotspot of the processor.

Acknowledgment

This paper is partially supported by the National Natural Science Foundation of China under Grants No.60703101 and No. 60903153, and the Fundamental Research Funds for the Central Universities.

References

[1] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro,* vol. 19, pp. 23-29, 1999.

[2] M. Chrobak*, et al.*, "Algorithms for Temperature-Aware Task Scheduling in Microprocessor Systems," presented at the Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management, Shanghai, China, 2008.

[3] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration," *SIGARCH Comput. Archit. News,* vol. 34, pp. 78-88, 2006.

[4] N. S. Fadi, "Simulation and Performance Analysis of Multi-core Thread Scheduling and Migration Algorithms," 2010, pp. 895-900.

[5] S. Heo*, et al.*, "Reducing power density through activity migration," presented at the Proceedings of the 2003 international symposium on Low power electronics and design, Seoul, Korea, 2003.

[6] M. Gomaa*, et al.*, "Heat-and-run: leveraging SMT and CMP to manage power density through the operating system," *SIGOPS Oper. Syst. Rev.,* vol. 38, pp. 260-270, 2004.

[7] J. A. Winter and D. H. Albonesi, "Scheduling algorithms for unpredictably heterogeneous CMP architectures," in *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, 2008, pp. 42-51.

[8] I. Ahmad*, et al.*, "Energy-Constrained Scheduling of DAGs on Multi-core Processors," in *Contemporary Computing*. vol. 40, S. Ranka*, et al.*, Eds., ed: Springer Berlin Heidelberg, 2009, pp. 592-603.

[9] I. Ahmad*, et al.*, "Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, 2008, pp. 1-6.

[10] R. K. Dash*, et al.*, "Market-Based Task Allocation Mechanisms for Limited-Capacity Suppliers," *Systems, Man and Cybernetics, Part A, IEEE Transactions on,* vol. 37, pp. 391-405, 2007.

[11] Y. Wang*, et al.*, "Economic-Inspired Truthful Reputation Feedback Mechanism in P2P Networks," in *FTDCS '07: Proceedings of the 11th IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2007, pp. 80-88.

[12] A. Pekeč and M. Rothkopf, "Combinatorial Auction Design," *Management Science,* vol. 49, pp. 1485-1503, 2003.

[13] M. H. Rothkopf*, et al.*, "Why Are Vickrey Auctions Rare?," *The Journal of Political Economy,* vol. 98, pp. 94-109, 1990.

[14] W. Guowei, "GTFTTS: A Generalized Tit-for-Tat Based Corporative Game for Temperature-Aware Task Scheduling in Multi-core Systems," 2010, pp. 81-88.

[15] W. Huang*, et al.*, "Hotspot: A compact thermal modeling method for CMOS VLSI systems," *IEEE Transactions on,* vol. 14, pp. 501-513, 2006.

[16] G. Wu and Z. Xu, "Temperature-aware task scheduling algorithm for soft real-time multi-core systems," *Journal of Systems and Software,* vol. 83, pp. 2579-2590, 2010.
[17] P. Michaud. (2009, *ATMI manual*. Available: http://www.irisa.fr/alf/
[18] G. Magklis*, et al.*, "Profile-based dynamic voltage and frequency scaling for a multiple clock domain microprocessor," *SIGARCH Comput. Archit. News,* vol. 31, pp. 14-27, 2003.
[19] H. Aydi*, et al.*, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," presented at the Proceedings of the 22nd IEEE Real-Time Systems Symposium, 2001.
[20] Y. Liu and A. K. Mok, "An Integrated Approach for Applying Dynamic Voltage Scaling to Hard Real-Time Systems," presented at the Proceedings of the The 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003.
[21] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," presented at the Proceedings of the eighteenth ACM symposium on Operating systems principles, Banff, Alberta, Canada, 2001.
[22] A. Qadi*, et al.*, "A Dynamic Voltage Scaling Algorithm for Sporadic Tasks," presented at the Proceedings of the 24th IEEE International Real-Time Systems Symposium, 2003.
[23] N. Bansal*, et al.*, "Dynamic Speed Scaling to Manage Energy and Temperature," presented at the Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, 2004.
[24] N. Bansal*, et al.*, "Speed scaling to manage energy and temperature," *J. ACM,* vol. 54, pp. 1-39, 2007.
[25] K. Skadron*, et al.*, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.,* vol. 1, pp. 94-125, 2004.
[26] P. Michaud*, et al.*, "A study of thread migration in temperature-constrained multicores," *ACM Trans. Archit. Code Optim.,* vol. 4, p. 9, 2007.
[27] A. Merkel and F. Bellosa, "Balancing power consumption in multiprocessor systems," *SIGOPS Oper. Syst. Rev.,* vol. 40, pp. 403-414, 2006.
[28] V. Swaminathan and K. Chakrabarty, "Real-time task scheduling for energy-aware embedded systems," *Journal of the Franklin Institute,* vol. 338, pp. 729-750, 2001.

Guowei Wu, born in 1973, male, is currently an associate professor in School of Software, Dalian University of Technology. He received his Ph.D degree from Harbin Engineering University, China, in 2003. His research interests include real-time embedded system, cyber-physical systems(CPS), wireless sensor network.

Zichuan Xu, born in 1986, male, received Master degree and B.E degree (all with highest honor) from Dalian University of Technology, China. His research interests include approximation algorithms, temperature-aware scheduling, cyber-physical systems (CPS), game theory, green computing.

Qiufen Xia, born in 1987, received the M.A. from Dalian University of Technology, Dalian in 2009. She is a master student of software school, Dalian University of Technology. Her study interests include grid resources scheduling, game theory, network security and cloud computing.

Jiankang Ren, born in 1986, male, received the B.E. and M.E. degrees from Dalian University of Technology, China, in 2008 and 2011, respectively. His current research interests involve fault-tolerant and reliable computation in cloud computing, QoS and managing strategies, security and privacy in Body Sensor Networks and Cyber Physical Systems.