# Energy-Aware Partitioning for Multiprocessor Real-Time Systems

M11007326 黃鈞臨

## 1、Introduction

- Multiprocessor scheduling of periodic tasks is one of the most extensively studied areas in real-time systems research.
- In global scheduling ,there is a single ready queue and task migrations among processors are allowed.
- In contrast, partitioning-based approach allocates each task to one processor permanently (thus, task migrations are not allowed) and resorts to well-established single-processor scheduling policies to guarantee the feasibility.

- The main low-power computing technique in real-time systems has been variable voltage scheduling (or, dynamic voltage scaling)
- In principle, it is possible to obtain striking (usually, quadratic) energy savings by reducing CPU speed.
- The feasibility of the schedule must be preserved even with reduced speed and this gives rise to the problem of minimizing energy consumption while meeting the deadlines.

- From energy-aware perspective, the advantage of partitioning-based approaches is the ability to apply well-established uniprocessor variable voltage scheduling techniques.
- The problem is invariably NP-Hard and appears in two variations: Minimizing the number of processors needed to guarantee the feasibility of the task set, or alternatively, given a fixed multiprocessor platform, finding sufficient schedulability (usually, utilization) bounds.
- Our work opts for the second setting, thus we assume the existence of a given number of processors

## 2、System Model

- The scheduling of a periodic real-time task set T = $\{T_1,...,T_n\}$ on a set of processors M= $\{M_1,...,M_m\}$.
- The period of $T_i$ is denoted by $P_i$, which is also equal to the relative deadline of the current invocation. All tasks are assumed to be independent and ready simultaneously at t = 0.
- If the speed of the processor $M_i$ during the time interval $[t_1, t_2]$ is given by S(t), then the energy consumed during this interval is

$$E(t1,t2)=\int_{t1}^{t2} g(S(t))\, dt$$

- We define $U_{tot}$ as the total utilization of the task set T under maximum speed Smax = 1, that is,

$$U_{tot} = \sum_{i=1}^{n} u_i = \sum_{i=1}^{n} \frac{C_i}{P_i}$$

- Finally, given a task-to-processor assignment Π, we will denote the utilization of processor $M_i$ under $S_{max}$ = 1.0 by $U_i(\Pi)$ .

# 3、Energy Minimization with Partitioning

- Find a task-to-processor assignment and compute task-level speeds on each processor such that: 1. the tasks assigned to each processor can be scheduled in a feasible manner, and 2. the total energy consumption of M is minimum (among all feasible task allocations)
- EDF is optimal from energy consumption point of view when used with a constant speed equal to the utilization of the task set assigned to that processor

- The schedules on all processors can be repeated at every hyperperiod P without hurting feasibility or energy-efficiency, we focus on minimizing energy consumption during P = lcm(P1,…,Pn).
- The energy consumption of task $T_j$ running on processor M in interval [0, P] when executed with constant speed S is given by:

$$g(S) \cdot \frac{P}{P_j} \cdot \frac{C_j}{S}$$

- The energy consumption of all tasks allocated to the processor Mi is therefore:

$$E(M_i) = \sum_{T_j \text{ assigned to } M_i} g(S) \cdot \frac{P}{P_j} \cdot \frac{C_j}{S}$$

- We find the minimum energy consumption on processor $M_i$ as

$$E^*(M_i) = P \cdot g(\bar{S}) \cdot \frac{U_i}{\bar{S}} = P \cdot g(U_i)$$

- Considering that P is a constant, independent of assignment and scheduling algorithm, we can now present the optimization problem which is equivalent to POWER-PARTITION:

$$\text{minimize} \quad \sum_{i=1}^{m} g(U_i) \qquad (3)$$

$$\text{subject to} \quad 0 \leq U_i \leq 1.0 \quad i = 1, \ldots, m \qquad (4)$$

$$\sum_{j=1}^{m} U_j = U_{tot} \qquad (5)$$

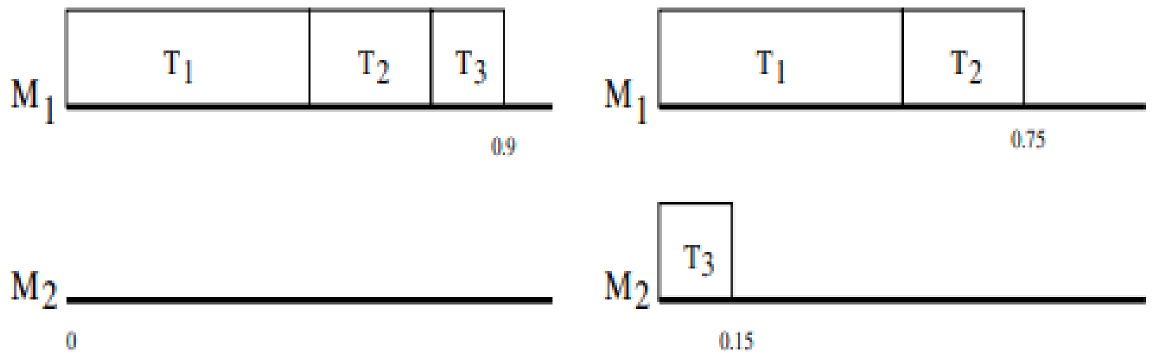

**Figure 1. Task Assignment Options 1 (left) and 2 (right)**

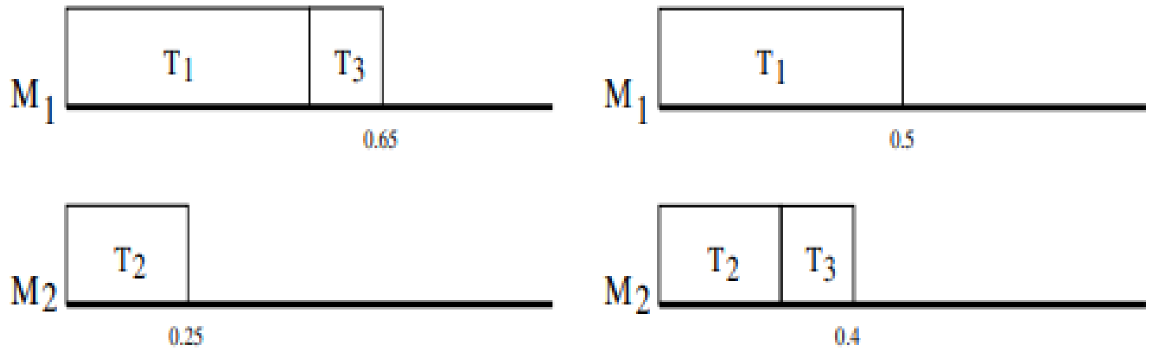

**Figure 2. Task Assignment Options 3 (left) and 4 (right)**

Consider three tasks with $u_1 = 0.5$, $u_2 = 0.25$ and $u_3 = 0.15$ to be executed on m = 2 identical processors2. Assume the power consumption function g(S) = S^2

1. All three tasks are allocated to one processor (Figure 1, left): Energy consumption= 0.9^2 = 0.81.
2. $T_1$ and $T_2$ are allocated to one processor and $T_3$ is allocated to the other processor (Figure 1, right): Energy consumption = 0.75^2 + 0.15^2 = 0.585.
3. $T_1$ and $T_3$ are allocated to one processor and $T_2$ is allocated to the other processor (Figure 2, left): Energy consumption = 0.65^2 + 0.25^2 = 0.485.
4. $T_2$ and $T_3$ are allocated to one processor and $T_1$ is allocated to the other processor (Figure 2, right): Energy consumption = 0.4^2 + 0.5^2 = 0.41.

# 4、Load Balancing For Energy Efficiency

- A task-to-processor assignment Π is said to be unbalanced if the total energy consumption can be reduced by moving one task from one processor to another. Otherwise, it is said to be balanced
- It is clear that the power-optimal task-to-processor assignment must be balanced, since by definition its total energy consumption cannot be reduced.
- In the motivational example of Section 3, the first three task assignments are unbalanced, while the fourth(optimal) one is balanced.
- However, a balanced partitioning is not necessarily optimal

## 5、Heuristics for POWER-PARTITION

- A wealth of efficient heuristics are already available for the feasibility aspect of the problem from Multiprocessor RealTime Scheduling: these include First-Fit (FF), Best-Fit (BF), Next-Fit(NF), Worst-Fit(WF).
- If the algorithm is allowed to preorder the task set according to utilizations, the term decreasing is added to its name : for example, we have Best-Fit Decreasing (BFD) version of Best-Fit (BF) algorithm.
- We call this class Reasonable Allocation Decreasing (or RAD, for short). Our simulation results support the expectation that the average case performance of this class of heuristics improves as well when they are first allowed to preorder tasks.

- Besides feasibility, our problem has an equally important goal: minimizing the energy consumption.
- We propose an additional metric called Timeliness/Energy that combines performances in both dimensions.
- Simulation Settings: We have generated a total of 1000000 task sets by varying the number of processors m, the total utilization of the task set $U = U_{tot}$ and the number of tasks n.
- In addition to these, the individual task utilization factor α has been another key parameter.
- When focusing on a multiprocessor platform with m processors, we modified U between m/10 (lightly loaded system) and m (heavily loaded system).

- For each heuristic H, we present:

➢The feasibility performance (FPH), given as the percentage of task sets that are feasibly scheduled by H.

➢The energy consumption performance (ECH), given as average energy consumption of task sets that are scheduled by H in feasible manner.

➢The timeliness/energy metric, given as FPH/ECH .

# 5.1 - Performance of Algorithms with Utilization Ordering

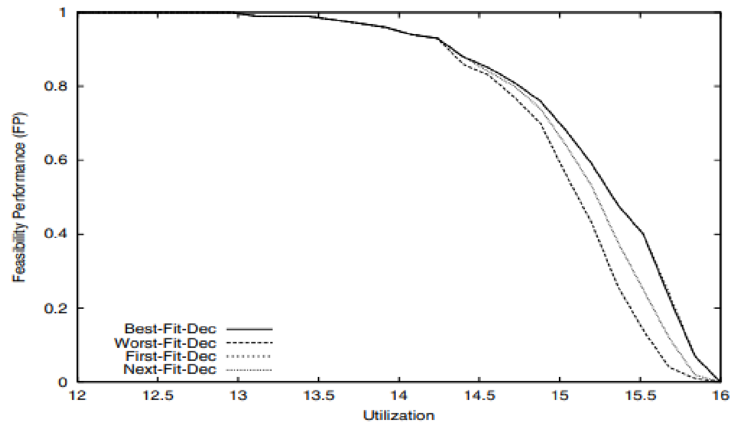

**Figure 3. Feasibility Performance for $\alpha = 1.0$**

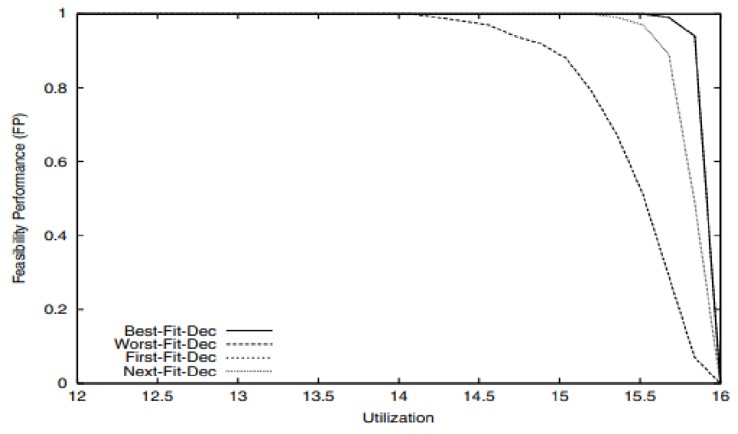

**Figure 4. Feasibility Performance for $\alpha = 0.5$**

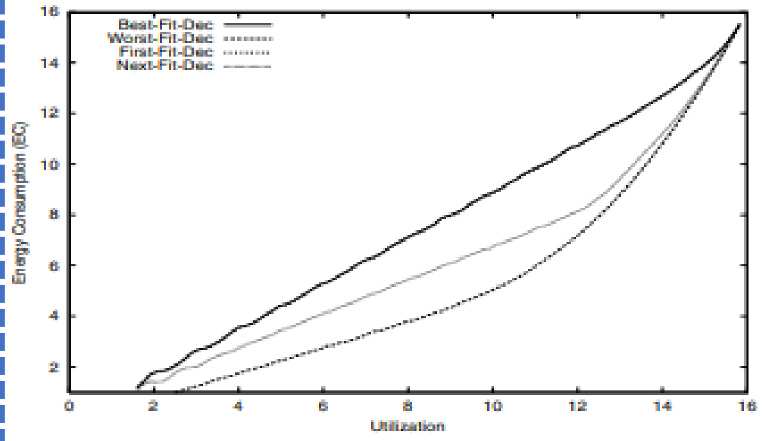

**Figure 5. Energy Performance for $\alpha = 1.0$**

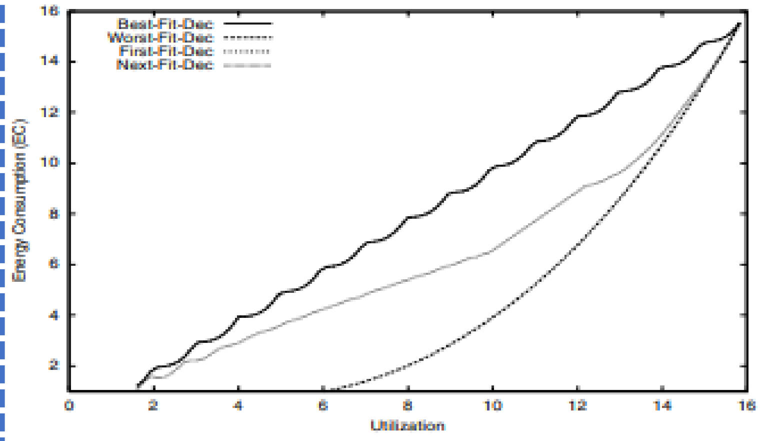

**Figure 6. Energy Performance for $\alpha = 0.5$**

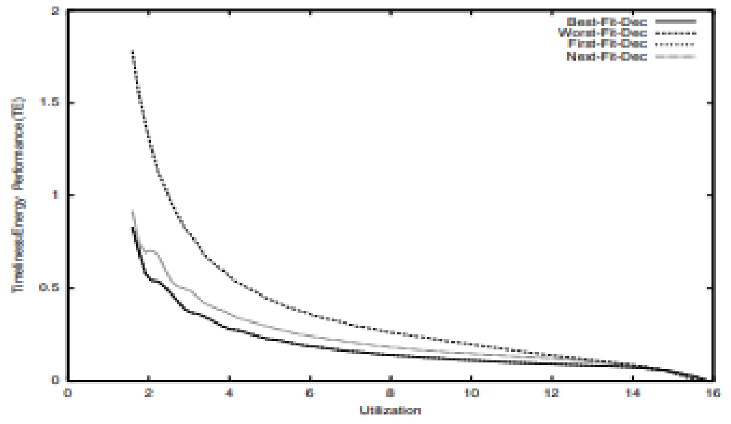

**Figure 7. Timeliness/Energy Performance for $\alpha = 1.0$**

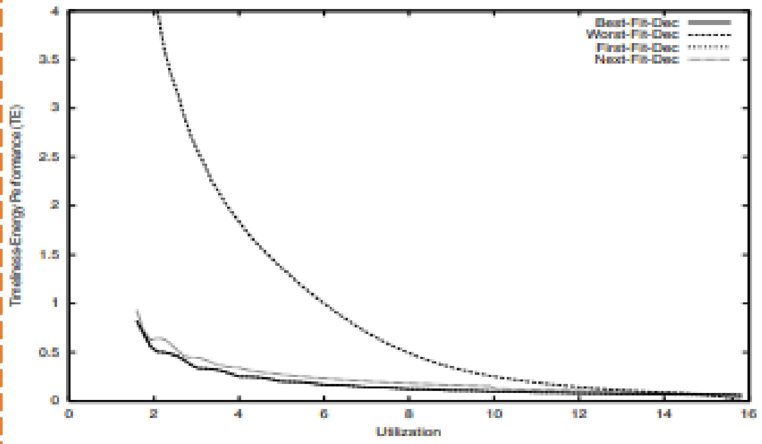

**Figure 8. Timeliness/Energy Performance for $\alpha = 0.5$**

## 5.2 - Performance of Algorithms without Utilization Ordering

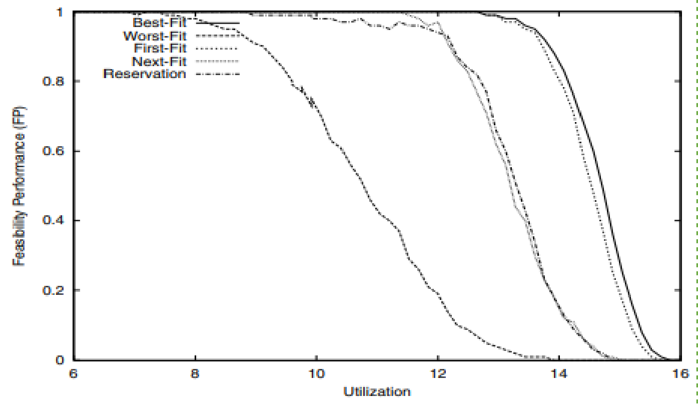

Figure 9. Feasibility Performance for $\alpha = 1.0$

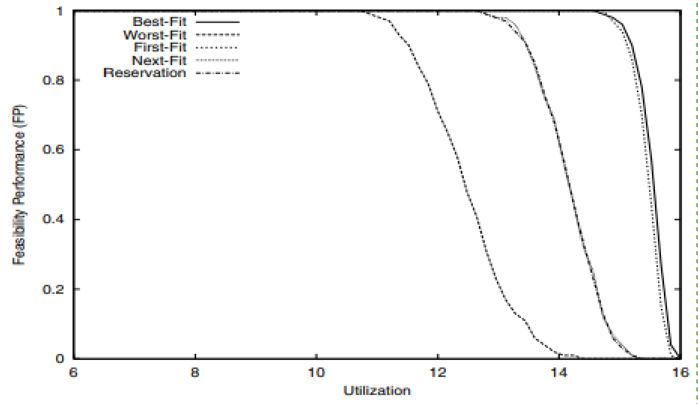

Figure 10. Feasibility Performance for $\alpha = 0.5$

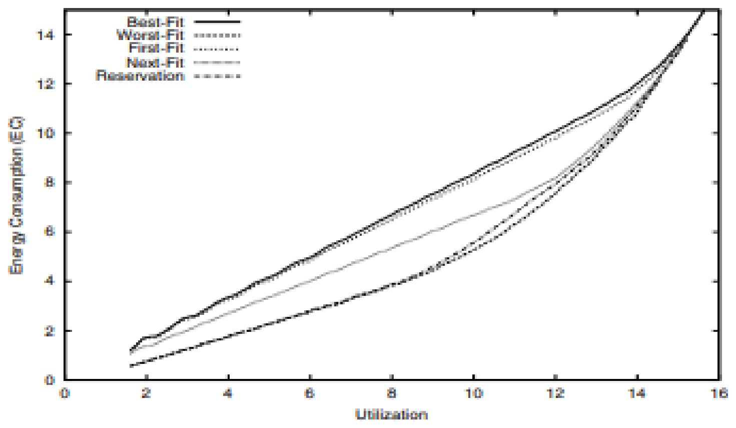

Figure 11. Energy Performance for $\alpha = 1.0$

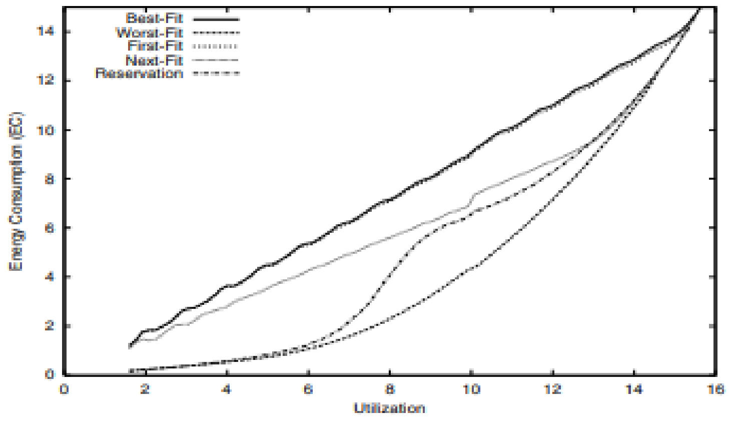

Figure 12. Energy Performance for $\alpha = 0.5$

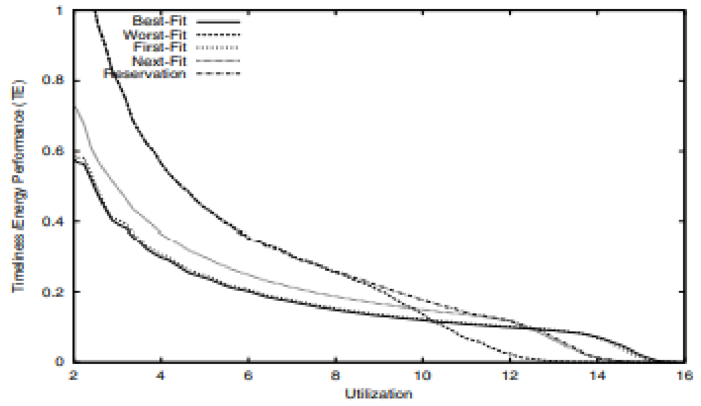

Figure 13. Timeliness/Energy Performance for $\alpha = 1.0$

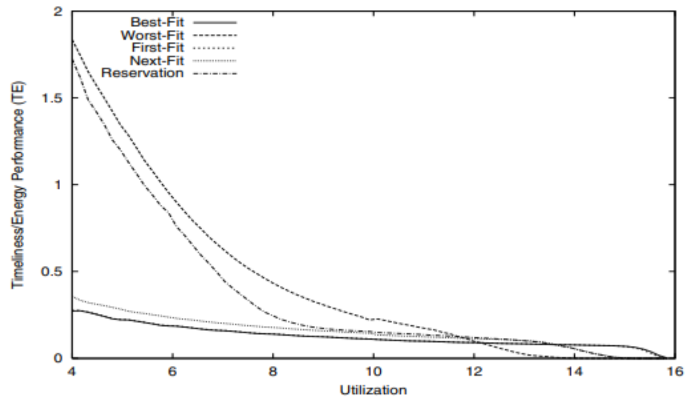

Figure 14. Timeliness/Energy Performance for $\alpha = 0.5$

- To overcome these difficulties, we present an algorithm called RESERVATION.
- The idea of the algorithm is to reserve half (more accurately, m/2 processors) of processor set for “light” tasks, and the other half for “heavy” tasks.
- A light task is defined to be a task with utilization not exceeding A.
- When presented a task $T_i$, the algorithm tries to allocate it to the corresponding subset of processors .
- RESERVATION algorithm is in fact a trade-off between the good feasibility performance of First/Best-Fit algorithms and the good energy performance of Worst-Fit algorithms.

# 6、Conclusion

- We showed that finding the partitioning with minimum energy consumption is NP-Hard in the strong sense, even when the feasibility of the task set is guaranteed a priori.
- Then we developed our load balancing framework, showing that some partitionings are unbalanced in that moving just one task from one processor to another can immediately improve energy savings.
- Our experimental evaluation shows that Worst-Fit-Decreasing heuristic is a clear winner in timeliness/energy performance. However, for the case where the algorithms are not allowed to preorder tasks according to utilizations, we proposed a new algorithm RESERVATION that does not exhibit large variances observed in other heuristics.