Amdahl's Law

- Assumes that the speedup is not superliner; i.e.,

$$S(n) = t_s / t_p \le n$$

- By Figure 1.29 (or slide #40),

$$t_p \le f\, t_s + (1\text{-}f)\, t_s$$

- Substituting above values into the above equation for S(n) and simplifying (see slide #41 or book) yields

$$S(n) \le \frac{n}{1+(n-1)f} \le \frac{1}{f}$$

- Above inequality is known as *Amdahl's law*.
- See Slide #41 or Fig. 1.30 for related details.
- Note that *S(n)* does not exceed *1/f* for all and approaches *1/f* as a limit as *n* increases.
- Example: If only 5% of the computation is serial, the maximum speedup is 20, no matter how many processors are used.
- **Observations:** Amdahl's law limitations to parallelism:
  - For a long time, Amdahl's law was viewed as a severe limit to the usefulness of parallelism.
  - Note that the argument focuses on the steps in a particular algorithm.

- – **Gustafon's Law:** The proportion of the computations that are sequential normally decreases as the problem size increases.
- – Also, Amdahl's law does not apply to non-standard problems were superlinearity occurs.
- – For details on superlinearity, see Parallel Computation: Models and Methods, Selim Akl, pgs 14-20 (Speedup Folklore Theorem) and Chapter 12.

More Metrics for Parallelism

- **Efficiency** is defined by

$$E = \frac{t_s}{t_p + n} = \frac{S(n)}{n}$$

  - Efficiency give the percentage of time that the processors are effectively being used on the computation.
- **Cost:** The cost of a parallel algorithm or parallel execution is defined by

$$\textit{Cost} = (\textit{running time}) \times (\textit{Nr. of Processors})$$
$$= t_p \times n$$

  - The *cost* of a parallel computation can be compared to the *running time* of a sequential computation.

More Metrics (cont.)

- – If a sequential algorithm is executed in parallel and each PE does *1/n* of the work in *1/n* of the sequential running time, then the parallel *cost* is the same as the sequential running time.
- **Cost-Optimal Parallel Algorithm:** A parallel algorithm for a problem is said to be cost-optimal if its cost is proportional to the running time of an optimal sequential algorithm for the same problem.
  - By *proportional*, we means that

$$\textit{cost} = t_p \times n = k \times t_s$$

    where *k* is a constant. (See pg 67 of text).
  - Equivalently, a parallel algorithm is optimal if

$$\textit{parallel cost} = O(f(t)),$$

    where *f(t)* is the running time of the optimal sequential algorithm.
  - In cases where no optimal sequential algorithm is known, then the "fastest known" sequential algorithm is often used instead.
    - Also, see pg 67 of text.

Evaluating and Debugging Message-Passing Programs

- Most of this chapter concerns MPI and PVM, which is being covered by Professor Walker.
- An overview of Section 2.3 (Evaluating Parallel Programs) and Section 2.4 (Debugging & Evaluating Parallel Programs) using some slides prepared by the textbook authors for Chapter 2 will be given next.
- Sections 2.3 and 2.4 are part of your reading assignment. Some parts of 2.3 and 2.4 may also be discussed as part of MPI & PVM programming, as needed.
- The author's slides that will be used are as follows:

  76-78, 81, 83, 85-90, 92-93.

- It is anticipated that the authors' slides for Ch. 2 will be posted in PDF at our website. Since the coverage of these slides will be brief, it may be simplest to review them online.
- The Big-O, Ω, and Θ notation and related definitions concerning complexity in Subsection 2.3.2 are standard concepts in basic algorithms. If you are unfamiliar with any of these, you should study this subsection carefully.