**1. Introduction to Analytical Modeling**

**Key Concepts**

* **Sequential vs. Parallel Evaluation**:
  + **Sequential**: Runtime depends only on input size (n).
  + **Parallel**: Runtime depends on:
    1. Input size (n).
    2. Number of processing elements (p).
    3. Computation/communication speeds.
* **Parallel System**: Combination of an **algorithm** and the **parallel architecture** it runs on.

**Performance Metrics**

* **Wall Clock Time**: Simple but not scalable (cannot extrapolate to larger problems/machines).
* **Parallelism Benefit**: Speedup compared to serial execution, but flawed if the serial algorithm is suboptimal.

**2. Sources of Overhead in Parallel Programs**

**Major Overheads**

1. **Inter-Process Interaction**:
   * Time spent communicating data (e.g., MPI messages).
   * **Dominant overhead** in most parallel systems.
2. **Idling**:
   * Caused by:
     + Load imbalance.
     + Synchronization points.
     + Serial sections (Amdahl’s Law).
3. **Excess Computation**:
   * Using a parallelizable but less efficient algorithm (e.g., slower convergence in parallel solvers).

**Visualization**

* **Execution Profile**:
  + **Useful Computation**: Time spent on actual problem-solving.
  + **Overhead**: Communication, idling, redundant work.

**3. Performance Metrics for Parallel Systems**

**Execution Time**

* **Serial Runtime (**Tₛ**)**: Time on a single processor.
* **Parallel Runtime (**Tₚ**)**: Time from start to last processor finishing.

**Total Parallel Overhead (**Tₒ**)**

* Defined as:

To=p⋅Tp−Ts*To*​=*p*⋅*Tp*​−*Ts*​

* + **Interpretation**: Extra time due to parallelism (communication, idling, etc.).

**Speedup (**S**)**

S=TsTp*S*=*Tp*​*Ts*​​

* **Ideal**: Linear speedup (S = p).
* **Real-World**: Sublinear due to overheads.

**Efficiency (**E**)**

E=Sp=Tsp⋅Tp*E*=*pS*​=*p*⋅*Tp*​*Ts*​​

* **Range**: 0 (worst) to 1 (ideal).
* Measures **utilization** of processing elements.

**Cost**

Cost=p⋅TpCost=*p*⋅*Tp*​

* **Cost-Optimal**: If cost grows asymptotically like the best serial algorithm (e.g., Θ(n)).
* **Example**: If Tₛ = O(n log n) and Tₚ = O(n log n / p), then cost is O(n log n) (optimal).

**4. Effect of Granularity on Performance**

**Granularity**

* **Definition**: Size of computation assigned to each processor (coarse-grained = larger chunks).
* **Scaling Down**: Using fewer processors (p) than the maximum possible (n).

**Virtual Processors**

* Simulate n virtual processors on p physical ones.
* **Cost Preservation**: If the original system is cost-optimal, the scaled-down version remains so.
* **Drawback**: Non-optimal systems stay non-optimal after scaling.

**5. Scalability of Parallel Systems**

**Challenges**

* Programs tested on small problems/machines may not scale to larger setups.
* **Example**: FFT algorithms show different speedups as problem size (n) increases (see Figure).

**Isoefficiency Metric**

* **Goal**: Maintain constant efficiency (E) as p increases by scaling problem size (n).
* **Isoefficiency Function**:
  + Rate at which n must grow with p to keep E fixed.
  + **Lower growth rate = Better scalability**.
* **Example**:
  + If n must grow as O(p log p) to keep E constant, the system is less scalable than one needing O(p).

**Visualization**

* **Isoefficiency Curves**: Plot problem size vs. processors for fixed efficiency.

**6. Other Scalability Metrics**

**Scaled Speedup**

* **Definition**: Speedup when problem size grows linearly with p.
* **Linear Scaled Speedup**: Indicates good scalability.
* **Methods**:
  1. **Memory-Bound Scaling**: Increase n to fill available memory (assumes memory grows with p).
  2. **Time-Bound Scaling**: Grow n with p while capping runtime.

**Real-Time Applications**

* **Constraint**: Fixed time bound (e.g., 40 ms/frame for MPEG decompression).
* **Goal**: Scale p to meet deadlines.

**Memory-Constrained Scaling**

* **Example**: Matrix multiplication where problem size is limited by aggregate memory.

**Key Takeaways**

1. **Overheads**: Communication, idling, and excess computation degrade performance.
2. **Metrics**:
   * **Speedup (**S**)**: Measures parallel benefit.
   * **Efficiency (**E**)**: Tracks processor utilization.
   * **Cost**: Evaluates resource usage.
3. **Granularity**: Coarser tasks reduce overhead but may limit parallelism.
4. **Scalability**:
   * **Isoefficiency**: Quantifies how n must grow with p.
   * **Scaled Speedup**: Tests real-world scaling behavior.
5. **Trade-offs**: No single metric captures all aspects; choose based on application (e.g., real-time vs. memory-bound).

**Example Calculations**

1. **Speedup**:
   * If Tₛ = 100s and Tₚ = 20s on p = 10 processors:

S=10020=5(sublinear)*S*=20100​=5(sublinear)

1. **Efficiency**:

E=510=0.5*E*=105​=0.5

1. **Cost**:

Cost=10×20=200(compare to Ts=100)Cost=10×20=200(compare to *Ts*​=100)

This explanation ensures **no content is omitted**, with **clear definitions** of technical terms (e.g., isoefficiency, granularity) and **structured formatting** for study purposes. Let me know if you'd like additional clarifications or examples!