湖南大學
HUNAN UNIVERSITY

论文阅读报告

报告人：高楠

2018年1月26日

湖南大學
HUNAN UNIVERSITY

2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Enhanced Energy-efficient Scheduling for Parallel Applications in Cloud

Qingjia Huang, Sen Su, Jian Li, Peng Xu, Kai Shuang, and Xiao Huang

State Key Lab of Networking and Switching Technology,
Beijing University of Posts and Telecommunications
{qjhuang, susen, upton, xupeng, shuangk, xiaohuang}@bupt.edu.cn

背景

湖南大學
HUNAN UNIVERSITY

- 在过去的几十年中，IT需求的快速增长促使企业将工作量转移到大型云数据中心。这些数据中心的能源消耗量一直在飞涨。
- 因此，如何降低数据中心的能源消耗一直是业界和学术界关注的重要课题。

- 常用技术手段：松弛回收（slack reclamation）
- 目标：对于异构分布式计算系统中的并行应用，尽量减少任务的能耗，同时还要满足基于性能的服务水平协议(SLA)。

模型

湖南大學
HUNAN UNIVERSITY

系统模型

- 支持DVFS的云数据中心，具有大量异构物理服务器，这些服务器对于运行任何应用程序具有同等的运行能力。
- 所有的处理器可以运行在不同的电压和频率水平。
- *P*：处理器集合
- *V*：电压集合
- *F*：工作频率集合，对于给定的V_i有其对应的F_i
- 当处理器处于空闲状态，它处于最低电压状态V_{min}（$V_{min}>0$）以达到最大限度节能。
- 频率转换的开销很小，不予考虑。

应用模型

- *DAG*: $G = (N, E)$

能量模型

$$P_{dynamic} = k \cdot v^2 \cdot f,$$

$$E_{dynamic} = P_{dynamic} \cdot \Delta t,$$

$$E_{total} = E_{tasks} + E_{idle}.$$

- E_{idle}: v_{lowest}, f_{lowest}

- 调度模型

将任务集合*V*分配到处理器集合*P*上，并为每个任务选择工作电压，目标是在满足截止期限约束的同时减少能耗。

TABLE I
EXECUTION TIME OF TASKS IN FIGURE 1(A)

task	1	2	3	4	5	6
$T_{exec}(s)$	1	1	2	1	4	1

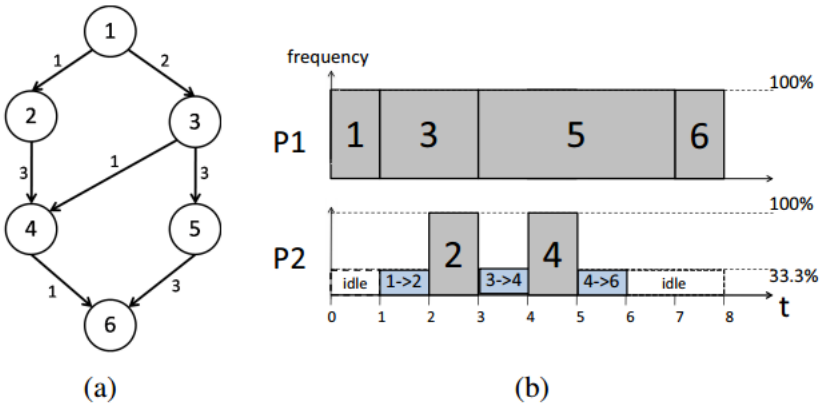


Fig. 1. HEFT initial task assignment without slacking

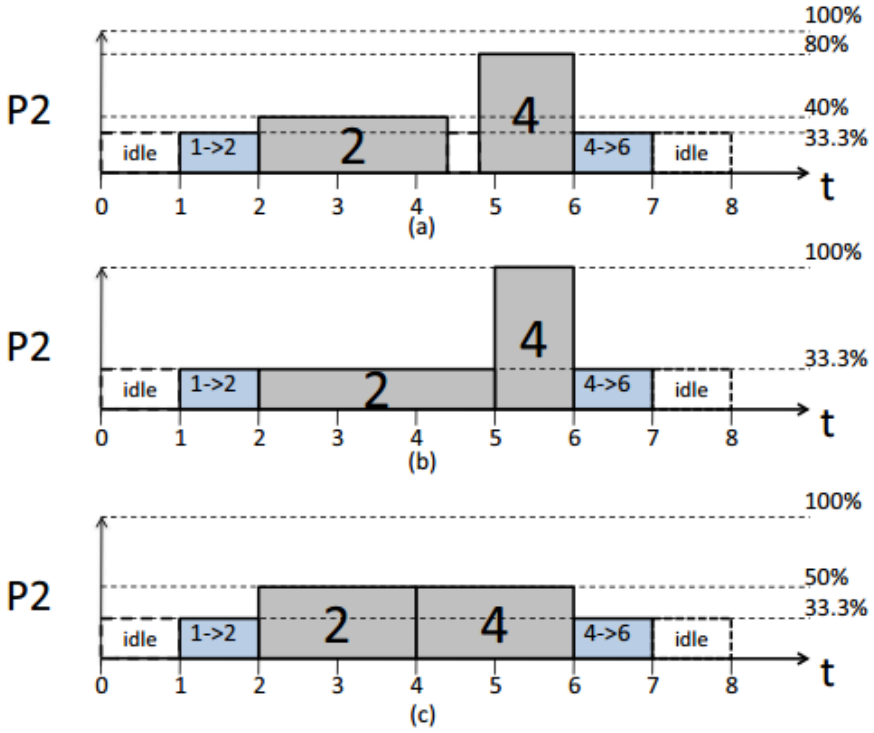


Fig. 2. Slack results by different approaches

- 本文策略：在同一处理器上安排附近的任务以统一的工作频率运行，以最大限度地节约能源。
- 主要想法是缩放非关键路径上面的任务的电源/频率。

算法设计

湖南大學
HUNAN UNIVERSITY

- 阶段1:Task Mapping

采用HEFT算法将任务分配到处理器上作为初始任务调度,调度长度记为M，每个任务的执行时间间隙为[$AST(n_i)$, $AFT(n_i)$]。

- 阶段2:Task Slacking

1) Extend to performance constraint

extension ratio : $\mu = \frac{T_{deadline}}{M}$

所有任务移动到新的执行时间间隙：$[AST'(n_i), AFT'(n_i)]$

其中

$$AFT'(n_i) = \mu \cdot AFT(n_i)$$

$$AST'(n_i) = AFT'(n_i) - T_{exec}(n_i)$$

- 2）Calculate slack time

$$T_{est}(n_i) = \max_{n_j \in pred(n_i)} (T_{est}(n_j) + T_{exec}(n_j))$$
$$T_{lft}(n_i) = \min_{n_j \in succ(n_i)} (T_{lft}(n_j) - T_{exec}(n_j))$$
$$T_{slack}(n_i) = T_{lft}(n_i) - T_{est}(n_i) - T_{exec}(n_i)$$

- 3）Lower the frequency

选择*Tlft*(*ni*)最大的任务，如果*Tslack*(*ni*)>0，在*ni*所在处理器先前重复检测是否 $T_{lft}(n_{p_j,k-1}) > T_{est}(n_{p_j,k})$，直到 $n_{p_j,x}$ 前一任务不存在时间间隙的叠加。所有存在松弛时间叠加的任务构成一个临时的集合：

$$S = \{n_{p_j,k}, n_{p_j,k-1}, \ldots, n_{p_j,x})\}$$

定义 $T_{exec}(S)$ ：S的执行时间之和

$T_{total}(S)$ ：可以分配给S的时间间隙的总和

$$T_{exec}(S) = T_{exec}(n_{p_j,k}) + T_{exec}(n_{p_j,k-1}) + \ldots + T_{exec}(n_{p_j,x})$$

$$T_{total}(S) = T_{lft}(n_{p_j,k}) - T_{est}(n_{p_j,x})$$

- 任务n_i的频率变为：

$$f_{global}(n_{p_j,k}) = f_{\max} \cdot \max\left(\frac{T_{exec}(n_{p_j,k})}{T_{exec}(n_{p_j,k}) + T_{slack}(n_{p_j,k})}, \frac{T_{exec}(S)}{T_{total}(S)}\right)$$

- 相应的执行时间变为：

$$T'_{exec}(n_i) = \frac{n_i}{f_{global}(n_i)}$$

- 分配给n_i的时间间隙变为：

$$[T_{lft}(n_i) - T'_{exec}(n_i), T_{lft}(n_i)].$$

● 4）Update and Repeat

更新任务*ni*的前驱结点和同处理器上前一任务的LFT。
重复第三步直到所有任务都被优化。

实验评估

湖南大學
HUNAN UNIVERSITY

- 三种异构处理器：AMD Turion MT-34, Pentium M ， AMD Athlon-64 ，一共24个，每种各8个。
- DAG模型：随机生成模型（结点数为20、40、60、80、100、200）和一种真实的并行应用（Gaussian Elimination算法，任务数选择在20到209之间）
- CCR（communication to computation ratio）设置为0.3。

● 对比指标

E_{base}：采用HEFT算法所产生的能耗作为基线。

M_{base}：采用HEFT算法的调度长度作为执行时间的基线。

所有算法的性能都是以归一化总消耗量来衡量的。

the extension rate γ：$T_{deadline} = M \cdot (1 + \gamma)$

γ设置为200%以内。

实验结果

湖南大學
HUNAN UNIVERSITY

- 1） Without Extension

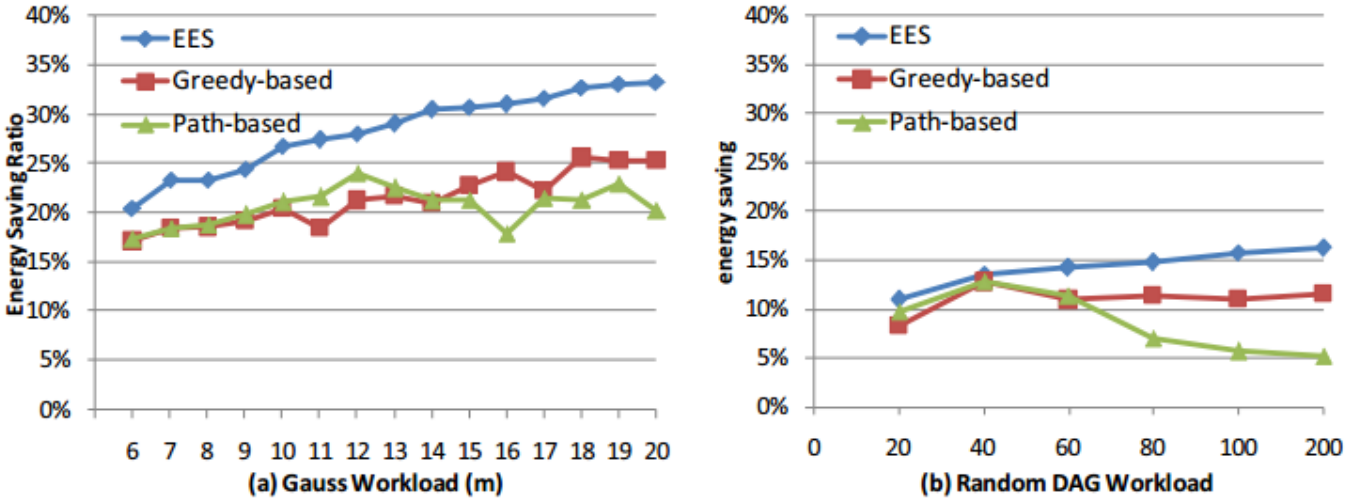


Fig. 3. Without extension

本文中算法在不超过截止时限的情况下可以实现相当可观的节能，节能率明显高于其他两个算法，且随着负载规模的增加而增加。

- 2）With Extension

基准工作负载：200结点的随机DAG和209结点的Gauss DAG 。

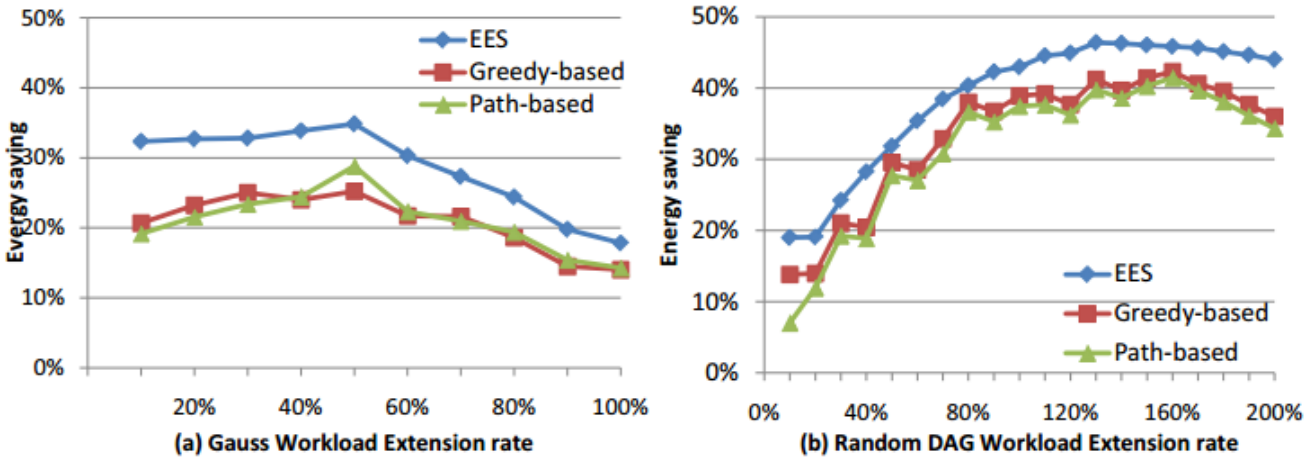


Fig. 4. With extension

由于DAG模型的特点，以及负载的松弛空间总是有限的，因此在降低功耗方面有一定的上限。

- 3）与ECS算法比较

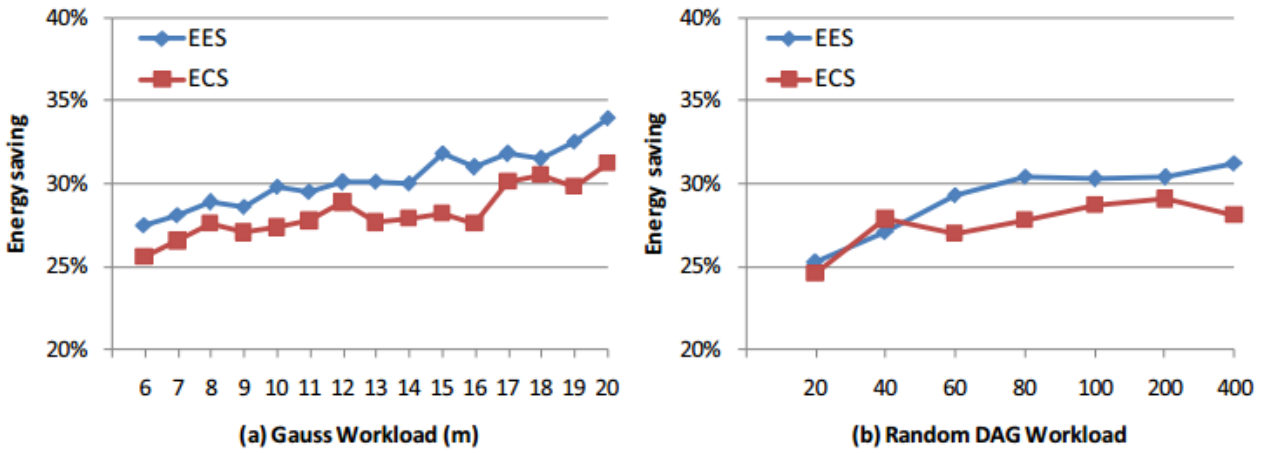


Fig. 5. Using ECS makespan as deadline

ECS算法在大多数情况下都能节省大量的能源，但是并没有受到硬性期限的限制。EES算法不仅可以减少更多的能耗，而且还可以灵活地满足之前定义的不同性能需求。

湖南大學
HUNAN UNIVERSITY

An Energy-Efficient Task Scheduling Algorithm in DVFS-enabled Cloud Environment

Zhuo Tang · Ling Qi · Zhenzhen Cheng · Kenli Li · Samee U. Khan · Keqin Li

- Journal of Grid Computing, 2016 , 14 (1) :55-74

- 本文提出一种能量感知任务调度算法：DEWTS（DVFS-enabled Energy-efficient Workflow Task Scheduling algorithm）

- 该算法在给定的截止时限内，可以将并行应用程序分配到适当的处理器上，并在适当的时间间隙中处理它们，以减少能耗，并满足所需的性能。

DEWTS算法

湖南大學
HUNAN UNIVERSITY

三个主要步骤：

1） initial task mapping phase

2） processors merging phase

3） task drawing phase

1）Initial Task Mapping Phase

- 采用HEFT算法完成初始调度
- 调度长度： $D = MS \times (1 + \alpha)$
- 截止时限

2）Processors Merging Phase

- 计算处理器的*rankm*值， *rankm*值等于处理器上分配的任务数。
- 根据*rankm*降序排序对处理器进行排序，当*rankm*值相同时，能量利用率（*peu*）较低的处理器放在后面，其中

$$p_{j_{eu}} = \frac{\left(\sum_{i=1}^{rank_m(p_j)} w_{i,j}\right) \times p_{j_{max}}}{\left(\sum_{i=1}^{rank_m(p_j)} w_{i,j}\right) \times p_{j_{max}} + \left(MS - \sum_{i=1}^{rank_m(p_j)} w_{i,j}\right) \times p_{j_{idle}}}$$

得到处理器序列{*p1*,*p2*,…*pk*}

- 在前k-1个处理器上采用HEFT算法进行任务调度，如果调度长度没有超过截止时限，则关闭最后一个没有参与的处理器成为idle状态，k值减1，重复这个操作直到调度长度超过截止时限，没有关闭的处理器集合记为*P*’。

● 3）Time Slacking Phase

湖南大學
HUNAN UNIVERSITY

Algorithm 4 *T_Slacking*()

Input:

Task slacking by using DVFS technique;
The scheduling results of Algorithm 3: MS', D, EST, every task's execution time.

Output:

A new executing tasks list with proper voltage/frequency pairs.

1: Calculate every task's LFT;
2: Calculate every task's slacking time $Slack(n_i)$;
3: **while** there are un-optimized tasks **do**
4: **if** The slacking time of task n_i $Slack(n_i)$ exist **then**
5: Select the task n_i with the largest LFT;
6: Calculate n_i 's ideal operating frequency f'_{n_i,p_j};
7: Pick out n_i 's actual operating frequency f_{n_i,p_j};
8: Update the frequency of task n_i from $f_{p_j max}$ to f_{n_i,p_j};
9: Update the execution time of task n_i to t'_i;
10: Assign time slot of task n_i to $[LFT(n_i)-t'_i, LFT(n_i)]$;
11: Update the LFT of all predecessor task for task n_i;
12: Update the LFT of the task n_x which be executed just before task n_i on p_j;
13: **end if**
14: **end while**
15: **return** an executing tasks list.

$$LFT(n_i)=\begin{cases} D, & if\, n_i = n_{exit} \\ min_{n_j \in succ(n_i)} \left(LFT(n_j) - t_j - c_{i,j}\right), & otherwise \end{cases}$$

实验与分析

湖南大學
HUNAN UNIVERSITY

- 对比算法：HEFT和EES
- 实验平台：CloudSim simulator
  CloudSim是一个广泛使用的建模框架。它可以用来模拟云计算基础设施和服务，可以提供可重复和可控的实验环境，不需要过多关注硬件细节。

● 性能指标：

1）ECR（Energy Consumption Ratio）

DAG中任务的总能耗与在关键路径上执行速度最快的处理器中执行任务的能耗的比值。

2）系统资源利用率（System Resource Utilization Ratio）

使用资源与总资源的比。

3）平均执行时间（Average Execution Time）

4）节能率（ Energy Saving Ratio）

与*Ebase*相比所节约的能耗。

● 实验设置

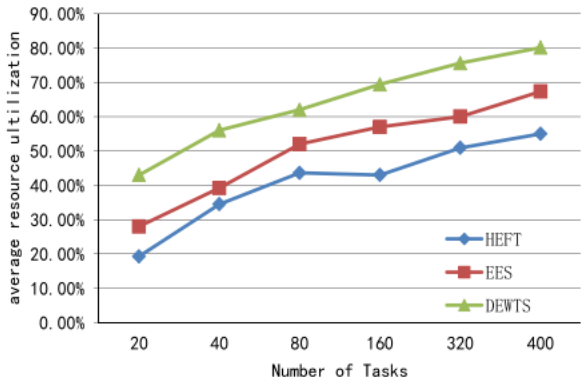
采用随机生成的DAG，仿真参数如下：

- The number of random DAG tasks: {20, 40, 80, 160, 320, 400}.
- The set of parallelism factor β: {0.2, 0.5, 1.0, 2.0, 5.0}. A high β will lead to a DAG with shorter length but high parallelism.
- The communication to computation ratio (CCR) set: {0.1, 0.5, 1.0, 2.0, 5.0}. if CCR value is very low, it can be considered as a computation-intensive application, otherwise, it can be considered as communication-intensive application.
- The set of processors available to use is from 2 to 32, incrementing by the power of 2.
- The extension ratio α in our experiments ranges from 0 to 180 %.

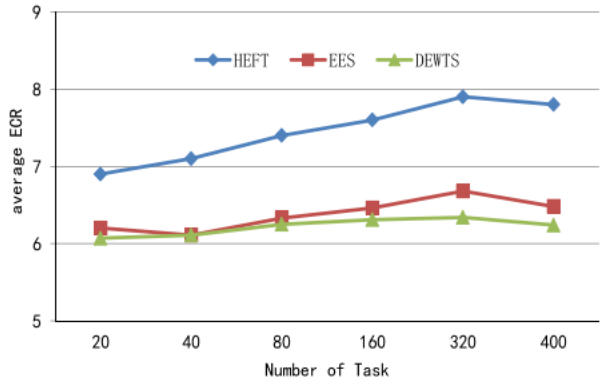
实验结果与分析

湖南大學 HUNAN UNIVERSITY

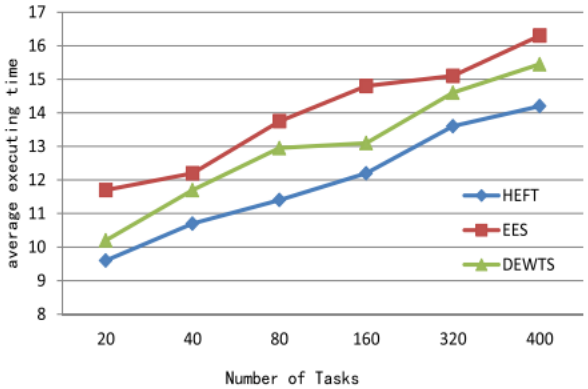
- 实验一：在最大性能条件下的评估



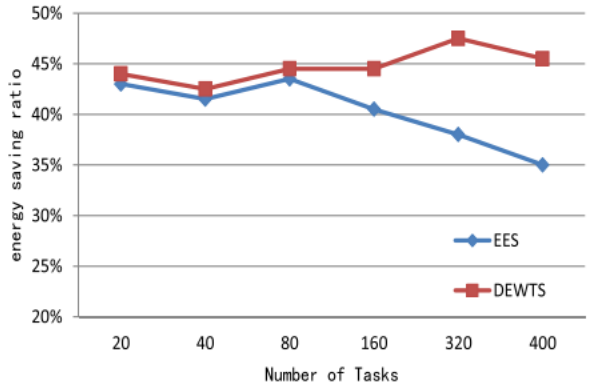
(a) Average resource utilization.



(b) Average ECR.



(c) Average running time.



(d) The energy saving ratio.

Fig. 5 Evaluation of random DAG without extension schedule length

- extension ratio $\alpha = 0$ 。
- 处理器数量为32，每种处理器8个。

结论：在最高性能条件下，DEWTS和EES算法在工作集相对较大时显示出良好的节能效果，DEWTS的能量优化效率略显明显。

● 实验二：处理器数量不同情况下的评估

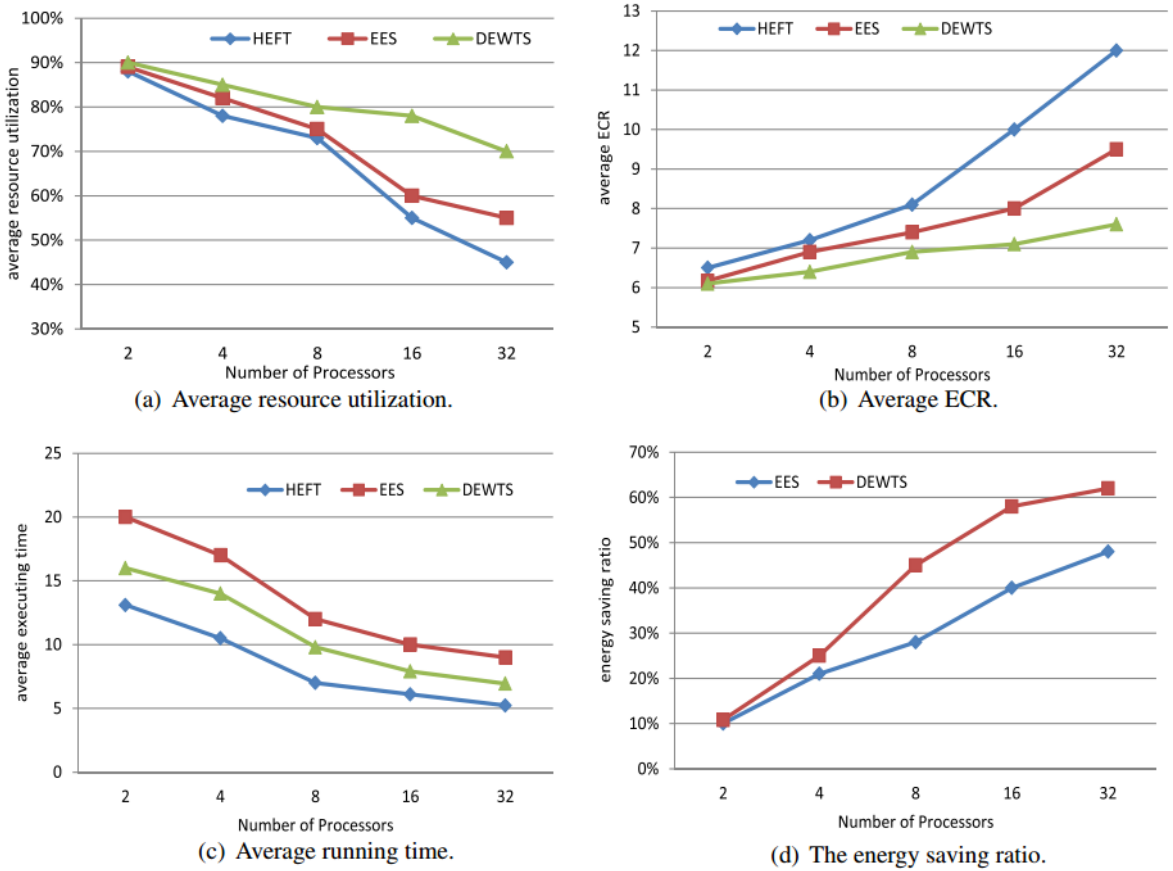
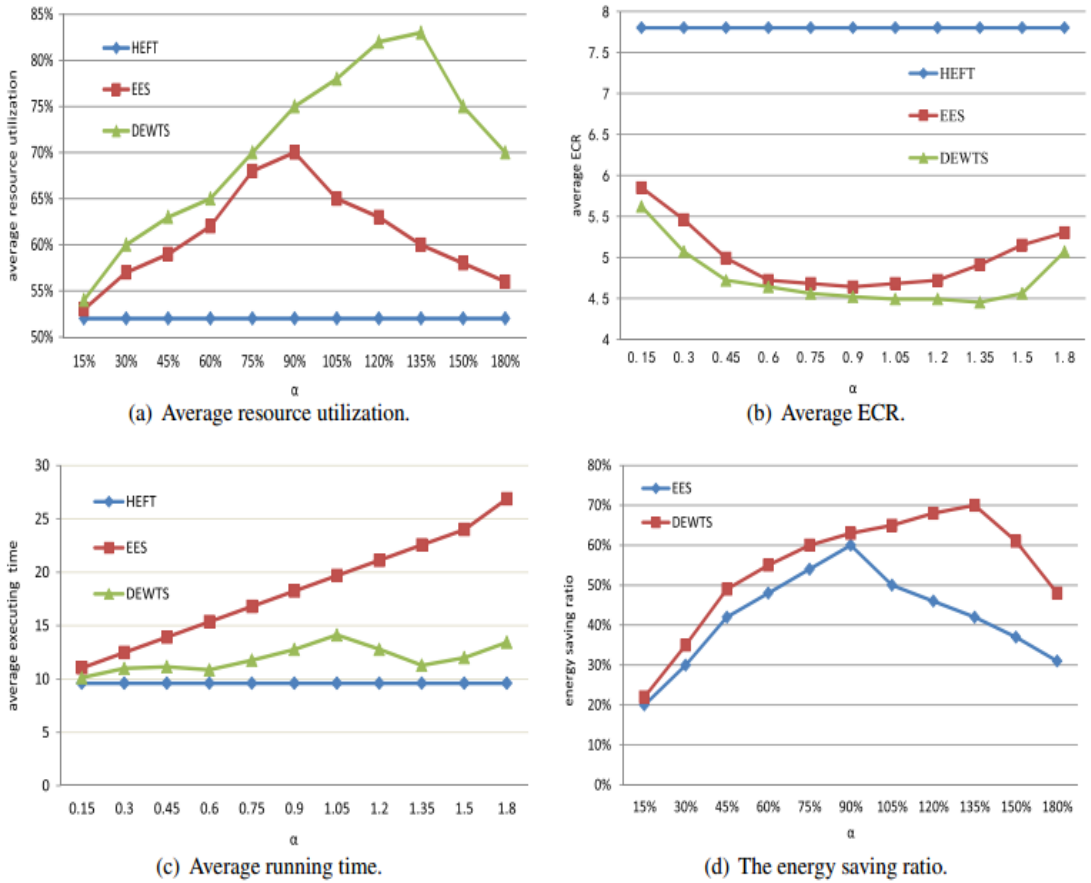


Fig. 6 Evaluation of random DAG with different number of processors

- 处理器数量从2变化到32，任务数固定为400，其他参数与实验一相同。
- 增加处理器的数量可以减少运行时间，但会带来资源浪费。因此，在实践中，不能盲目地增加处理器的数量。在这个实验中，8个处理器是最好的选择。

● 实验三：不同Extension Ratios下的评估



(a) Average resource utilization.

(b) Average ECR.

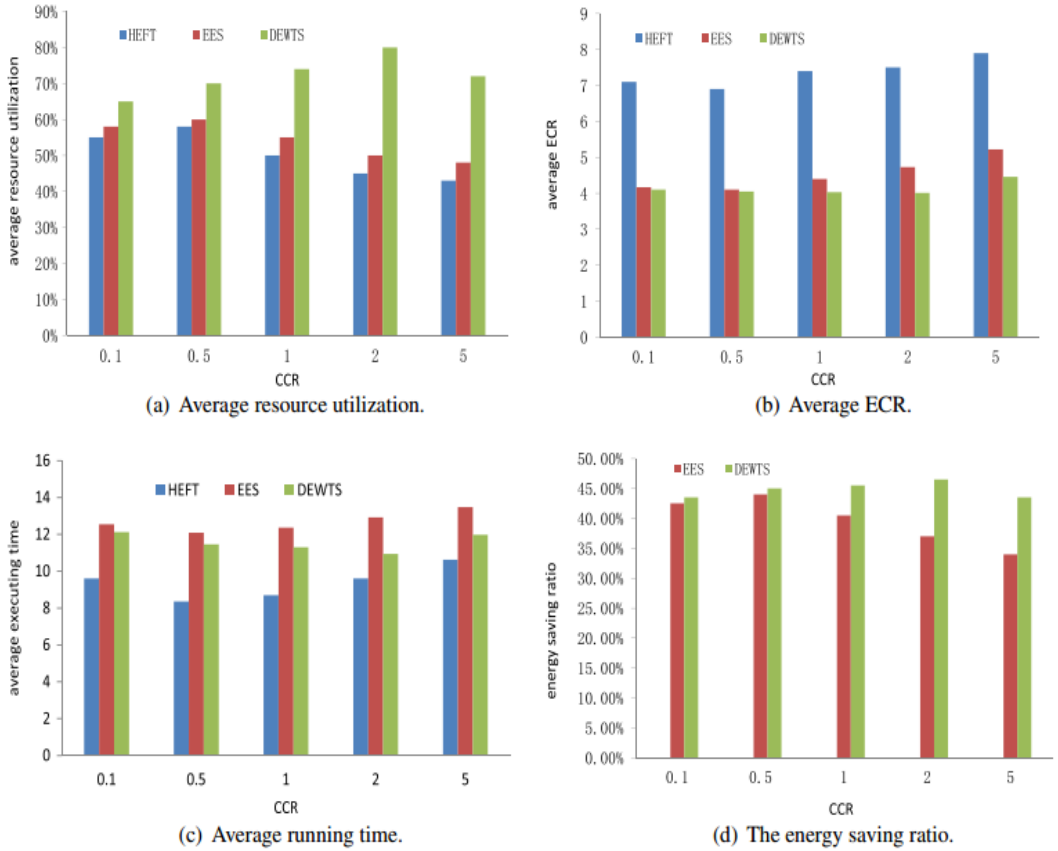
(c) Average running time.

(d) The energy saving ratio.

Fig. 7 Evaluation of random DAG with different extension schedule length

- 任务数为400，处理器数为48.
- （a）:适当牺牲调度长度，资源的利用将大大提高。
- （b）:DEWTS算法中空闲时隙被更充分利用。
- （c）:EES算法为了节能牺牲太多性能， DEWTS在性能和减少能量耗散之间做出了更好的权衡。
- 结论：性能的四个特性随着α的提高而提高直到α超过一个阈值。由于DAG模型的特点和工作负载的松弛空间都是有限的，因此在降低功耗方面存在一定的上限。

● 实验四：不同CCR值下的评估



(a) Average resource utilization.

(b) Average ECR.

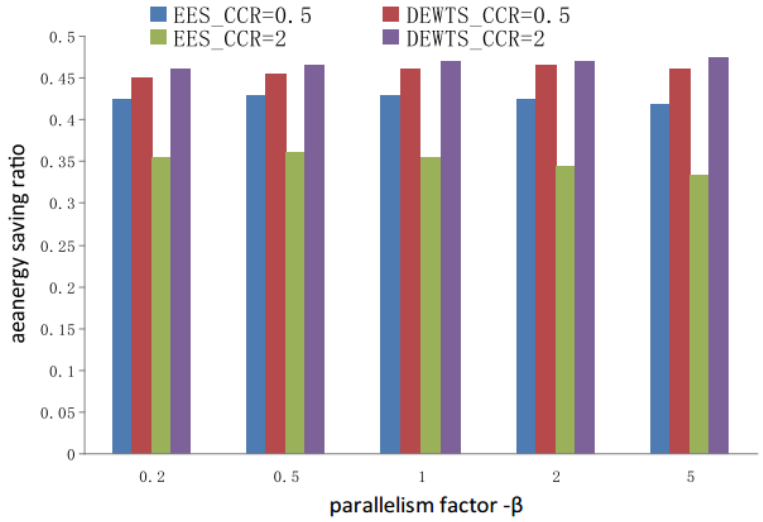
(c) Average running time.

(d) The energy saving ratio.

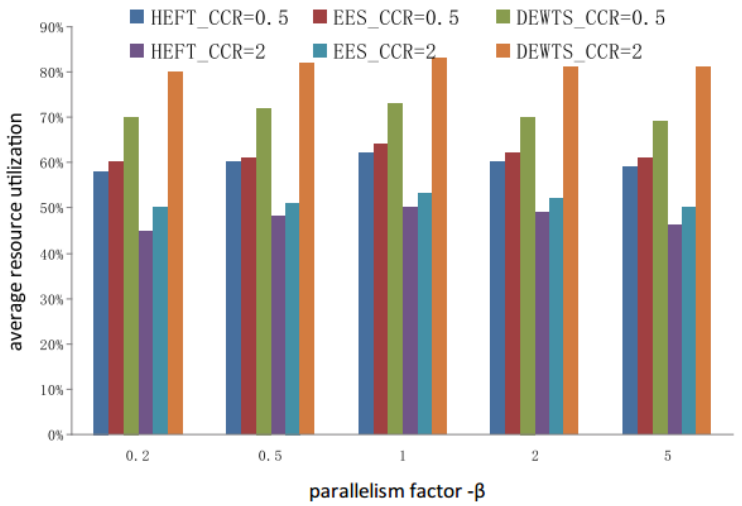
Fig. 8 Estimate in Different CCR

- α = 1.0，任务数为400，处理器数为48。
- （a）:EES和HEFT为计算密集型应用， DEWTS同时适用计算密集和通信密集型应用。
- （b）:DEWTS中CCR对ECR的影响较小。
- （c）: DEWTS中减少处理器的使用带来能耗的减少。
- 结论：同（a）。

- 实验五:不同程度的并行性下的评估



(a) Average resource utilization.



(b) Average ECR.

Fig. 9 Estimate in Different parallelism factor

并行性因素对能耗几乎没有影响,但是当β= 5,CCR = 2 情况下DEWTS可以实现更高的节能率,因为在这种情况下,有更有效的空闲时隙可以让DEWTS通过合并处理器的数量来提高资源利用率。因此，它可以实现更高的资源利用率，节省更多的能耗。

- 实验结论

在DEWTS中，在不违法优先级约束的前提下，运行处理器的数量比其他比较算法要少。同时，在较低的处理器电压下，任务的执行时间被slacked，以减少能耗。在所有这些实验的基础上，通过不同类型的DAG任务集，DEWTS可以满足用户约束的截止期限，不仅保持了良好的性能，而且降低了空闲成本和浪费的能耗。

● 收获总结

可有效降低能耗的技术手段有：

1） DVFS-enabled

2）关闭一定数量处理器

3）全局优化调度， slack reclamation

湖南大學
HUNAN UNIVERSITY

Thanks!