Chapter 3: Principles of Scalable Performance

- Performance measures
- Speedup laws
- Scalability principles
- Scaling up vs. scaling down

Performance metrics and measures

- Parallelism profiles
- Asymptotic speedup factor
- System efficiency, utilization and quality
- Standard performance measures

Degree of parallelism

- Reflects the matching of software and hardware parallelism
- Discrete time function – measure, for each time period, the # of processors used
- Parallelism profile is a plot of the DOP as a function of time
- Ideally have unlimited resources

Factors affecting parallelism profiles

- Algorithm structure
- Program optimization
- Resource utilization
- Run-time conditions
- Realistically limited by # of available processors, memory, and other nonprocessor resources

Average parallelism variables

- n – homogeneous processors
- m – maximum parallelism in a profile
- Δ - computing capacity of a single processor (execution rate only, no overhead)
- $DOP=i$ – # processors busy during an observation period

Average parallelism

- Total amount of work performed is proportional to the area under the profile curve

$$W = \Delta \int_{t1}^{t2} DOP(t)dt$$

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

Average parallelism

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

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

Example: parallelism profile and average parallelism

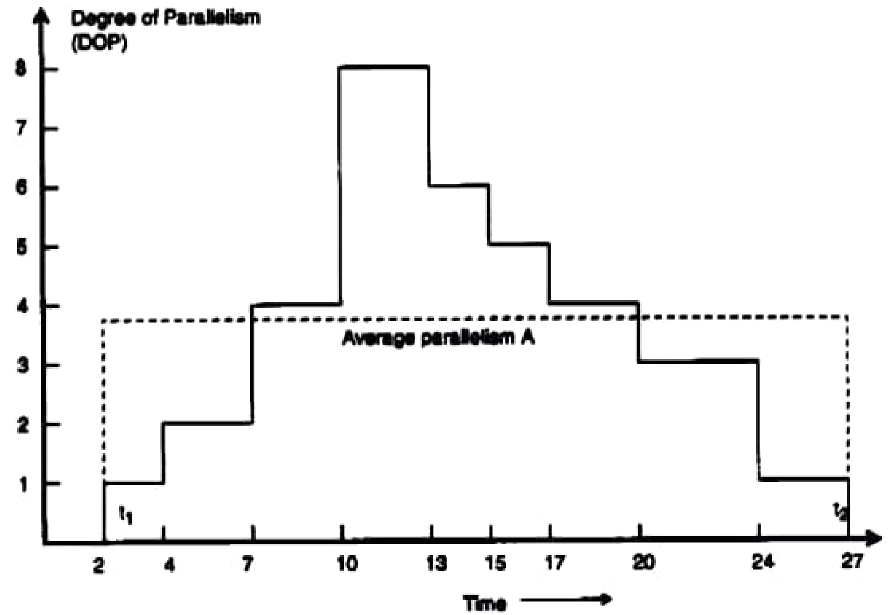

Figure 3.1 Parallelism profile of a divide-and-conquer algorithm.

Asymptotic speedup

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

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

(response time)

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

= A in the ideal case

Performance measures

- Consider *n* processors executing *m* programs in various modes
- Want to define the mean performance of these multimode computers:
  - Arithmetic mean performance
  - Geometric mean performance
  - Harmonic mean performance

Arithmetic mean performance

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

Arithmetic mean execution rate (assumes equal weighting)

$$R_a^* = \sum_{i=1}^{m} (f_i R_i)$$

Weighted arithmetic mean execution rate

-proportional to the sum of the inverses of execution times

Geometric mean performance

$$R_g = \prod_{i=1}^{m} R_i^{1/m}$$

Geometric mean execution rate

$$R_g^* = \prod_{i=1}^{m} R_i^{f_i}$$

Weighted geometric mean execution rate

-does not summarize the real performance since it does not have the inverse relation with the total time

Harmonic mean performance

$$T_i = 1/R_i$$

Mean execution time per instruction
For program i

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

Arithmetic mean execution time per instruction

Harmonic mean performance

$$R_h = 1/T_a = \frac{m}{\sum_{i=1}^{m} (1/R_i)}$$

Harmonic mean execution rate

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

Weighted harmonic mean execution rate

-corresponds to total # of operations divided by the total time (closest to the real performance)

Harmonic Mean Speedup

- Ties the various modes of a program to the number of processors used
- Program is in *mode* i if i processors used
- Sequential execution time $T_1 = 1/R_1 = 1$

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

Harmonic Mean Speedup Performance

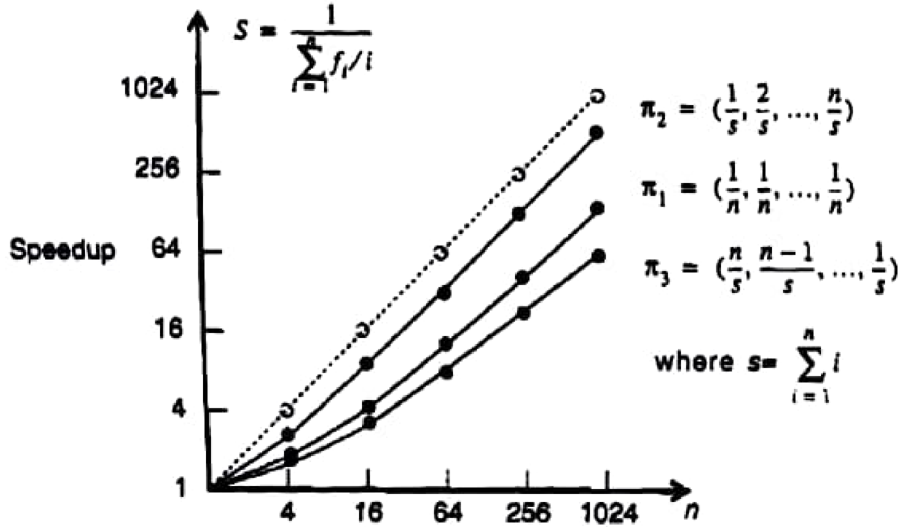

Figure 3.2 Harmonic mean speedup performance with respect to three probability distributions: π_1 for uniform distribution, π_2 in favor of using more processors, and π_3 in favor of using fewer processors

Amdahl's Law

- Assume $R_i = i$, $w = (\alpha, 0, 0, \ldots, 1-\alpha)$
- System is either sequential, with probability α, or fully parallel with prob. $1-\alpha$

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

- Implies $S \to 1/\alpha$ as $n \to \infty$

Speedup Performance

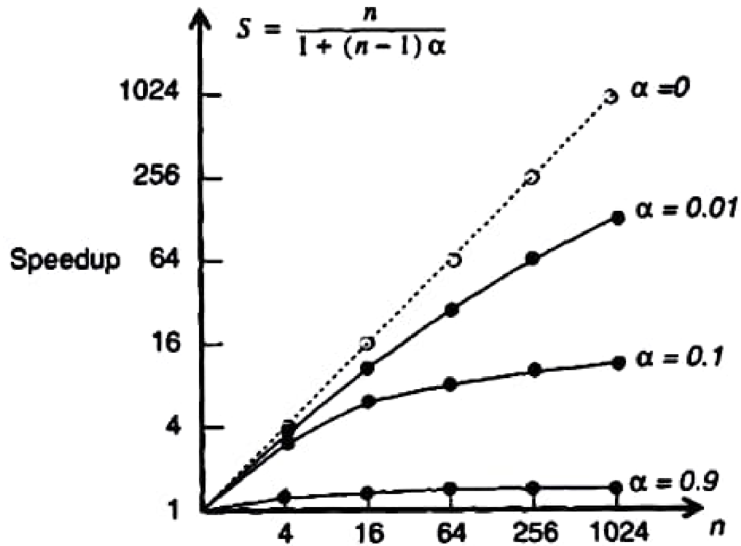

Figure 3.3 Speedup performance with respect to the probability distribution $\pi = (\alpha, 0, \ldots, 0, 1-\alpha)$ where α is the fraction of sequential bottleneck

System Efficiency

- $O(n)$ is the total # of unit operations
- $T(n)$ is execution time in unit time steps
- $T(n) < O(n)$ and $T(1) = O(1)$

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

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

Redundancy and Utilization

- Redundancy signifies the extent of matching software and hardware parallelism

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

- Utilization indicates the percentage of resources kept busy during execution

$$U(n) = R(n)E(n) = \frac{O(n)}{nT(n)}$$

Quality of Parallelism

- Directly proportional to the speedup and efficiency and inversely related to the redundancy
- Upper-bounded by the speedup $S(n)$

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

Example of Performance

- Given $O(1) = T(1) = n^3$, $O(n) = n^3 + n^2 \log n$, and $T(n) = 4\, n^3/(n+3)$
  - ⌘ $S(n) \quad = \quad (n+3)/4$
  - ⌘ $E(n) \quad = \quad (n+3)/(4n)$
  - ⌘ $R(n) \quad = \quad (n + \log n)/n$
  - ⌘ $U(n) \quad = \quad (n+3)(n + \log n)/(4n^2)$
  - ⌘ $Q(n) \quad = \quad (n+3)^2 / (16(n + \log n))$

Standard Performance Measures

- MIPS and Mflops
  - Depends on instruction set and program used
- Dhrystone results
  - Measure of integer performance
- Whestone results
  - Measure of floating-point performance
- TPS and KLIPS ratings
  - Transaction performance and reasoning power

Parallel Processing Applications

- Drug design
- High-speed civil transport
- Ocean modeling
- Ozone depletion research
- Air pollution
- Digital anatomy

Application Models for Parallel Computers

- Fixed-load model
  - Constant workload
- Fixed-time model
  - Demands constant program execution time
- Fixed-memory model
  - Limited by the memory bound

Algorithm Characteristics

- Deterministic vs. nondeterministic
- Computational granularity
- Parallelism profile
- Communication patterns and synchronization requirements
- Uniformity of operations
- Memory requirement and data structures

Isoefficiency Concept

- Relates workload to machine size n needed to maintain a fixed efficiency

workload

$$E = \frac{w(s)}{w(s) + h(s,n)}$$

overhead

- The smaller the power of n, the more scalable the system

Isoefficiency Function

- To maintain a constant E, $w(s)$ should grow in proportion to $h(s,n)$

$$w(s) = \frac{E}{1-E} \times h(s,n)$$

- $C = E/(1-E)$ is constant for fixed E

$$f_E(n) = C \times h(s,n)$$

Speedup Performance Laws

- Amdahl's law
  - for fixed workload or fixed problem size
- Gustafson's law
  - for scaled problems (problem size increases with increased machine size)
- Speedup model
  - for scaled problems bounded by memory capacity

Amdahl's Law

- As # of processors increase, the fixed load is distributed to more processors
- Minimal turnaround time is primary goal
- Speedup factor is upper-bounded by a sequential bottleneck
- Two cases:
  - ⋆ DOP $< n$
  - ⋆ DOP $\geq n$

Fixed Load Speedup Factor

- Case 1: DOP > n

$$t_i(i) = \frac{W_i}{i\Delta}\left\lceil \frac{i}{n} \right\rceil$$

$$T(n) = \sum_{i=1}^{m} \frac{W_i}{i\Delta}\left\lceil \frac{i}{n} \right\rceil$$

- Case 2: DOP < n

$$t_i(n) = t_i(\infty) = \frac{W_i}{i\Delta}$$

$$S_n = \frac{T(1)}{T(n)} = \frac{\sum_{i=i}^{m} W_i}{\sum_{i=1}^{m} \frac{W_i}{i}\left\lceil \frac{i}{n} \right\rceil}$$

Gustafson's Law

- With Amdahl's Law, the workload cannot scale to match the available computing power as n increases
- Gustafson's Law fixes the time, allowing the problem size to increase with higher n
- Not saving time, but increasing accuracy

Fixed-time Speedup

- As the machine size increases, have increased workload and new profile
- In general, $W_i' > W_i$ for $2 \leq i \leq m'$ and $W_1' = W_1$
- Assume $T(1) = T'(n)$

Gustafson's Scaled Speedup

$$\sum_{i=1}^{m} W_i = \sum_{i=1}^{m} \frac{W_i^{'}}{i} \left\lceil \frac{i}{n} \right\rceil + Q(n)$$

$$S_n^{'} = \frac{\sum_{i=1}^{m^{'}} W_i^{'}}{\sum_{i=1}^{m} W_i} = \frac{W_1 + nW_n}{W_1 + W_n}$$

Memory Bounded Speedup Model

- Idea is to solve largest problem, limited by memory space
- Results in a scaled workload and higher accuracy
- Each node can handle only a small subproblem for distributed memory
- Using a large # of nodes collectively increases the memory capacity proportionally

Fixed-Memory Speedup

- Let M be the memory requirement and W the computational workload: $W = g(M)$
- $g^*(nM) = G(n)g(M) = G(n)W_n$

$$S_n^* = \frac{\sum_{i=1}^{m^*} W_i^*}{\sum_{i=1}^{m^*} \frac{W_i^*}{i} \left\lceil \frac{i}{n} \right\rceil + Q(n)} = \frac{W_1 + G(n)W_n}{W_1 + G(n)W_n / n}$$

Relating Speedup Models

- $G(n)$ reflects the increase in workload as memory increases n times
- $G(n) = 1$: Fixed problem size (Amdahl)
- $G(n) = n$: Workload increases n times when memory increased n times (Gustafson)
- $G(n) > n$: workload increases faster than memory than the memory requirement

Scalability Metrics

- Machine size (n) : # of processors
- Clock rate (f) : determines basic m/c cycle
- Problem size (s) : amount of computational workload. Directly proportional to $T(s,1)$.
- CPU time ($T(s,n)$) : actual CPU time for execution
- I/O demand (d) : demand in moving the program, data, and results for a given run

Scalability Metrics

- Memory capacity (m) : max # of memory words demanded
- Communication overhead ($h(s,n)$) : amount of time for interprocessor communication, synchronization, etc.
- Computer cost (c) : total cost of h/w and s/w resources required
- Programming overhead (p) : development overhead associated with an application program

Speedup and Efficiency

- The problem size is the independent parameter

$$S(s,n)=\frac{T(s,1)}{T(s,n)+h(s,n)}$$

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

Scalable Systems

- Ideally, if $E(s,n)=1$ for all algorithms and any s and n, system is scalable
- Practically, consider the scalability of a m/c

$$\Phi(s,n) = \frac{S(s,n)}{S_I(s,n)} = \frac{T_I(s,n)}{T(s,n)}$$