SOFT 437

Performance Analysis

Chapter 9: Performance-Oriented Design

Performance Principles

- Identify design alternatives likely to meet performance objectives
- Generalized and abstracted knowledge used by experienced performance specialists when constructing software systems

Role of Quantitative Techniques

- Performance improvements involve tradeoffs
- Quantitative SPE techniques
  - provide data to evaluate the net effect of a design alternative
  - Weigh the performance improvement of one alternative against its effects on other **quality attributes**

3 categories of performance principles

- **Performance control principles**
- **Independent principles**
- **Synergistic**

3 categories of performance principles

- **Performance control principles**
- **Independent principles**
- **Synergistic**

Helps to **control** the performance of the system as it is being developed

3 categories of performance principles

- **Performance control principles**
  - performance objectives
  - instrumenting
- **Independent principles**
  - centering
  - fixing-point
  - locality
  - processing vs. frequency
- **Synergistic**
  - shared resources
  - parallel processing
  - spread-the-load

3 categories of performance principles

- **Performance control principles**
  - control performance by explicitly stating the required performance rigorously enough so that you can quantitatively determine whether or not the sw system meets the objective
- **Independent principles**
  - They can be applied independently; they don't conflict.
- **Synergistic**
  - They improve the overall performance of a system via cooperation among processes competing for computer resources.
- Summary, section 9.6, p. 259

Performance Objectives Principle -- (performance control principles)

- *Define specific, quantitative, measurable performance objectives for performance scenarios*
- Example:

The end-to-end time for completion of a typical correct ATM cash withdrawal performance scenario must be less than 1 minute and a screen result must be presented to the user within 1 second of the user's input

- May vary dependent on workload, etc.

Performance Objective Principle (con't)

- Performance objectives change over product's lifetime
- Consider future uses, if possible; anticipate future
- If models are built now; can be reused with modification to reflect system changes

Instrumenting Principle -- (performance control principles)

- Instrumenting software
  - Inserting code (probes) at key points to enable measurement of pertinent execution characteristics
- Example??
- Steps for instrumentation
  - inserting probes
  - activating them to record the data
  - analyzing and reporting the results

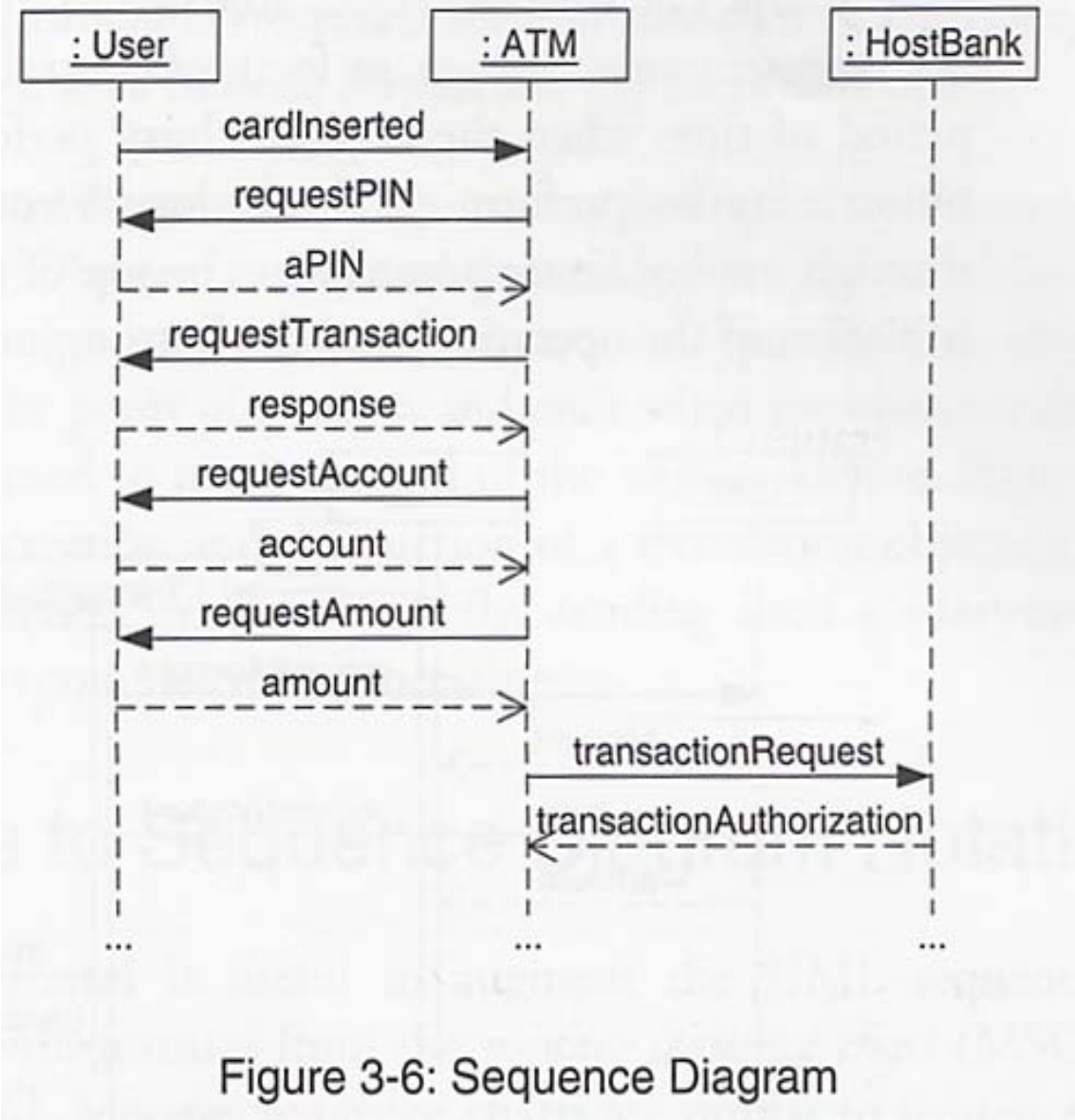

Figure 3-6: Sequence Diagram

Probe role

- Does not directly improve performance
  - may slightly decrease it
- Essential to improving performance
- Much easier to *design* probes into the system
- Multi-programming environments
  - measurements of time spent in separate tasks/components
  - enables measuring end-to-end user tasks

Centering principle – independent principles

- Identify the dominant workload functions and minimize their processing
  - focuses attention on the software parts that have the greatest impact on performance
  - 80-20 rule of code w.r.t. computer resource usage
    - extends to demand for system functions
- Identify the subset of system functions that will be used most of the time
- Improving their performance has significant impact on overall performance of the system

Dominant workloads ... performance scenarios

- Those that have the greatest number of requests
  - dominate user's perception of performance
- Consider also those with large resource demands -- may interfere with the more frequently executing functions
- Those whose performance is critical, such as hard deadlines
- (This is first step in the SPE process)

How minimize processing?

- For the dominant workload functions, create special streamlined execution paths
- These may be “trivial transactions”, not requiring much creative design
  - tendency therefore to defer work on these designs until the interesting parts have been specified
  - implementation of the “interesting parts” may have implied a design with constraints on data organization or other design constraints

What is “fixing”?

- Fixing in the sense of anchoring
- **Fixing** connects:
  - the desired *action* to the *instructions* used to accomplish that action
  - the desired *result* to the *data* used to produce it

Fixing-Point Principle
-- independent principles

- The *fixing point* is a point in time.
- The latest fixing point is during execution, just before instructions are to be executed
- *For responsiveness, fixing should establish data connections at the earliest feasible point in time, such that retaining the connection is cost-effective.*

Did you ever have to finally decide?

- Early fixing may reduce the flexibility of your design
- “Flexibility” may be an excuse for not addressing how users need to use the system and making a decision
- You may need to compromise on flexibility to achieve performance objectives

Example of changing fixing point ...

- Banker needs summary data of detailed records from multiple accounts
- Latest possible fixing point of the data
  - summarize the data when it is requested
  - operational cost is?
- Earlier fixing point
  - updating the summary data as the account detail records arrive
  - Retention cost is ?
  - “Update summary as details arrive” operational cost?

More Examples

- Add new items to a collection
  - Sort the collection when a new item arrives
  - Sort the collection when the sorted data is required
- Web cache

Locality Principle

- Locality: Closeness of desired actions, functions, and results to the physical resources used to *produce* them.
- Types of locality
  - Spatial
  - Temporal (i.e., time)
  - Effectual (i.e., purpose or intent)
  - Degree (i.e., intensity or size)

Locality Principle
-- independent principles

- *Create actions, functions, and results that are close to physical computer **resources**.*
- Tradeoffs:
  - consider effectual locality and portability
- Examples:
  - Multiple queries to the remote databases
  - Regional offices vs. central offices

Processing vs. Frequency Principle

- Concerned with the amount of work done in processing a request and the number of requests received.
  - seeks to make a trade-off between the two
  - may be possible to reduce # of requests by doing more work per request
- *Minimize the product of processing times frequency*
- Application of the Fixing-Point Principle
  - Example: create thread pool

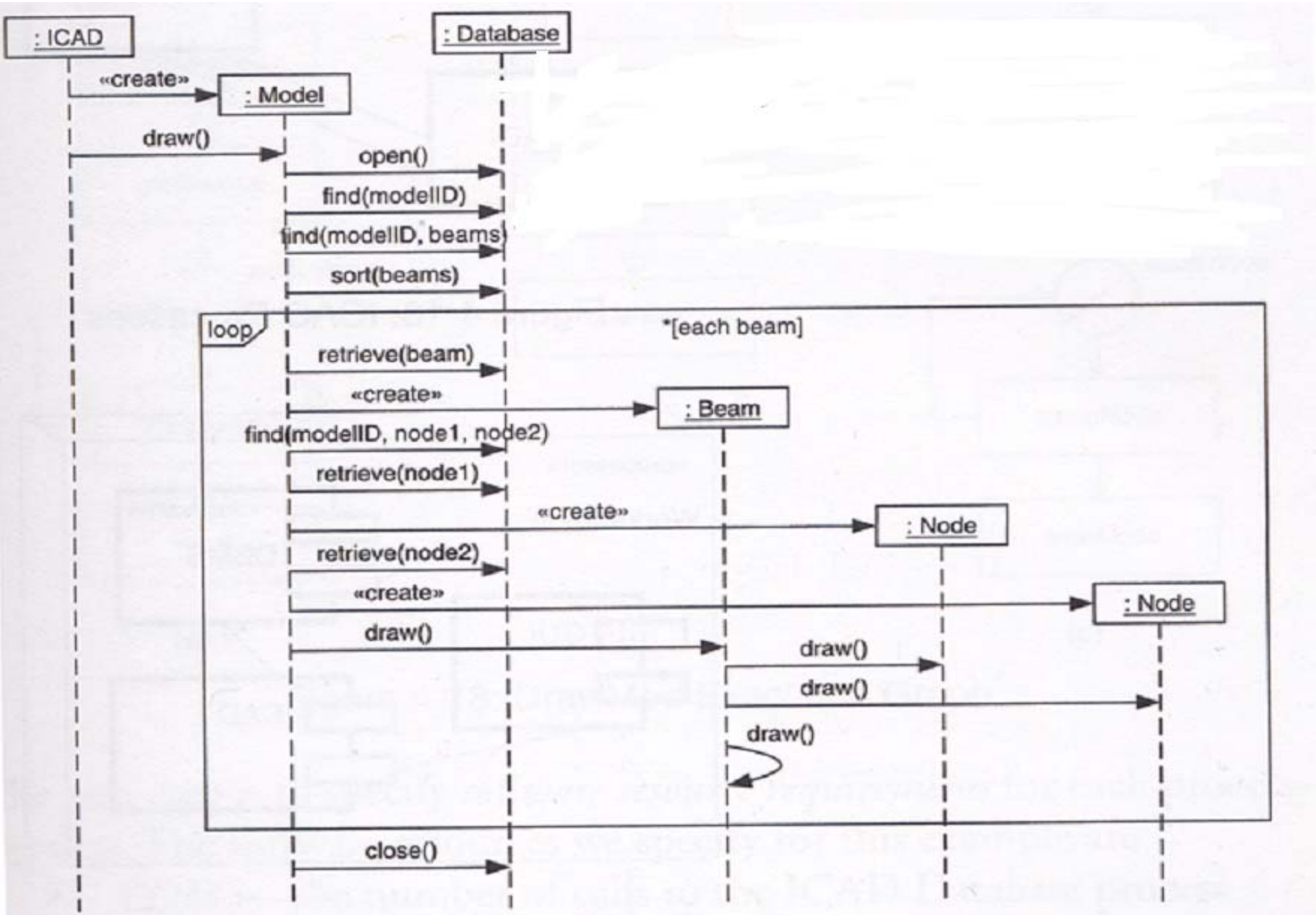

Figure 4-15: Expanded DrawMod Scenario

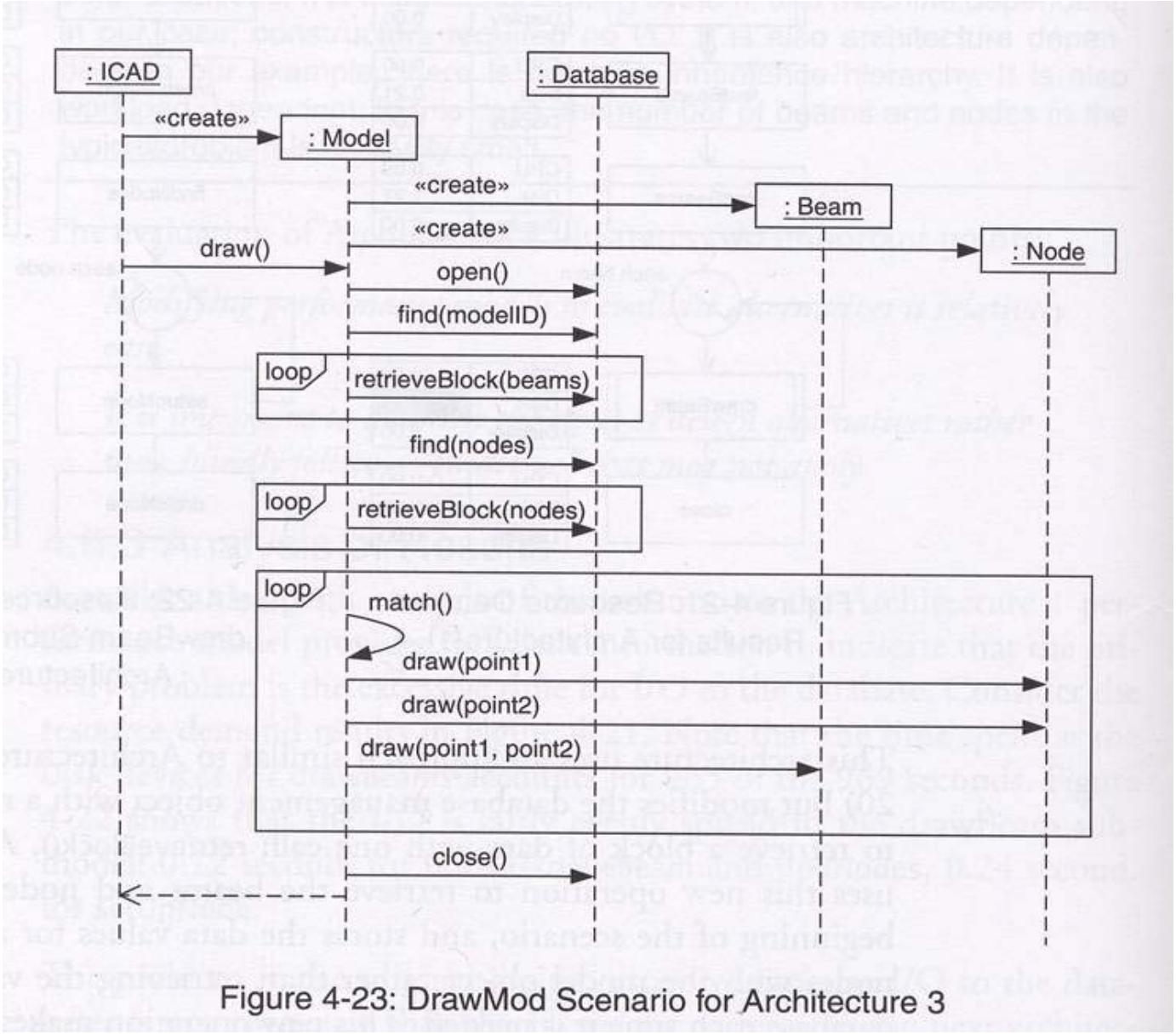

Figure 4-23: DrawMod Scenario for Architecture 3

Synergistic Principles

- Improve overall performance via cooperation among processes competing for computer resources
- Depend on cooperation to reduce delays for resource contention -- if all objects do not cooperate, may not achieve the desired improvement
- They are:
  - Shared resources
  - Parallel processing
  - Spread-the-load

Shared Resources Principle

- *Share resources when possible. When exclusive access is required, minimize the sum of the holding time and the scheduling time.*
- Need for exclusive access to a resource
  - additional processing overhead is needed to schedule access to the resource
  - potentially a contention delay as processes wait their turn

Minimizes *what*?

- Lock the entire database while being updated
  - minimizes *scheduling time*
  - requires less overhead -- check lock indicator, not whether it applies to individual record
- Lock the individual record
  - minimizes *holding time* (other processes can access other records)
  - maximizes *scheduling time* (separate lock status indicator for each record)

More Examples?

- Use phone numbers to map the subscribers

Parallel Processing Principle

- Overall processing time can *sometimes* be reduced by partitioning a computation into multiple concurrent processes.
- *Real concurrency* -- processes execute simultaneously on different processors. Processing time is reduced by an amount proportional to the number of processors.
- *Apparent concurrency* -- processes are multiplexed on a single processor. While some processing may be overlapped, each process will sometimes experience additional wait time due to contention for the same resource.

Real and Apparent concurrency

- Require processing overhead for the communication and coordination among the concurrent processes.
- This overhead can exceed the time saved by partitioning the computation into concurrent processes.
- *Parallel processing principle is: Execute processing in parallel when the processing speedup offsets communication overhead and resource contention delays.*

Where applied?

- Batch oriented jobs such as printing statements for a large number of customers
  - printed at night to minimize conflicts with daily processing (what principle is that?)
  - jobs must complete within the batch window
  - large jobs: partition the jobs and run several in parallel

Situation

- When multiple processes require exclusive use of one or more resources, can reduce these resource contention delays if you can…
  - schedule the processes so that they do not use the resource at the same time
  - divide the resource so that the processes use distinct parts of the resource and thus do not need the *same* resource

Spread-the-Load Principle

- *Spread the load when possible by processing conflicting loads at different times or in different places.*
- Similar to shared resources principle
- SRP minimizes scheduling time and holding time
- Reduce the number of processes needing the resource at a given time
- Reduce the amount of the resource they need

Using the principles

- Apply the principles to sw components that are critical to performance – no time for more!
- Use performance models to quantify the effect of improvements on overall performance
- Apply principles until you comply with performance objectives
- Confirm that performance objectives are realistic and that it is cost effective to achieve them
- Create a customized list of examples of each principle specific to your application domain -- publicize this to others in the same domain
- Document and explain performance improvements using the principles