周报告

汇报人： 袁 娜

2017.03.16

■满足可靠性目标的资源最小化研究

➢问题

•汽车嵌入式系统处理器性能大幅度提升的同时，系统故障发生率不断增加；汽车行驶过程中电磁干扰、消息丢失等容易造成ECU和通信网络出现瞬时故障。任务的可靠性指系统中任务无故障运行的概率。

•嵌入式系统资源往往是有限的，因而需要在设计阶段对资源消耗进行优化。

■满足可靠性目标的资源最小化研究

➢可靠性模型

ECU发生故障的概率服从泊松分布，单位时间t内的可靠性$R(t)=e^{-\lambda t}$，λ为单位时间内的恒定故障率。

令λ_k为处理器u_k的恒定故障率，则任务n_i在u_k上的可靠性为

$$R(n_i, u_k) = e^{-\lambda_k w_{i,k}}$$

任务间存在优先级约束的并行应用的可靠性为任务可靠性之积

$$R(G) = \prod_{n_i \in N} R(n_i, u_{proc(n_i)})$$

■满足可靠性目标的资源最小化研究

➢资源成本模型

定义计算资源成本率γ，执行时间w，则计算资源成本为γ*w
定义通信成本率γ_{comm}，通信时间$c_{x,i}$，则通信资源成本为$\sum_{n_x \in pred(n_i)} c'_{x,i} \times \gamma_{comm}$

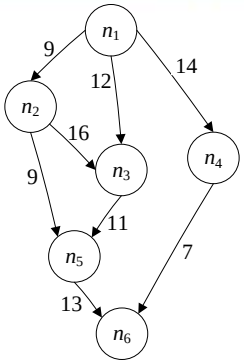
任务n_i在处理器u_k上的资源消耗为

$$cost(n_i, u_k) = w_{i,k} \times \gamma_k + \sum_{n_x \in pred(n_i)} c'_{x,i} \times \gamma_{comm}$$

并行的资源消耗为任务资源之和

$$cost(G) = \sum_{n_i \in N} cost\left(n_i, u_{proc(i)}\right)$$

■满足可靠性目标的资源最小化研究



➢调度策略比较

该调度的关键在于如何将功能的可靠性目标$R_{goal}(G)$转换成任务的可靠性目标。

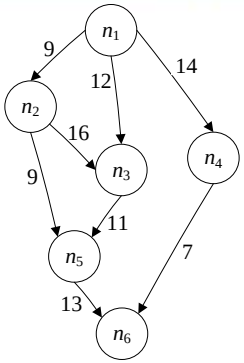
①L. Zhao, Y. Ren等人提出的动态副本数量的容错调度算法MaxRe算法提出将任务可靠性目标设为平均值：

$$R_{\text{goal}}(n_1) = \sqrt[|N|]{R_{\text{goal}}(G)}$$

②由于任务过高的可靠性目标需要花费更多的资源来满足，此后L. Zhao, Y. Ren等人在其可靠工作流调度算法RR算法中对可靠性目标加以改进：

$$R_{\text{goal}}(n_{seq(j)}) = \sqrt[|N|-j+1]{\frac{R_{\text{goal}}(G)}{\prod_{x=1}^{j-1} R(n_x)}}.$$

■满足可靠性目标的资源最小化研究



➢调度策略比较

该调度的关键在于如何将功能的可靠性目标$R_{goal}(G)$转换成任务的可靠性目标。

③RR算法可靠性目标中，未调度的任务的目标仍有可优化的空间

$$R_{seq(j)}(G)=\prod_{x=1}^{j-1}R\left(n_{seq(x)},u_{proc(seq(x))}\right)$$

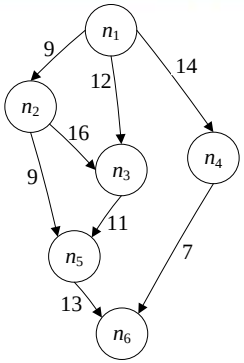
$$\times R\left(n_{seq(j)},u_{proc(seq(j))}\right)\times\prod_{y=j+1}^{|N|}R_{\max}\left(n_{seq(y)}\right)\geqslant R_{\text{goal}}(G)$$

未调度任务以其在多个处理器上的最大可靠性计算

$$R(n_{seq(j)},u_k)\geqslant R_{\text{goal}}(G)\Big/\left(\prod_{x=1}^{j-1}R(n_{seq(x)},u_{proc(seq(x))})\times\prod_{y=j+1}^{|N|}R_{\max}(n_{seq(y)})\right)$$

调度算法：在筛选满足任务可靠性目标的处理器后，将任务调度至资源消耗最小的处理器上。

■满足可靠性目标的资源最小化研究



➢调度策略比较

该调度的关键在于如何将功能的可靠性目标$R_{goal}(G)$转换成任务的可靠性目标。

③RR算法可靠性目标中，未调度的任务的目标仍有可优化的空间

$$R_{seq(j)}(G) = \prod_{x=1}^{j-1} R\left(n_{seq(x)}, u_{proc(seq(x))}\right)$$

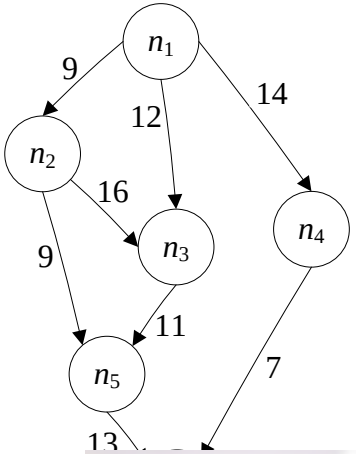
$$\times R\left(n_{seq(j)}, u_{proc(seq(j))}\right) \times \prod_{y=j+1}^{|N|} R_{\max}\left(n_{seq(y)}\right) \geqslant R_{\text{goal}}(G)$$

$$R(n_{seq(j)}, u_k) \geqslant R_{\text{goal}}(G) \Big/ \left(\prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times \prod_{y=j+1}^{|N|} R_{\max}(n_{seq(y)})\right)$$

优化想法：寻求$\sqrt[|N|]{R_{goal}(G)}$与$R_{max}(n_{seq(y)})$之间的合理值

算法演算与问题：
该方法能达到要求，但可靠性目标还有优化空间；
尝试以MaxRe算法中的任务可靠性目标计算方法 $R_{\text{goal}}(n_1) = \sqrt[|N|]{R_{\text{goal}}(G)}$ 计算未调度任务的可靠性，发现可靠性目标越界。

■HEFT算法还原



Task	u_1	u_2	u_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16

①任务优先级排序——向上排序值

$$\overline{c_{i,k}} = \overline{L} + \frac{data_{i,k}}{\overline{B}}$$

$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} (\overline{c_{i,j}} + rank_u(n_j))$$

```
int main()
{
    int i,j;
    FILE *fp;
    fp=fopen("Input.txt","r+");
//文件中读取计算开销、数据传输率、数据传输量
    for(i=0; i<no_tasks; i++)
        for(j=0; j<no_machines; j++)
            fscanf(fp,"%lf",&computation_costs[i][j]);
    for(i=0; i<no_machines; i++)
        for(j=0; j<no_machines; j++)
            fscanf(fp,"%lf",&data_transfer_rate[i][j]);
    for(i=0; i<no_tasks; i++)
        for(j=0; j<no_tasks; j++)
            fscanf(fp,"%lf",&data[i][j]);
    //向上排序值计算
    for(i=0; i<no_tasks; i++)
    {
        if(tasks_upper_rank[i]==-1)
        {
            tasks_upper_rank[i]=calculate_upper_rank(i);
            insertinto(i,tasks_upper_rank[i]);
        }
    }
    printf("向上排序值rank:\n");
    for(i=0; i<no_tasks; i++)
        printf("任务%d : %.2lf\n",i,tasks_upper_rank[i]);
    for(i=0; i<no_tasks; i++)
        printf("任务%d :  %.2lf\n",sorted_tasks[i],tasks_upper_rank[sorted_tasks[i]]);
```

```
double calculate_upper_rank(int task)
{
    int j;
    double avg_communication_cost,successor,avg=0.0,max=0,rank_successor;
    //平均计算开销
    for(j=0; j<no_machines; j++)
        avg+=computation_costs[task][j];
    avg/=no_machines;

    for(j=0; j<no_tasks; j++)
    {
        if(data[task][j]!=-1)
        {
            avg_communication_cost=avg_communicationcost(task,j);//平均通信开销
            if(tasks_upper_rank[j]==-1)//j任务尚未计算rank值，则递归计算出
            {
                rank_successor= tasks_upper_rank[j]= calculate_upper_rank(j);
                insertinto(j,rank_successor);
            }
            else
                rank_successor= tasks_upper_rank[j];//已计算出后驱集的rank值，则直接使用

            successor=avg_communication_cost+rank_successor;

            if(a<successor)
                a=successor;
        }
    }
    return(avg+max);
}
```

$+ c_{m,i})\}$

```
C:\Users\Administrator.GOS-01610221041\D
向上排序值rank:
任务0 : 108.00
任务1 : 77.00
任务2 : 80.00
任务3 : 80.00
任务4 : 69.00
任务5 : 63.33
任务6 : 42.67
任务7 : 35.67
任务8 : 44.33
任务9 : 14.67
任务0 :  108.00
任务3 :  80.00
任务2 :  80.00
任务1 :  77.00
任务4 :  69.00
任务5 :  63.33
任务8 :  44.33
任务6 :  42.67
任务7 :  35.67
任务9 :  14.67
```

下周计划

- 阅读文献，寻找不同的双目标优化问题（Bi-criteria optima or Pareto optima problem）解决策略
- HEFT算法实现