1218206:- Roll no. / 3C01

Parallelism Profile in Programs:-

The degree of parrelelism reflects an extend to which software " matches Hardware parrlelism.

Degree of Parallelism:-

* The execution of a Program on a parallel computer may use different number of processors at different time periods during execution cycle.
* So DOP (Degree of Parallelism) is no. of processors used to execute a program for each time period.
* It is considered a discrete time function assuming only non-neg integers.
* Parallelism Profile:-
  - The Plot of the DOP as function of time is called parallelism profile of a given program. For simplicity we can concentrate on the analysis of single - program profiles. Some software can also trace the parallism profile.
  - The fluctuation of profile during an observation period depends on the algorithmic structure, program optimization, resource utilization and run time conditions of computer system.
  - In DOP there were some assumptions like having unbound number of avalaible processors and other necessary resources. But DOP not be avaliable on a real computer with limited resources.

Average Parallelism :- (A)

* We are considering a parallel computer of "n" homogenous processors.
* In ideal case :- $n >> m$
* Computing capacity Δ of a single processor is approximated by the execution rate such as MIPS or Mflops etc.

* with i PEs busy, DOP will be i,

Then total work will be

$$W = \Delta \int_{t_1}^{t_2} DOP(t)\,dt$$

// total W (instructions or computations) performed is proportional to area under the profile curve.

$$W = \Delta \sum_{i=1}^{m} i . t_i$$

where t_i is total amount of time that DOP = i and $\sum_{i=1}^{m} t_i = t_2 + t_1$ is total elapsed time

→ Avalaible parallism (A) is computed by :-

$$A = \frac{1}{t_2 - t_1} \int_{t_2}^{t_2} DOP(t)\,dt$$

In discrete :-

$$A = \frac{\left(\sum_{i=1}^{m} i . t_i\right)}{\sum_{i=1}^{m} t_i}$$

1218206

* Available Parallesm:-

ø optimistic side:-

① 500-3500 concurrent arithmatic operations

⑪ 90 parallism factor available for very long-instruction word architects.

ø Pessimistic side:-

① David wall indicated that the limits of instructnal- level parallelis m is around 5 rarely exceeding 7.

⑪ Butler et al. reported that when all constraits are removed the DOP in programs may exceeds 17 instrction per cycle.

These number shows Pessimictic side of Available parrallelis m.

ø Basic block:-

A basic block is a seq or block of instruction in a program that has single entry and single exit point.

Limiting parallelism extraction to basic block limits the potential instruction level parallism to a factor of 2-5 in ordinary programs.

Can be pushed to 1000s for scientic applications using basic block.

Asymptotic speedup:-

$$W = \sum_{i=1}^{m} w_i$$

$$t_i(K) = \frac{w_i}{K\Delta} \quad \text{as} \quad w_i = i\,\Delta t_i \;\cancel{\Delta o}\; \cancel{w_i}.$$

$$t_i(1) = \frac{w_i}{\Delta}$$

$$t_i(\infty) = \frac{w_i}{i\Delta} \quad 1 \leq i \leq m$$

Response time:-

$$T(1) = \sum_{i=1}^{m} t_i(1) = \sum_{i=1}^{m} \frac{w_i}{\Delta}$$

similarly

$$T(\infty) = \sum_{i=1}^{m} t_i(\infty) = \sum_{i=1}^{m} \frac{w_i}{i\Delta}$$

Asymtotic speedup:-

S_∞ is defined by ratio of $T(1)$ to $T(\infty)$

$$S_\infty = \frac{T(1)}{T(\infty)} = \frac{\sum_{i=1}^{m} w_i}{\sum_{i=1}^{m} w_i/i}$$

old time by new time

ideally $S_\infty = A$

In genral $S_\infty < A$

$\rightarrow$ Because S_∞ and A are defined under the assumption $n = \infty$ & $n >> m$

Mean Performance :-

consider a parallel computer with n processors executing m programs in various mode with different performance level.
we want to define the mean performance of such multimode computers.

Different execution modes may correspond to scaler, vector, sequential or parallel Processing with different program parts. Each program may be executed with a combination of these modes.

Arithmetic Mean Performance :-

Let $\{R_i\}$ be the execution rate of programs i.e. $i = 1, 2 \ldots m$, i.e. $i = 1, 2, \ldots, m$ upto m.

$$R_a = \sum_{i=1}^{m} \frac{R_i}{m}$$

The expression R_a assumes equal weighting $(1/m)$ on all m programs. &

$$R_a^* = \sum_{i=1}^{m} (f_i R_i) \quad // \text{ weighted arthematic mean.}$$

If the Programs are weighted with a distribution $\pi = \{f_i\}$, where $i = 1, 2, 3 \ldots, m\}$ we say its weighted arthmetic mean.

* Arthematic mean execution rate is proportinal to sum of inverse of execution times. the Ra fails to rep the real time consumed.

Harmonic Mean Performance:- $T_a = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{R_i}$

The Harmonic mean execution rate across m benchmark programs is thus defined by fact $R_h = 1/T_a$ so: $R_h = \frac{m}{\sum_{i=1}^{m}(1/R_i)}$

Therefore the Harmonic mean performance is related to the average execution time.

Weighted Harmonic Mean execution:-

$$R^*_h = \frac{1}{\sum_{i=1}^{m}(f_i/R_i)}$$

The above harmonic mean performance expressions correspond to the total number of operation divided by the total time.

Compared to arithmetic mean the harmonic mean execution rate is closer to real performance.

Harmonic Mean Speed-up:-

suppose a program is to be executed on n-processor system. During the execution period, the program may use $i = 1, 2, \ldots, n$ processors in different time periods.

we say the program is executed in mode i if "i" processors are used.

The corresponding exen rate R_i, is used to reflect the collective speed of i processors.

Assuming $T_1 = 1/R_1 = 1$ if seq execution ~~time~~ on a uniprocessor ~~used to reflect~~ ~~collective speed~~ ~~of i processors.~~

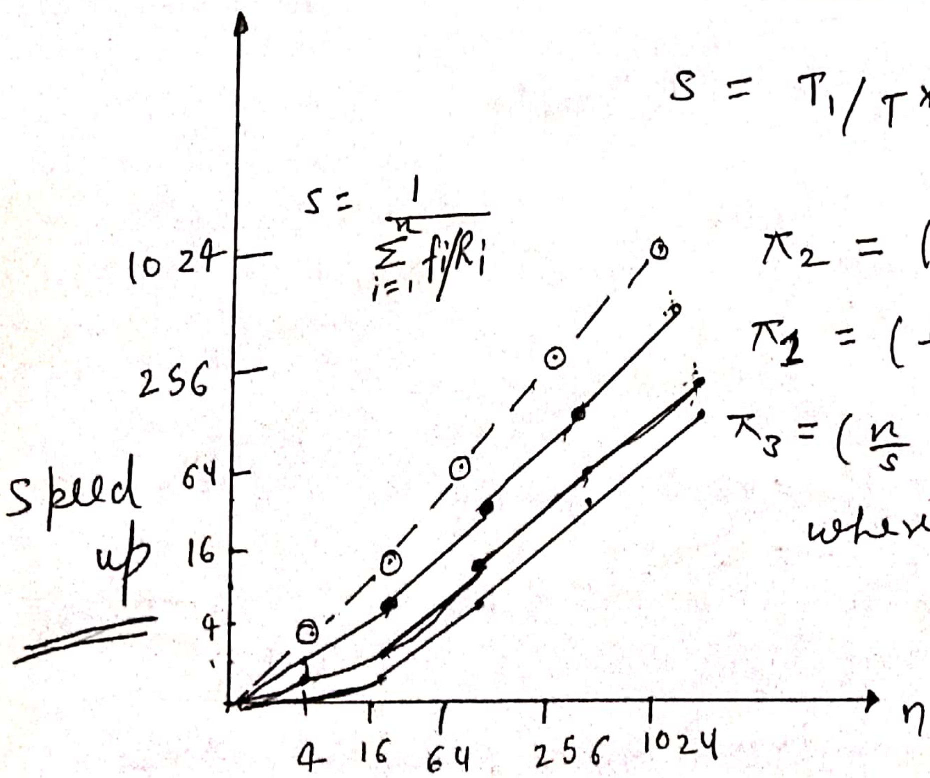
with an exen rate of $R_i = i$

in ideal case:

T^* is weighed arith mean speed up.

where $T^* \rightarrow$ weighed ~~arith~~ harmonic speed up

$$S = T_1 / T^* = \frac{1}{\sum_{i=1}^{n} f_i / R_i}$$



$$\pi_2 = \left(\frac{1}{s}, \frac{2}{s}, \ldots, \frac{n}{s}\right)$$

$$\pi_1 = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)$$

$$\pi_3 = \left(\frac{n}{s}, \frac{n-1}{s}, \ldots, \frac{1}{s}\right)$$

where $s = \sum_{i=1}^{n} i$

$\pi_1 \rightarrow$ avg

$\pi_3 \rightarrow$ rev

$\pi_2 \rightarrow$ for spin

Amdahl's law:-

Assume:-

$R_i \rightarrow$ Ratio of execution $= i$

$w = (\alpha, 0, 0, 0 \ldots 1-\alpha)$ ↳ alpha

$w_1 = \alpha \rightarrow$ execution in sequential fe

$w_n = 1-\alpha \rightarrow$ execution in parallel

for w_1, R_1 will be 1

w_n it will be $R_n \rightarrow n$

we can speed up expression:-

Expressions

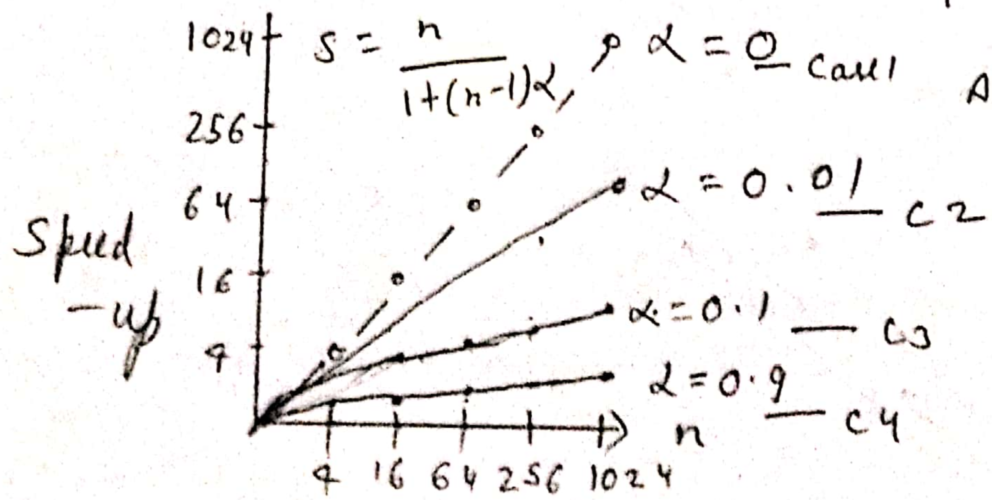
$S \rightarrow 1/\alpha$

as $n \rightarrow \infty$

$$S_n = \frac{1}{(\alpha/1) + ((1-\alpha)/n)}$$

$$S_n = \frac{n}{1+(n-1)\alpha}$$ after solving

This is known as Amdahl's law.



as $\alpha \uparrow$ the speed $\Downarrow$ sharply

we plot as a func^n of n for four values of α

where $\alpha = 0$ (case 1) the ideal speed up is achieved.

from 0.01 (c-2) to 0.1 (c-3) to 0.9 (c-4), speed prod drops

Efficiency, Utillization, Quality:-

Ruby Lee (1980) has defined several parameters for evaluating parallel computing. These are the fundamentals concepts of parallel computing.

System efficiency:-

Let $O(n)$ be the total no. of unit operations performed by n-processor system and $T(n)$ be the execution time in unit time steps. Ingem $T(n) < O(n)$ if more than one operation is performed by n processers per unit time, where $n > 2$. Assume $T(1) = O(1)$ in uniprocessor system. The speed up factor is defined as

$$S(n) = T(1) / T(n)$$

The system efficiency for n-processors is defined by:-

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{n\,T(n)}$$

Efficiency is an indication of the actual degree of speedup performance achieved as compared with maximum value. Since $1 \leq S(n) \leq n$, we have $1/n \leq E(n) \leq 1$.

The max efficiency can be achieved by when all n processors are fully utilized throughout the execution period.

Redundancy and utilization:-

The redundancy in parallel computing is defined as the ratio of $O(n)$ to $O(1)$:

$$R(n) = O(n)/O(1)$$

The ratio signifies the extent of matching b/w software and hardware parallelism.

obviously:- $U(n) = R(n)E(n) = O(n)/nT(n)$

The system utilization indicates the percentage of resources (processors, memory, etc) that was kept busy during the execution of a parallel program.

Quality of Parallelism:-

The quality of a parallel computing is directly proportinal to the speedup and efficiency and inversely related to the redundancy.

Thus we have :-

$$Q(n) = \frac{S(n)E(n)}{R(n)} = \frac{T^3(1)}{nT^2(n)\,O(n)}$$

Since $E(n)$ is always a fraction and $R(n)$ is a number b/w 1 and n; the quality $Q(n)$ is always bounded by the speedup $S(n)$.

Example:- Hypothetical workload:-

$S(n) = (n+3)/4$

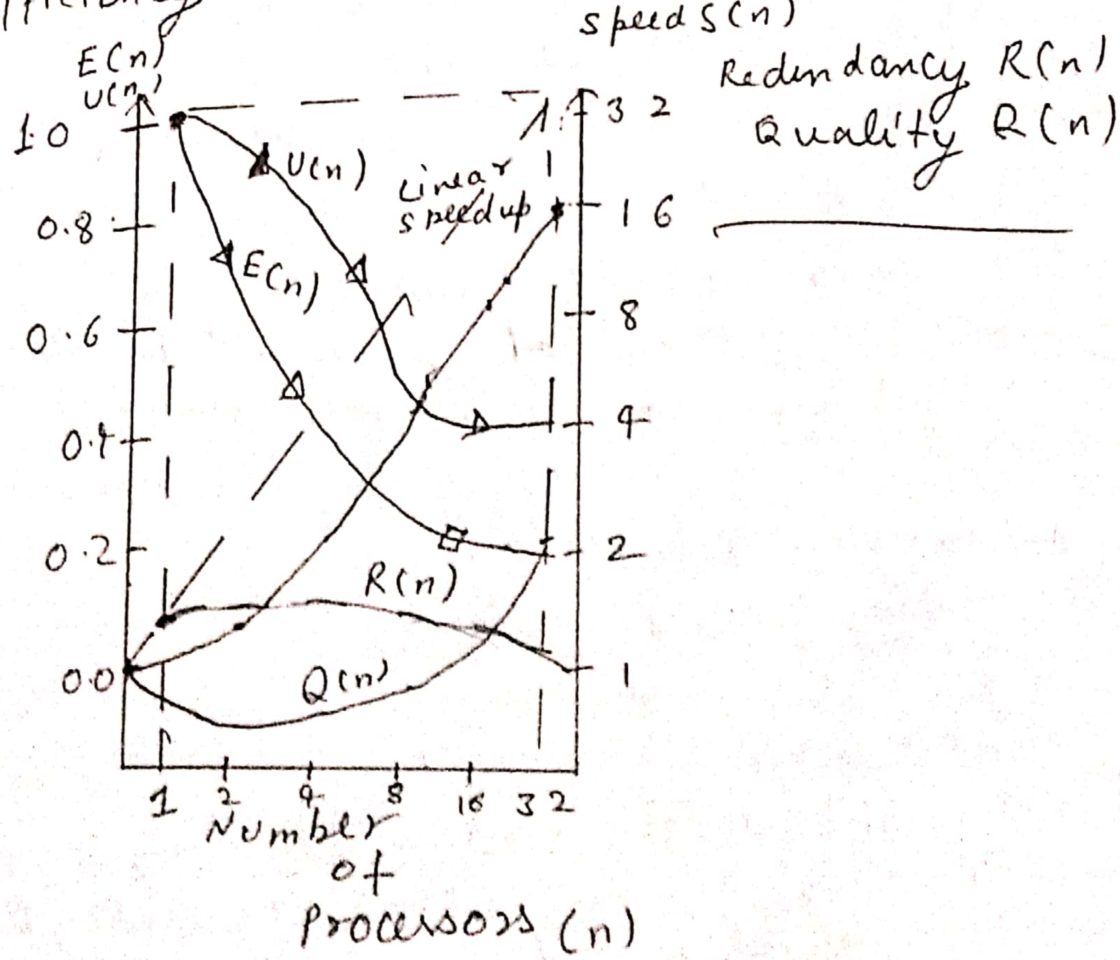
$E(n) = (n+3)/(4n)$

$R(n) = (n+\log_2 n)/n$

$U(n) = (n+3)(n+\log_2 n)/(4n^2)$

$Q(n) = (n+3)^2/16(n+\log_2 n))$

The relationships $1/n \leq E(n) \leq U(n)$ and $0 \leq Q(n) \leq S(n) \leq n$ are observed which the linear speed up. corresponds to the ideal case of 100% efficiency.



Benchmarks and Performance Metrics:-

* The Drystone Result:-

- The CPU intensive synthetic benchmark consisting of a mix of about 100 high level language instructions.
- The Drystone statements are balenced with respect to statement type, data type and locality of reference with no OS or not using library or subrutines.
- Measure of integer performance
- No floats
- Unit:- KDhrystones/s
- The Drystone reating should be used as measure of integer performance.

* The whetstone Result:-

- Floating point Performance
- Synthetic benchmark.
- Unit : kwhetstones/S
- This include both integer and floating point opreation indexing, subroutine calls, parameter passing conditinal branching and trignometric function
- whetstone does not contain any vectorizable code and shows dependance on system's mathematic lib and efficiency of the code generated by a compiler.

The whetstone performance is not equivalent to the Mflops performance, although the whetstone contains a large number of scaler floating-point operation

* Both the Dhrystone & whetstone are considered synthetic benchmarks programs result deeply on compiler used. As a matter of fact the Dhrystone benchmark is originally written to test the CPU and compiler performance. for a typical program. However both benchmarks are criticized for being unable to pridict proformance of user programs.

// TPS & KLIPS Range:-

"TPS stands for Transactions per second"

On-line transaction processing application demand rapid, interactive processing for large number of relatively simple transaction. They are typically supported by very-large databases. Automated teller machines and airlines reservation are familier example:- Today many application are web based.

=> ON-line transaction processing is often measured in TPS.

=> Each transaction may involve a DB search query answering, database update operation.

=> Bussness computers & servers should designed to deliver high TPS rate.

KLIPS:-

In AI applications "kilo logic inferences per sec" (KLIPS) was used at one time to indicate the reasoning power of AI machine. For example:-

A High-speed inference machine dev. in japan's fifth-Gen computers system claimed 400KLIPS.