Energy-Efficient Allocation of Periodic Real-Time Tasks for Heterogeneous System by Execution-Variance

Yang Qin[*], Gang Zeng[†], Ryo Kurachi[*], Yutaka Matsubara[*] and Hiroaki Takada[*]

Abstract--The research of energy-efficient heterogeneous multiprocessor scheduling is becoming more and more popular today. The problem can be addressed as allocating tasks to the right core and set the appropriate operating frequency. In this paper, we present a task allocation algorithm for a single-ISA heterogeneous system consisted of two different types of cores: high-performance core and energy-efficient core. Based on the execution-variance of two types of cores, a heuristic that approximate the optimal load distribution is proposed. We experimentally simulate our method and compare it to state-of-the-art solutions, resulting average in about 20% less energy consumption (and up to 48% for some cases) with guaranteeing the deadline of all tasks.

1. INTRODUCTION

The energy consumption is a critical issue for real-time tasks which determines the utility of battery life directly in embedded system. The breakdown of Dennard Scaling causes most CPU manufactures to focus on multi-core processors as an alternative way, while with the diverse needs for performance, the research on heterogeneous system becomes more and more popular in recent decades. A single-ISA heterogeneous multi-core has attracted much attention as a solution to minimize energy while improve performance, where all the cores on the chip share the same instruction set architecture but differ with power and performance characteristics. A typical example of single-ISA multi-core is big.Little architecture, which consists of two different kinds of cores, i.e. EXYNOS 5 Octa series [1] employ Cortex-A7 as energy-tuned core, which named ‘Little core’, and choose Cortex-A15 as high-performance core, which named ‘Big core’.

Several studies have been made on the energy-efficient scheduling of big.Little architecture, and the most simple and intuitive scheme is Lower-power First, for which the lower-power cluster was always first set to run at the maximum frequency until the tasks violate the deadline. For example, to fully utilize the lower-power cores, Liu [2] adopted the task-splitting approach to make little cores operate at full speed. Besides, according to the algorithm proposed by Pagani [3], the allocation result derived on platform EXYNOS 5422 is the same as Lower-power First.

Honestly, the power consumption of big core is always higher than little core because of its architecture design, but in terms of energy consumption for real-time system, we have to take heterogeneity into consideration. In this paper, we define a key factor $r_{dif,i} = \frac{WCET_{L,i}}{WCET_{B,i}}$ to illustrate the execution-variance of two cores, where $WCET_{L,i}$ and $WCET_{B,i}$ are Worst-Case-Execution-Time of a real-time task τ_i on little and big cores, respectively. The contributions of this work are summarized as follows:

1) We extend the previous algorithms to consider each core is DVFS-Capable based on the single-cluster design supported by dynamIQ technology, and propose tasks allocated by the ordering of execution-variance r_{dif} can reduce system utilization and save more energy.

2) We analyze the properties of execution-variance r_{dif} and propose a simplified method to calculate the desired utilization distribution, and based on the calculated result, we propose a heuristic to approximate the distribution.

2. SYSTEM MODELS

In this section, we present the platform model, task model, as well as the energy model used in this work.

2.1. Platform Model

We consider a heterogeneous multi-core system which consists of two types of processing elements: high-performance (out-of-order) and efficiency-tuned (in-order), where both the elements composed with N number of processors. We use π_L and π_B to denote the

[*]Graduate School of Information Science, Nagoya University
[†]Graduate School of Engineering, Nagoya University

cluster composed with little cores and big cores, respectively. The two types of cores differ by their power and performance characteristics but share the same ISA. Based on the single-cluster design supported by next generation DynamIQ big.Little technology [4], we assume each core is DVFS-capable. The power consumption at frequency f is presented as following formulation [5]:

$$P(f) = kf^{\alpha} + \beta \qquad (1)$$

Where the first term is the frequency-dependent power consumption and the second term is the frequency-independent power consumption.

2.2. Task Model

We model a real-time application by considering a task set Γ consists of n independent periodic real-time tasks $\{\tau_1, \tau_2, ..., \tau_n\}$. Task $\tau_i \in \Gamma$ is characterized by $\{WCET_{L,i}, WCET_{B,i}, T_i, D_i\}$, where T_i is period time, which is assumed to be equal to its relative deadline D_i, i.e. $T_i = D_i$. The utilization of task τ_i allocated to core j is $u_{j,i} = \frac{WCET_{j,i}}{T_i}$, and the sum utilization of a task set is $U_j = \sum_{\tau_i \in \Gamma_j} u_{j,i}$. As we consider two types of cores only in this paper, core j is either big core or little core.

2.3. Task Model

Considering different tasks may have different periods, the energy consumption of system can be calculated within the minimum repeating interval, hyperperiod time L, where $L = LCM(\{T_i : \tau_i \in \Gamma\})$. With the assumption of system and task model above, the energy consumption of a core over the interval L is the sum energy in active mode and idle mode, which can be calculated as follows:

$$E(f_j, U_j) = L[\frac{U_j f_{j,MAX}}{f_j} P_j(f_j) + (1 - \frac{U_j f_{j,MAX}}{f_j}) Pidle_j] \qquad (2)$$

Where, $f_{j,MAX}$ is the maximum frequency, U_j is the computational of Γ, $Pidle_j$ is the idle power.

3. TASK ALLOCATION

3.1 Desired Utilization Division

For a given task set, the task allocation for the big.Little platform can be concluded as dividing the total utilization U into two parts U_1 and U_2, where U_1 denotes the utilization capacity of tasks that allocated to the little cluster $U_1 = \sum_{\tau_i \in \Gamma_L} u_{L,i}$, and U_2 denotes the utilization allocated to the big cluster $U_2 = \sum_{\tau_i \in \Gamma_B} u_{B,i}$. According to previous method DL-CAP [6], the optimal distribution is formulated as a convex optimization problem by dividing the load into as many fragments as cores and splitting each fragment into as many sub-fragments as tasks. As their solution has to be solved by specified toolbox, the tasks are generated in the condition that the optimal division is known in advance, which makes their method not general. We propose a simplified solution for the convex optimization problem by using execution-variance r_{dif}.

Theorem 1 If a platform consists of two different cores, assigning tasks with large r_{dif} to big core and tasks with small r_{dif} to little core guarantees the minimum system utilization.

Proof. Suppose that, U_1 and U_2 are optimal utilization distribution for a given task set. If part of the utilization U_y from big core is picked out to switch to little core, where for U_y the corresponding execution-variation is r_y; In this case, we choose to switch U_x from little core to big core, where $U_x = U_y r_y$, to guarantee the utilization of little core equals U_1. Considering U_x is selected from little core, we can get the execution-variance r_x meeting: $r_x \le r_y$.

The variation of system utilization after switching can be presented as: $U' = (U_y r_y + \frac{U_y r_y}{r_x}) - (U_y + U_x)$, where the first term $(U_y r_y + \frac{U_y r_y}{r_x})$ is the additional utilization, the second term $(U_y + U_x)$ is the removed utilization. As $U_x = U_y r_y$, by replacing U_x with $U_y r_y$, U' can be simplified as: $U' = \frac{U_y r_y}{r_x} - U_y$. As $r_x \le r_y$, we can get $U' \ge 0$, which means, switching tasks that violate the ordering rule of r_{dif} must increase the total system utilization.■

According to previous work, load-balance has been proven to be most energy efficient for homogeneous system, combined with the above theorem, the desired utilization distribution of a task set ranked in increasing order of r_{dif} can be calculated as follows:

$$\text{Minimize} \quad E = N[E(U_1, f_1) + E(U_2, f_2)] \qquad (3)$$

$$\text{s.t.} \quad U_1 + U_2 = \frac{U}{N}$$

$$U_1 = \sum_{i=0}^{m} u_{L,i} + x u_{L,m+1}$$

$$U_2 = \sum_{i=m+2}^{n} u_{L,i} + (1-x) u_{B,m+1}$$

$$0 \le m \le n \text{, m is integer}$$

$$0 \le x \le 1$$

Where *n* is the number of tasks, *m* is the first *m* tasks selected to allocate to little cluster, *x* is the partition of the first *m+1* task divided to little cluster. All possible *m* and *x*, with minimum step of 0.01, are tested to find the most energy-efficient division of utilization. Consider that, after the utilization is divided into two parts, each cluster is executed at a single voltage/frequency, specifically, the Single Frequency Approximation (SFA) scheme [7] which aims at finding the minimum frequency can be used for calculating the operating frequencies f_1 and f_2 for little and big clusters, respectively.

3.2. Proposed Task Allocation Algorithm

As part 3.1 has introduced the calculation method of desired utilization distribution, in this section, we describe how the tasks are allocated to heterogeneous systems by DL-EVBA (Desired-Load Execution-Variance- Based Algorithm).

Algorithm 1: DL-EVBA

Input: task set Γ, platform Π;

Output: task allocation queue Θ and frequency settings of each core Q;

```
1: Function DL-EVBA (Π, Γ)
2: order tasks in order of increasing $r_{dif}$
3: ($U_1$,$U_2$,$f_1$,$f_2$)<--Utilization-Division-Calculation (Π, Γ)
4: (Θ and Q)<--m-pwr(Γ, $f_1$,$f_2$)
5:   if Θ!= ∅
6:       (Θ and Q)<--m-pwr(Γ, $f_{L,MAX}$, $f_{B,MAX}$)
7:   end if
8: return Θ and Q
9: end Function
```

Algorithm 1 gives a general description of our heuristic method, denoted by DL-EVBA. Tasks are first ordered by their execution-variance in line 2, then desired utilization division and minimum operating frequencies in line 3 are calculated as part 3.1. Available capacities of each cluster are first set as f_1 and f_2 as line 4, but if tasks cannot be finished allocating, the capacities are restored to the maximal frequencies $f_{L,MAX}$ and $f_{B,MAX}$ as line 6. In this paper, to make fair comparison to DL-CAP, algorithm m-pwr is chosen to allocate tasks between the same cores in line 4 and 6, for which the minimum power consumption core is always selected for each task [6].

4. EXPERIMENTAL EVALUATION

In this section, we present the evaluation results of proposed method compared to two state-of-the-art related approaches: Lower-power First which aims at allocating more tasks to energy-efficient cores by Largest-Task-First (LTF) strategy [2,3], and DL-CAP which uses a heuristic to approximate the load distribution [6].

In the experiment, the power parameters are obtained from [6], where they tested the experiments based on real platform ODROID XU-3, and the energy consumption is measured as a function of two task set factors: the execution-variance r_{dif} and the total utilization *U* on little core. r_{dif} is the reflection of the difference between two types of cores and *U* is a measurement of the workload. The task generation procedure is summarized as follows:

- r_{dif} is selected from a uniform random distribution in the range of [2.8, 4.2], which is calculated from the performance variance between Cortex-A7 and Cortex-A15 [1].
- When the total utilization *U* is fixed, the utilization of each task on little core $u_{L,i}$ is generated in the range of [0.01, *MTU*] one by one, where *MTU* is the maximum utilization of each task. Then the utilization of the same task on big core can be computed as: $u_{B,i} = \frac{u_{L,i}}{r_{dif,i}}$.
- For each task, period T_i is randomly generated using a uniform distribution in the interval [1, 1000] [8], then the *WCET* is computed as: $WCET_{L,i}=T_i u_{L,i}$, $WCET_{B,i}=T_i u_{B,i}$.

All the experimental results are plotted as Figure 1 and 2. We generate 100 task sets and calculate the average normalized energy consumption for each experiment.

1) Impact of Execution-Variance: In the first experiment, as Fig.1, we fix the total utilization *U* as 10, *MTU* as 0.3 and vary the upper bound of r_{dif} interval. Here *U*=10 is randomly selected, as the same results can be obtained when *U* is different. As we can see, compared to Lower-power First, both the normalized energy of DL-CAP and DL-EVBA are decreased because of larger r_{dif}. Actually, When r_{dif} is from [2.8, 3.5], the average saved energy is 14.6%, while when r_{dif} is from [3.5, 4.2], the average saved energy is up to 23.1%. The reason of more saved energy is that larger r_{dif} contributes to better performance of big core. Briefly, when r_{dif} is increased, the execution time for the same task operated on big core is shortened, in that case, Lower-power First becomes energy wasting. Besides, DL-EVBA algorithm separates tasks into two sections in the first phase which effectively utilizes the priorities of r_{dif} and reduce the system utilization, as a result, when the value of r_{dif} increases, DL-EVBA saves more energy.

2）Impact of Per-Core-DVFS Assumption: In this experiment, we compare our method DL-EVBA to well performed DL-CAP. We fix the range of r_{dif} as [2.8,4.2], *MTU* as 0.3 and vary the total utilization *U*. Fig.2 shows that regardless of the workload, Per-Core-DVFS assumption ("different f") is always more energy-efficient than Per-Cluster-DVFA assumption ("fixed f"). Besides, r_{dif}-based ordering does not save energy for the case of "fixed *f*", but saves energy for "different *f*". The reason is that utilization-based allocation makes workload more balanced, while r_{dif}-based allocation guarantees less system utilization. When the frequency of the cluster is fixed, frequency is determined by the maximum utilization, as a result, balanced-workload is more energy-efficient. But when we assume frequency is different, frequency of each core is determined by their own utilization, in this case, less utilization makes less energy.

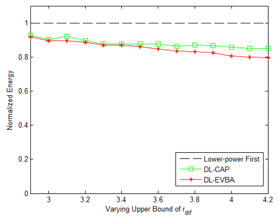

Figure 1. Normalized energy with respect to Lower-power First

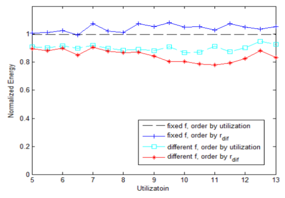

Figure 2. Normalized energy with respect to DL-CAP

5.CONCLUSION

The Desired-Load Distribution method [6] is studied in this work to efficiently schedule real-times tasks in heterogeneous system. We simplify the calculation method by using execution-variance and propose the DL-EVBA algorithm to allocate tasks. In contrast to existing allocation approaches, we propose to order tasks by r_{dif} based on the assumption that frequencies of the cores from one cluster could be different. The experimental results show the efficiency of our proposed method. For the future work, an alternative approach AMBFF [9] will be used for further comparison and task migration will be considered to make improvements.

REFERENCES

[1] P. Greenhalgh, Big. little processing with arm cortex-a15 and cortex-a7. *White Paper*, 2011.

[2] D. Liu et al, Energy-efficient scheduling of real-time tasks on heterogeneous multicores using task splitting. *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2016.

[3] S. Pagani, A. Pathania et al, Energy efficiency for clustered heterogeneous multicores. *IEEE Transactions on Parallel and Distributed Systems,* 2017.

[4] ARMDeveloper, "http://developer.arm/technologies/dynamiq."

[5] J.-J. Chen and C.-F. Kuo, Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2007.

[6] A. Colin, A, A. Kandhalu, R. R. Rajkimar, Energy-efficient allocation of real-time applications onto single-ISA heterogeneous multi-core processors. *Journal of Signal Processing Systems*, 2015.

[7] V. Devadas, H. Aydin. Coordinated power management of periodic real-time tasks on chip multiprocessors. In *Proceedings of International Conference on Green Computing (GREENCOMP' 10)*.

[8] R. Davis and A. Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems*, 2011.

[9] G. Zeng, T. Yokoyama and H. Takada, Practical energy-aware scheduling for real-time multiprocessor systems. *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2009.