Bounds on Multiprocessing Timing Anomalies

R. L. Graham

Presented by

Dakshina Dasari

Outline of the presentation

- Introduction
- System model
- Examples of Anomalies in multiprocessors
- Bounds for some cases
- Conclusion

Introduction

- More resources to increase speed of processing : Employ multiprocessors
- Minimize dependency between tasks to exploit parallelism
- Generally true, but some exceptions (or anomalies do exist)
- Good to know about these exceptions when we allocate resources

System Model

- N identical processing units P_i {i=1..n}
- Set of tasks T_i {i=1..n}
- Partial order $<$ on T
  - If $T_i < T_j$ then T_i cannot be started until T_j has been completed.
- Function $\mu : T \rightarrow (0,\infty)$
  - Task T_j takes $\mu(T_j)$ units of time
- Tasks are run to completion and not interrupted
- Sequence $L = (T_{i1}, \ldots, T_{ir})$ contains Tasks ready to be executed . Also called priority list

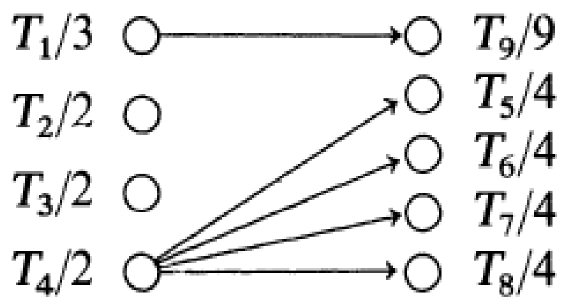
System description

- Processors : scan the list when idle
- Search for tasks which are ready to be executed
  - Have no precedence constraints
  - 2 processors scan list L together ?
    - Assign task to the processor with the smaller index
- If no tasks ready processors becomes idle
  - (We say that it executes an empty task ϕ)
- Finishing time ω : Time at which all tasks are completed

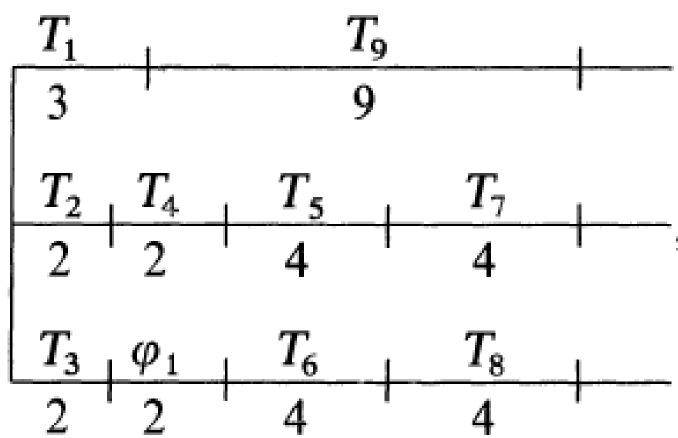
Example to illustrate anomalies

Example. $n = 3; L = (T_1, T_2, \cdots, T_9)$.

$G(\prec, \mu)$:



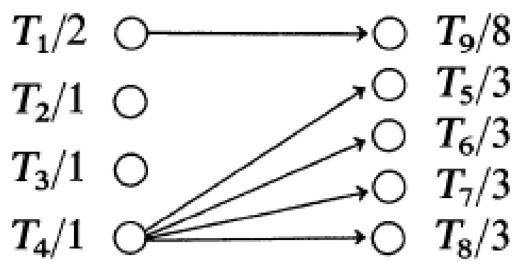
D:



, $\omega = 12.$

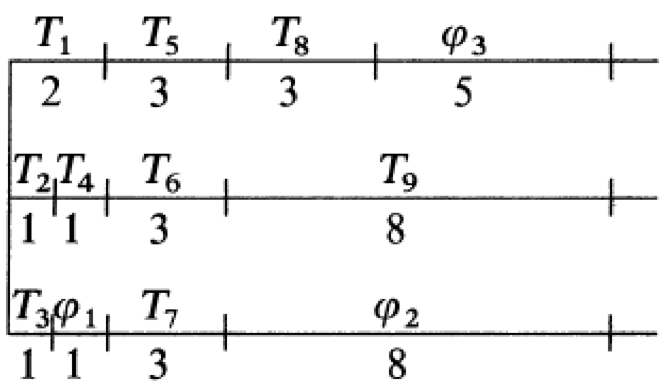
Decreasing computation times

(iii) Decrease μ to μ' by defining $\mu'(T_i) = \mu(T_i) - 1$ for all i. In this case $G(\prec, \mu)$ becomes



and

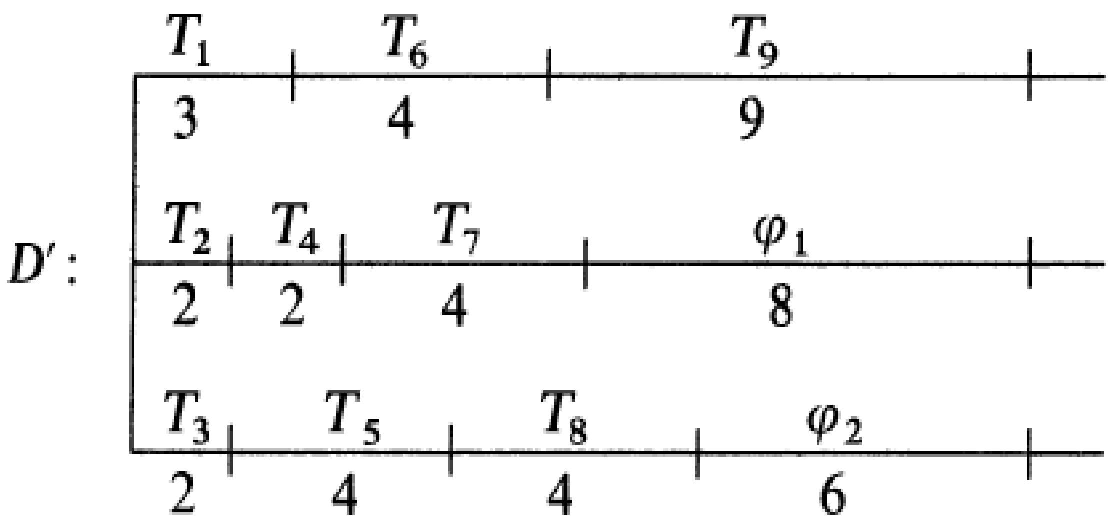
D':



with $\omega' = \omega'(L, \mu', \prec, n) = 13$.

Relaxation of precedence constraints

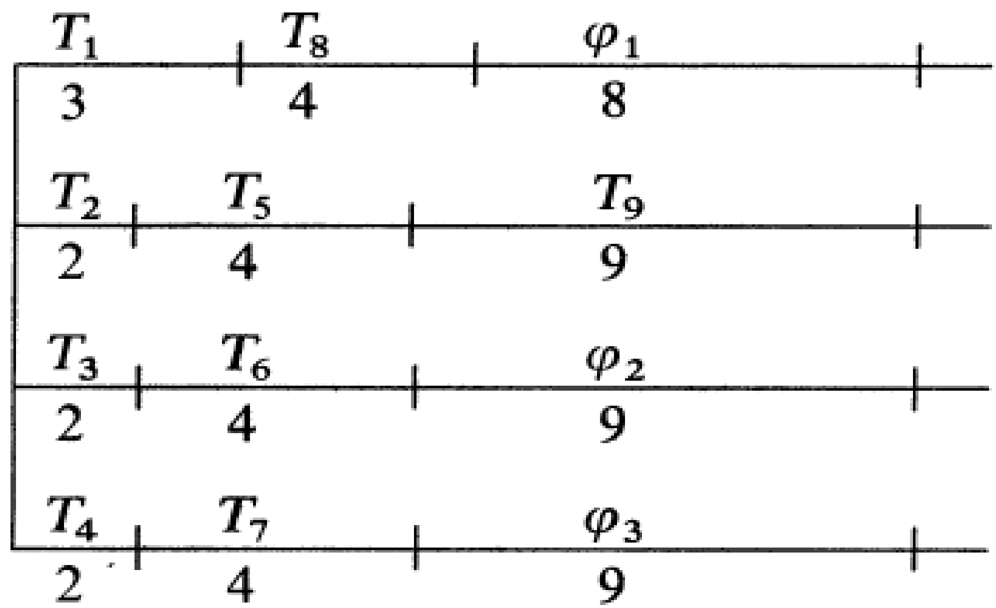
Change $\prec$ to $\prec'$ by removing $T_4 \rightarrow T_5$ and $T_4 \rightarrow T_6$.



and $\omega' = \omega'(L, \mu, \prec', n) = 16$.

Increasing the number of processors

(iv) Increase n from 3 to 4. Then



and $\omega' = 15$.

Observations

- The finishing time can increased even after
  - Relaxing the precedence constraints
  - Increasing the number of processors or
  - Decreasing the computation time

Theorem 1

Case 1 :

- Given a set of T tasks
- A function μ , a partial order $<$
- A list L , n identical processors,
- ω the finishing time for this task set

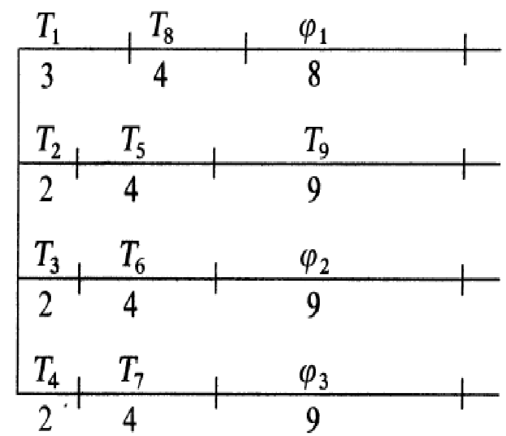
Case 2 :

- Given the same set T of tasks as in Case 1
- A function $\mu' \leq \mu$, a partial order $<'$ which is a subset of $<$
- A list L', n' identical processors,
- ω' the finishing time for this task set

Then $\boldsymbol{\omega' \leq \omega \left(1+(n-1)/n' \right)}$

Some observations

$$\omega' = \frac{1}{n'}\left\{\sum_{T_k \in T} \mu'(T_k) + \sum_{\varphi'_i \in D'} \mu'(\varphi'_i)\right\}$$



$$\sum_{\varphi'_i \in D'} \mu'(\varphi'_i) \leqq (n' - 1) \sum_{k=1}^{m} \mu'(T_{jk}),$$

sum (in time units) of empty tasks

Proof of the theorem

tasks

$$T_{j_m} \prec' T_{j_{m-1}} \prec' \cdots \prec' T_{j_2} \prec' T_{j_1} \tag{1}$$

in D' such that *at every time* $t \in B$, *some* T_{j_k} *is being executed*. We say that this chain *covers* B. The important thing to notice about this chain is

$$\sum_{\varphi_i' \in D'} \mu'(\varphi_i') \leqq (n' - 1) \sum_{k=1}^{m} \mu'(T_{j_k}), \tag{2}$$

where the left-hand sum is over all empty tasks φ_i' in D'. But (1) and the hypothesis $\prec' \subseteq \prec$ imply

$$T_{j_m} \prec T_{j_{m-1}} \prec \cdots \prec T_{j_2} \prec T_{j_1}. \tag{3}$$

Thus

$$\omega \geqq \sum_{k=1}^{m} \mu(T_{j_k}) \geqq \sum_{k=1}^{m} \mu'(T_{j_k}). \tag{4}$$

Consequently, by (2) and (4),

$$\omega' = \frac{1}{n'}\left\{\sum_{T_k \in T} \mu'(T_k) + \sum_{\varphi_i' \in D'} \mu'(\varphi_i')\right\} \tag{5}$$

$$\leqq \frac{1}{n'}(n\omega + (n' - 1)\omega).$$

From this we obtain

$$\frac{\omega'}{\omega} \leqq 1 + \frac{n-1}{n'}, \tag{6}$$

and the theorem is proved.

Theorem 1 (Contd..)

$$\omega' \le \omega\left(1 + (n-1)/n'\right)$$

- For $n = 1$, then ω' is never greater than ω
- For $n > 1$, ω' can be greater than ω even though n' is very high
- For $n = n'$ the ratio ω'/ω goes to $(2-1/n)$

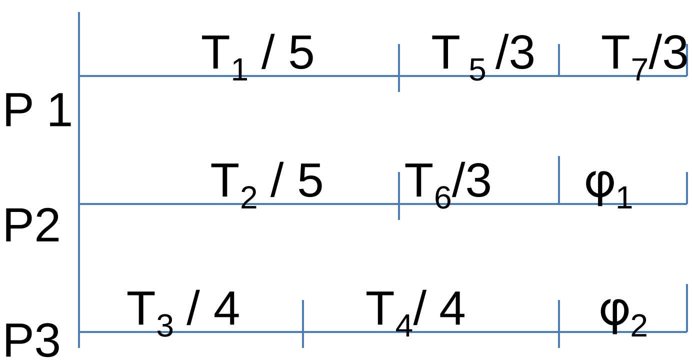
Theorem 2: When no precedence constraints exist

- Tasks can execute when ready
- Consider r tasks
- Let ω_L be the finishing time for the task set
- Let ω_0 be the minimum possible finishing time
- Algo: A free processor always starts to execute the longest unexecuted task
- Then the best possible bound is

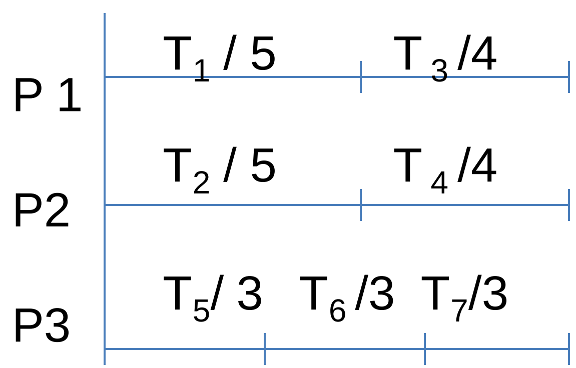
$$\boldsymbol{\omega_L \leq \omega_0 \ (\ (4/3) - 1/n \ )}$$

An example

- Let n = 3,
- Num tasks = r= 2*n + 1 = 7
- T= (T_1,T_2, T_3, T_4,T_5,T_6,T_7) with execution times
- μ = (5, 5 , 4 , 4, 3, 3, 3)



ω_L = 4*n-1 = 4*3-1 = 11



ω_0 = 3*n = 3*3 = 9

Task set

To show that this bound is best possible, we consider the following set of task lengths:

$$\alpha_j = \mu(T_j)$$

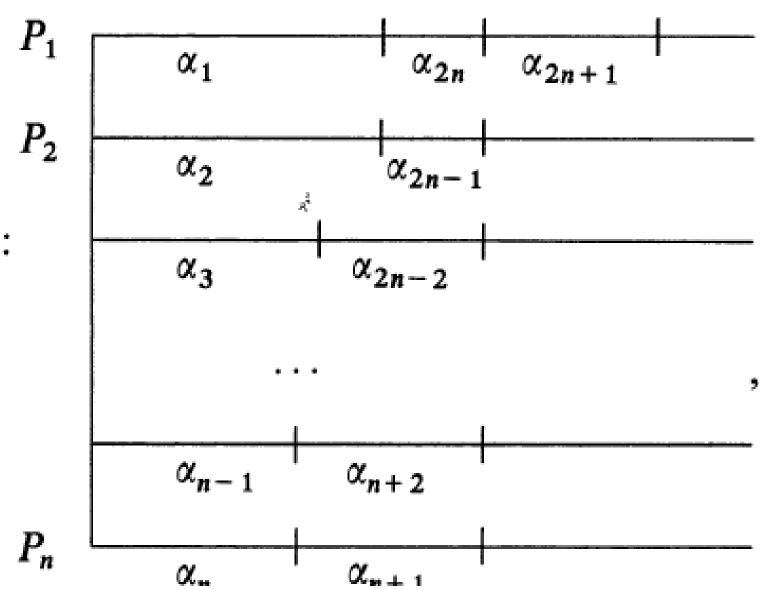
$$(\alpha_1, \alpha_2, \cdots, \alpha_r) = (2n-1, 2n-1, 2n-2, 2n-2, \cdots, n+1, n+1, n, n, n),$$

where $r = 2n+1$. Specifically we have

$$\alpha_k = 2n - \left[\frac{k+1}{2}\right], \qquad k = 1, \cdots, 2n, \quad \text{and} \quad \alpha_{2n+1} = n.$$

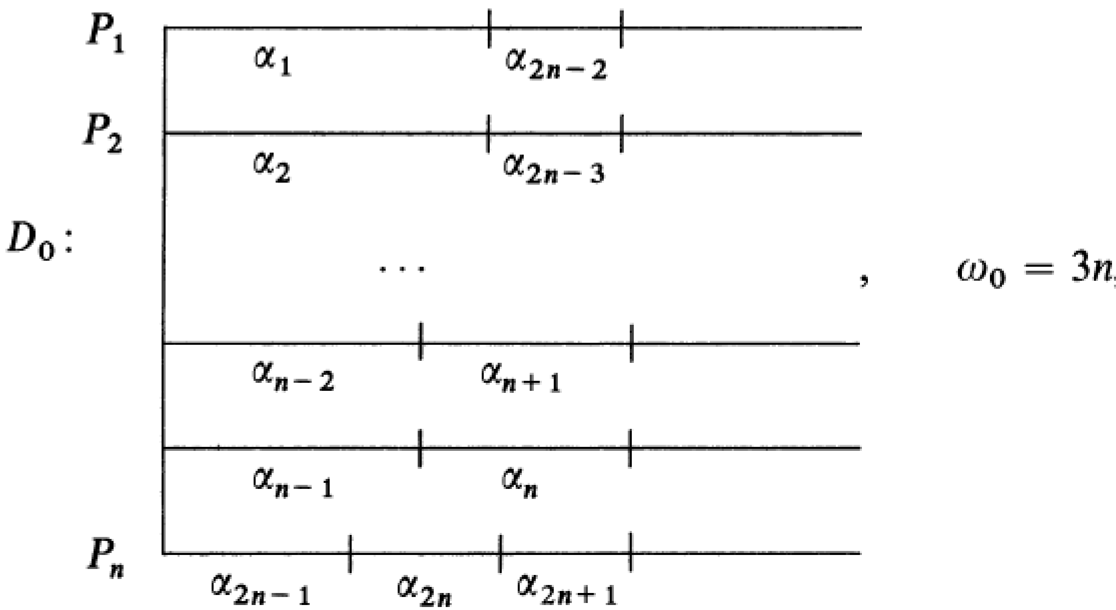
In this case

D_L:



, $\quad \omega_L = 4n - 1,$

Example task set (contd..)



$\omega_L = 4n - 1, \qquad \omega_0 = 3n,$

$$\frac{\omega_L}{\omega_0} = \frac{4}{3} - \frac{1}{3n}.$$

Theorem 3

- No precedence constraints
- For a integer $k \geq 0$, chose k longest tasks of the task set $T = \{T_1, T_2 \ldots. T_r)$
- Arrange them in a list L to get the optimal solution ω_k for the k tasks
- Extend L to a sequence containing all remaining r-k tasks by adjoining them arbitrarily to form the the list L(k)
- Let $\omega(k)$ denote the finishing time of this task set

Theorem 3 (Contd..)

- Let ω_0 denote the min possible finishing time
- Then

$$\frac{\omega(k)}{\omega_0} \leqq 1 + \frac{1 - 1/n}{1 + [k/n]}.$$

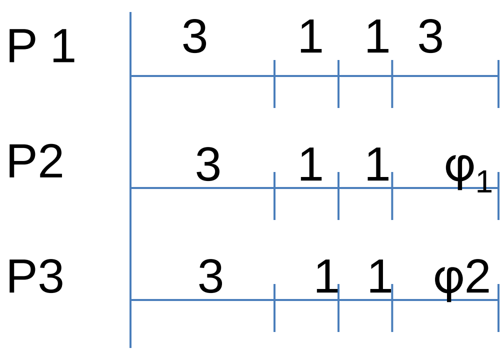
This bound is best possible for k=0(mod n)
Note : If k = 0 then

$$\frac{\omega(0)}{\omega_0} \leqq 2 - \frac{1}{n}$$

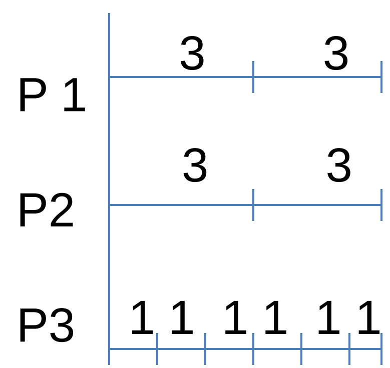
As in theorem 1 for n=n´

An example

- Let n = 3, k=3
- Numtasks = r= k+1 + n*(n – 1) = 4+6 = 10
- μ = (3,3,3,3,1,1,1,1,1,1)



ω (k) = k +2*n-1 = 3+6-1 = 8



ω_0 = k+n = 3 +3 = 6

An example task set

To show that this bound is best possible when $k \equiv 0 \pmod{n}$ we present the following example: Define α_i for $1 \leqq i \leqq k + 1 + n(n-1)$ by

$$\alpha_i = \begin{cases} n \text{ for } 1 \leqq i \leqq k+1, \\ 1 \text{ for } k+2 \leqq i \leqq k+1+n(n-1). \end{cases}$$

For this set of tasks and the list $L(k) = (T_1, \cdots, T_k, T_{k+2}, \cdots, T_{k+1+n(n-1)}, T_{k+1})$ we have $\omega(k) = k + 2n - 1$. Since $\omega_0 = k + n$,

$$\frac{\omega(k)}{\omega_0} = \frac{k+2n-1}{k+n} = 1 + \frac{n-1}{k+n} = 1 + \frac{1-1/n}{1+k/n} = 1 + \frac{1-1/n}{1+[k/n]}.$$

- Let $n = 3$
- Let $k = 3$
- Let $r = 10$ (No of tasks $= k+1+n*(n-1)$)

End-notes

- This paper was presented in 1969 in the SIAM (Society for Industrial and Applied Mathematics) Journal on Applied Mathematics
- A Sequel to this was presented in 1972 by the same author
- Presented some anomalies
- Provided some algorithms for tasks assignments to processors and these could be used in different fields.