## گزارش فاز اول پروژه درس طراحی مدارهای دیجیتال کمتوان

مهدی بهرامیان - ۴۰۱۱۷۱۵۹۳

عنوان مقاله : TMDS:Temperature aware Makespan Minimizing DAG Scheduler for Heterogeneous Distributed Systems

این مقاله یک الگوریتم زمانبندی مبتنی بر گراف تسک و با توجه به دما، روی پردازنده های ناهمگن ارائه میدهد. نکته تمایز این تحقیق از مقالات دیگر یکی همین ناهمگن بودن پردازنده هاست و دیگری توجه به دما و ویژگی های حرارتی پردازنده علاوه بر اهمیت زمان بر وظایف بیدرنگ است.

این مقاله با تعریف نحوه مدلسازی مساله شروع میکند:

- pow\_t : توان مصرفی زمان
- pow\_d : توان مصرفی یویا زمان t
- pow\_s : توان مصرفی ایستا زمان t
- $pow_t = pow_d + pow_s = C0 f^3 + (C1 f + C2 f T_t)$ 
  - $A = a(C0 f^3 + C1 f), B = (b aC2 f)$  •
- (T\_q = A/B + (T0 A/B)e^(-B(q-0)): رابطه دما وقتی وظیفه بر روی پردازنده درحال اجراست

سپس هدف این بهینه سازی به صورت کمینه کردن طول زمان کل این زمانبندی به طوری که شرایط حرارتی حفظ شود تعریف میشود.

راهی که این مقاله برای این مساله ارائه میدهد به صورت ترکیب ۲ تابع TMDS(G, P) و TETT(t, p, temp, T\_est[t,p] است که اولی الگوریتم اصلی زمانبندی و دومی یک الگوریتم کمکی برای بررسی نتیجه زمانبندی وظیفه t روی پردازنده p در دمای temp است.

این دو الگوریتم را در زیر مشاهده میکنید :

## **Algorithm 1:** TMDS(G, P)

**Input:** Application DAG G(V, E), processor set P

Output: A temperature-aware schedule that minimizes makespan

- $\forall p_n \in P$ , Set avail[n] = 0;
- 2 Determine OFTs, Ranks of all tasks using equations 17, 18;
- 3 Initialize ready list  $taskList = \{\tau_{entry}\};$
- 4 // Let  $\Gamma_n^{cur}$  be a temporary variable which holds the temperature of processor  $p_n$  at any time during partial schedule generation
- 5 **while**  $taskList \neq \phi$  **do**

8

10

 $\tau_i = \text{Extract the task with highest } Rank \text{ from ready list } taskList;$ 

**for** each processor  $p_n$  in P **do** 

Compute  $EST[\tau_j, p_n]$  using equation 19; // Start time of  $\tau_j$  if selected for execution on  $p_n$ 

 $\Gamma_{j,n}^{est} = T_{cool}(\Gamma^a, \Gamma_n^{cur}, EST[\tau_j, p_n] - avail[n]);$  // Tentatively determine temperature of  $p_n$  at the instant

 $EST[\tau_j, p_n]$   $(\Gamma_{j,n}^{est})$ , given: (i) its temperature at avail[n]  $(\Gamma_n^{cur})$ , and (ii) that  $p_n$  idles in the interval [avail[n],

 $EST[\tau_j, p_n]] \langle EFT[\tau_j, p_n], Sch_{j,n}, \Gamma_n^{fin} \rangle = TETT(\tau_j, p_n, EST[\tau_j, p_n], \Gamma_{j,n}^{est}); // The TETT() function returns: (i) effective finish time <math>(EFT[\tau_j, p_n])$  of  $\tau_j$  on  $p_n$ , (ii) Schedule  $(Sch_{j,n})$  of execution and cooling intervals w.r.t. task  $\tau_j$  on  $p_n$  and (iii) temperature of  $p_n$  at  $EFT[\tau_j, p_n]$ 

 $SCT[\tau_j, p_n] = EFT[\tau_j, p_n] + OFT[\tau_j, p_n] - \omega_{j,n}$ ; // Estimate schedule completion time of  $\tau_j$  on  $p_n$ 

Determine:  $p_{n'} = \min_{p_r \in P} SCT[\tau_j, p_r]$ ; // Find the processor  $p_{n'}$  for which  $SCT[\tau_j, p_{n'}]$  is minimum

12 Assign  $\tau_i$  on processor  $p_{n'}$ ;

Set  $AFT[\tau_j] = avail[n'] = EFT[\tau_j, p_{n'}], \Gamma_{n'}^{cur} = \Gamma_{n'}^{fin}; // Update: (i) actual finish time of <math>\tau_j$  (AFT[ $\tau_j$ ]), (ii) avail[n'] and (iii) temperature  $\Gamma_{n'}^{cur}$  of  $p'_n$  at avail[n']

Insert all ready to execute successor tasks of  $\tau_i$  in *taskList*;

```
Algorithm 2: TETT(\tau_j, p_n, curT, \Gamma_{j,n}^{est})
   Input: Task \tau_j, processor p_n, time curT, temperature of p_n at EFT[\tau_j, p_n] (\Gamma_{i,n}^{est})
   Output: Schedule Sch and effective finish time EFT of a task \tau_j on p_n and temperature of p_n at EFT
 1 allowET = MDE(\tau_j, p_n, \Gamma_{j,n}^{est}) refer equation 22;
 2 if \omega_{j,n} \leq allowET then
        return \langle Sch_{j,n}, curT + \omega_{j,n}, T_{heat}(\Gamma_{j,n}^{ss}, \Gamma^0, \omega_{j,n}) \rangle;
 4 else
         remET = \omega_{j,n} - allowET;
        curT = curT + allowET;
7 idleT = coolT(\Gamma^{c}_{j,n}, \Gamma^{lim}_{n}) refer equation 23;
 s allowET = MDE(\tau_j, p_n, \Gamma_{j,n}^c) refer equation 22;
   while remET > 0 do
         if remET ≥ allowET then
              // p_n should idle for an interval [curT, curT + idleT + 2 \times OV]
              curT = curT + idleT + allowET + 2 \times OV;
12
           // \tau_j should execute for an interval [curT – allowET,curT]
13
         else
14
              Calculate \Gamma_{i,n}^{cur} (using equation 26), the max temperature necessary to continuously execute \tau_j for its last
15
                remaining remET time units, while not breaching \Gamma_n^{lim};
              idleT = coolT(\Gamma_{j,n}^{cur}, \Gamma_{n}^{lim}) refer equation 23;
16
              curT = curT + idelT + remET + 2 \times OV;
17
        remET = remET - allowET;
19 return \langle curT, Sch_{j,n}, \Gamma_n^{lim} \rangle;
```

در نهایت برای آزمون الگوریتم خود، این زمانبندی را روی دو الگوریتم Gaussian Elimination و Laplace آزموده اند که گراف وظایفشان به شکل زیر است :

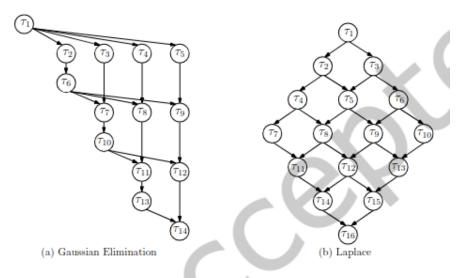


Fig. 5. Benchmark Task Graphs.

و نتایج آزمایششان در مقابل مقالات گذشته THEFT, TPEFT, TPPTS و TPSL به صورت زیر است :

		TODS	TPSLS	TPPTS	TPEFT	THEFT	
TMDS	better	67.8%	66.7%	62.2%	54.0%	65.6%	
	equal	1.3%	6.5%	1.3%	14.2%	1.3%	
	worse	30.9%	26.8%	36.5%	31.8%	33.1%	
THEFT		ı		40.5%			
	equal	1.5%	1.1%	1.7%	1.6%		
	worse	41.2%	55.2%	57.8%	59.1%		
TPEFT	better	64.1%	56.3%	54.9%			4
	equal	2.1%	12.5%	1.8%			
	worse	33.8%	31.2%	43.3%			
TPPTS	better	60.3%	52.1%				
	equal	0.8%	1.4%			4	
	worse	38.9%	46.5%				
TPSLS	better	57.7%					
	equal	1.5%		1			
	worse	40.8%					٦