Solving Energy-Latency Dilemma: Task Allocation for Parallel Applications in Heterogeneous Embedded Systems

Tao Xie

1. Introduction

In this project, we address the issue of allocating a group of parallel tasks on a heterogeneous embedded system with an objective of energy-saving and short-latency. A novel task allocation strategy, or BEATA (Balanced Energy-Aware Task Allocation), is developed to find an optimal allocation that minimizes overall energy consumption while confining the length of schedule to an ideal range. So far we have completed the following tasks: (1) BEATA strategy design and implementation; (2) experimental environment setup; and (3) preliminary results analysis. In what follows we will explain each of these three completed tasks in detail.

2. Algorithm design and implementation

The BEATA algorithm (see Fig. 1) is conducive to increasing heterogeneous embedded nodes' lifetime while maintaining high performance in terms of makespan time for collaborative applications running on networked embedded systems. In other words, BEATA can increase embedded nodes' lifetimes by dramatically reducing energy dissipation (see Step 13). Before minimizing the energy consumption of task t_i, BEATA organizes all the nodes in a non-decreasing order in terms of t_i's finish time. Step 8 determines the energy consumption incurred by the task on an embedded node, whereas Steps 9-10 calculate the energy consumed by all the messages received by the task from its predecessors. Among all the candidate nodes listed in the energy-adaptive window, Step 13 chooses the most appropriate node that yields the minimal energy dissipation for the task and its corresponding messages, thereby conserving energy without excessive performance deterioration. Then, Step 14 allocates the task to the best candidate node. After the allocation of the task is accomplished, Step 15 updates the schedule of the node to which the task is allocated.

```
1.for each task $t_i \in T$ do
2.    for each node $p_u \in P$ in the system do
3.        Compute $est_u(t_i)$
4.        Compute $f_u(t_i)$ (see Eq. 15)
5.    end for
6.    Sort all nodes in finish time of $t_i$
7.    for each node in energy-adaptive window do
8.        Compute energy consumption of $t_i$
9.        for each $t_i$'s predecessor $t_j$, do
10.           Compute the energy consumption
              cause by message $(t_j, t_i)$
11.           Compute the total energy consumed
              by $t_i$ and the messages sent from
              the predecessors
12.       end for
13.   end for
14.   Select $p_v$ in energy-adaptive window that
      offers the smallest energy consumption for $t_i$
      and messages sent from $t_i$'s predecessors
15.   Assign $t_i$ to $p_v$
16.   Update the schedule on node $p_v$
17.   Compute the energy consumed by $t_i$ on $p_v$
      and the messages received by $t_i$
18.   Record start time and finish time for task $t_i$
19.end for
```

Figure 1. The BEATA strategy.

3. Experimental environment setup

Now we are in a position to evaluate the effectiveness of the proposed energy-latency driven task allocation scheme. To demonstrate the strength of BEATA, we compare it with the list scheduling scheme, which is a well-known scheduler for parallel applications and a baseline scheme GEATA (Greedy Energy-Aware Task Allocation). The two algorithms are briefly described below.

(1) *LIST*: For each task allocation, it chooses the computing node that can offer the task earliest finish time considering both computation time and communication time. Its goal is to generate a schedule for a DAG with the shortest length.

(2) *GEATA*: For each task allocation, it selects the computing node that can provide least energy consumption for the task including computation energy consumption and communication energy consumption. Its goal is to make a schedule with the least total energy consumption.

Before presenting empirical results, we present the simulation model as follows. Table 1 summarizes the configuration parameters of simulated networked embedded systems used in our experiments. The parameters of computing nodes in the networked embedded systems are chosen to resemble real-world processors like Intel StrongARM 1100. The relationship between energy rate and transmission rate is 100 mW at 100 Kbps, which means the time and energy cost for transmitting one bit are around 10 µsec and 1 µJoule. All synthetic parallel jobs used were created by TGFF, a randomized task graph generator.

Although number of tasks, number of computing nodes, out degree, and task execution time are synthetically generated, we examined impacts of these important workload parameters on system performance by controlling the parameters. The performance metrics by which we evaluate system include:

- *Makespan* (the latest task completion time in the task set represented by a DAG).
- *Energy consumption*: total energy consumed by the task set including computation energy consumption and communication energy consumption.
- *Utilization standard deviation (USD)*: standard deviation of computing nodes utilization in the simulated networked embedded systems.
- *Energy standard deviation (PSD)*: standard deviation of computing nodes energy consumption in the simulated networked embedded systems.

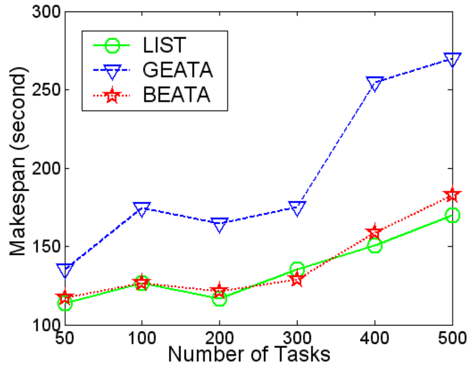
Table 1. System parameters.

Parameter	Value (Fixed)-(Varied)
Number of tasks	(300) – (50, 100, 200, 300, 400, 500)
Energy-adaptive window	(4) – (2, 4, 6, 8, 10, 12, 14, 16)
Number of nodes	(64)
Energy consumption rate heterogeneity	1.2 (see Eq. 16)
Standard node energy consumption rate	200 mW

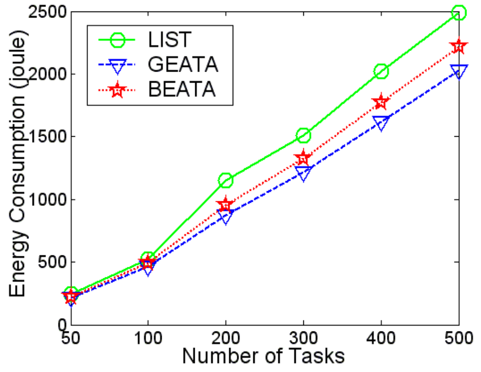
4. Preliminary results analysis

The goal of this experiment is to compare the proposed BEATA algorithm against the conventional list scheduling scheme and a baseline scheme GEATA, and to understand the sensitivity of the two heuristics to the number of tasks in a DAG. We tested 6 task graphs with the number of tasks varying from 50 to 500 with precedence constraints.

We observe from Figure 2a that BEATA and LIST exhibit very similar performance in terms of makespan, whereas BEATA noticeably outperforms the GEATA algorithm. An interesting observation is that BEATA even generates a shorter schedule than LIST when the number of tasks is 300. The "anomaly" can be explained by the fact that the LIST algorithm cannot



(a)



(b)

Figure 2. Performance impacts of the number of tasks.

guarantee the shortest schedule in a heterogeneous system due to lack of the information about tasks not yet scheduled and the varying execution times for each task on different computing nodes. Compared with LIST, BEATA on average only increases makespan by 2.9% but saves energy by 12.1%. Figure 2b reveals that BEATA and GEATA consistently performs better than LIST in terms of energy consumption. In particular, GEATA achieves improvement on averages of 19.3%.

5. Conclusions

In this project, we address the issue of allocating tasks of parallel applications in heterogeneous embedded systems with an objective of energy-saving and latency-reducing. We will conduct a comprehensive simulation study to evaluate the BEATA strategy in both performance and energy-efficiency. In particular, we will conduct experiments to examine the impact of energy-adaptive window, scalability, and heterogeneity on BEATA. Next, to validate the results from the synthetic collaborative applications, we will evaluate the BEATA algorithm using a real system – digital signal processing system.