Tactics for Performance

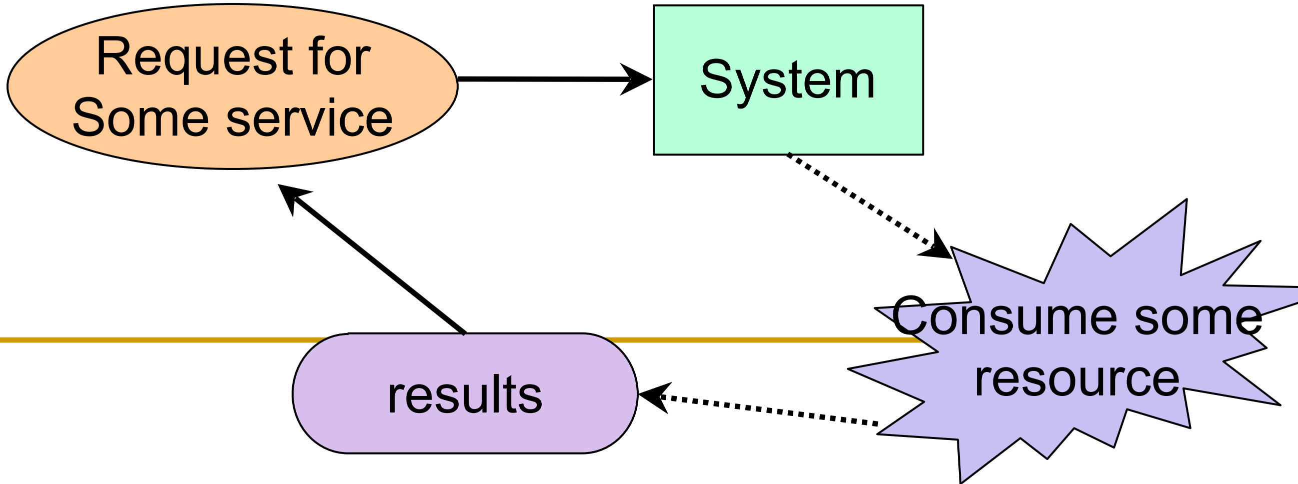
主讲教师：王灿
Email: wcan@zju.edu.cn
TA: 李奇平 liqiping1991@gmail.com
Course FTP: ftp://sa:sa@10.214.51.13

Performance Evaluation for Various Systems

- Performance is generally concerned with how long it takes the system to respond when an event occurs
- Two typical performance scenarios
  - A Web-based financial system
    - Events coming from numerous users
    - Response measured by transaction per minute
  - An engine control system
    - Events coming from a timer internal to the system
    - Response measured by variation in response time

Server-side Perspective

- Characterizing patterns of events arriving and patterns of responses
  - Multiple users or other loading factors can be modeled by varying the arrival patterns for events
    - What matters is the arrival pattern at the server and dependencies within the requests
  - The response can be characterized by latency, the throughput of the system etc.



Event Arrival Patterns

- Event arrival patterns
  - Periodic
    - E.g., an event arriving every 10 ms in a real-time system
  - Stochastic
    - Events arrive according to some probabilistic distribution
  - Sporadic
    - Irregular event arrival patterns

Response Measures

- Response can be measured by:
  - Latency
    - The time between the arrival of the stimulus and the system’s response to it
  - Deadlines in processing
    - The event shall be processed within a deadline
  - Throughput
    - E.g., transactions per minute
  - Jitter
    - The variation in latency
  - Events not processed when the system was too busy (miss rate)

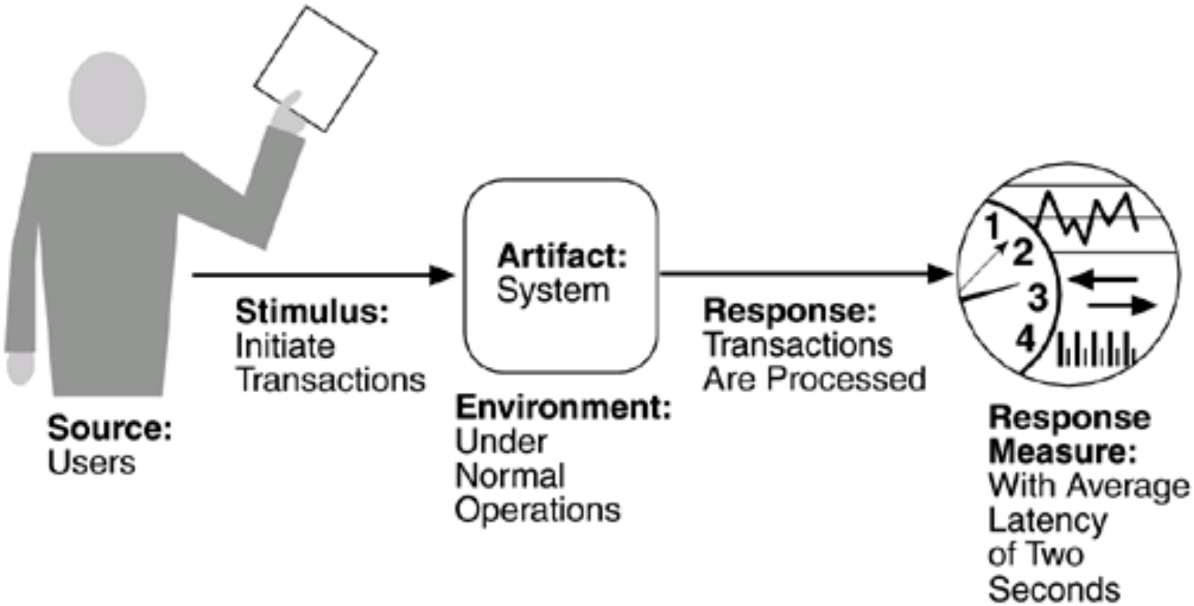
Performance General Scenario (1)

- Stimulus: event arrivals with patterns recognized as
  - Periodic
  - Stochastic
  - Sporadic
- Source of stimulus
  - External (possibly multiple) or internal sources
- Response
  - Processing the arriving events; may cause a change in system environment

Performance General Scenario (2)

- Response Measure
  - Latency, deadline, throughput, jitter, *miss rate, data loss*…
- Artifact
  - The system or one or more of its components
- Environment
  - Normal mode
  - Overloaded mode
  - Emergency mode

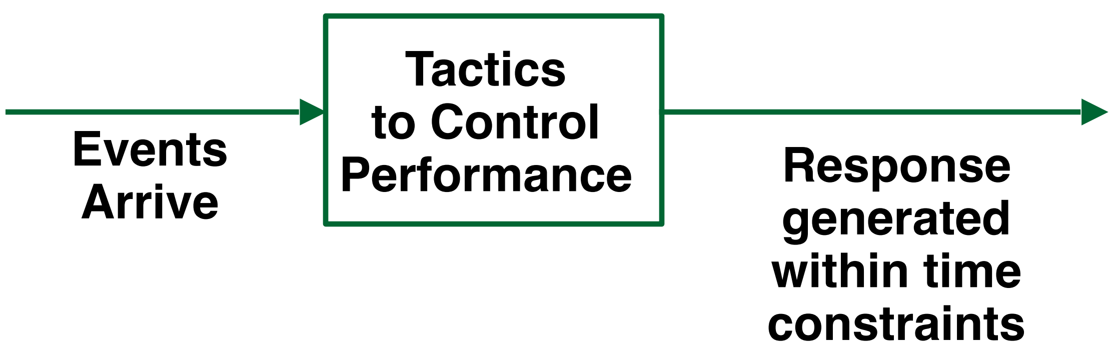
A Sample Concrete Performance Scenario



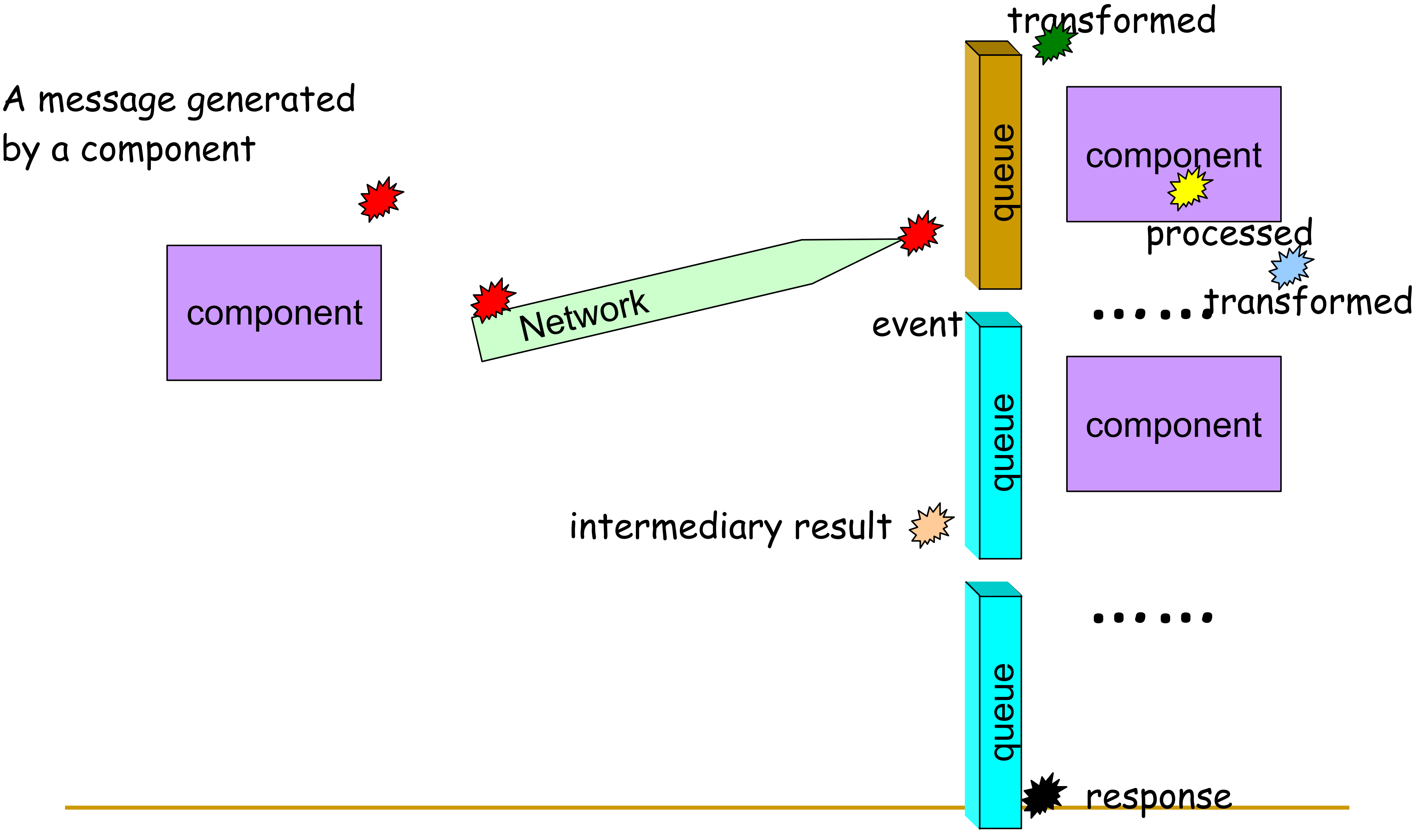
"Users initiate 1,000 transactions per minute stochastically under normal operations, and these transactions are processed with an average latency of two seconds."

Performance Tactics

- The goal of the performance tactics is:
  - To generate a response to an event within some time constraint



Processing Sequence of an Event



Two Contributors to the Response Time (1)

- When an event arrives it is either processed or blocked for some reason.
  - Response time = *Processing time* **+** *blocked-time*
- Processing time: time in consuming resources
  - Processors, data stores, network bandwidth, memory ...
  - System specific resources:
    - Buffers allocated
    - Critical sections (must be accessed sequentially)

Two Contributors to the Response Time (2)

- Block time: a computation can be blocked because:
  - Contention for resources
    - Resource to be used by a single client at a time VS. multiple streams of events
    - Generally, more event streams → more contention → more latency
      - This also depends on the arbitration mechanism for contention

Two Contributors to the Response Time (3)

- A computation can be blocked because:
  - Availability of resources
    - Resource unavailability (e.g., offline, crash, etc.) contributes to latency
  - Dependency on other computation
    - Synchronization
    - Waiting for the results of another computation

Performance Tactics

- The two contributors of the response time are both concerned with resources, either consuming it or awaiting it
- Consequently all our performance tactics are about resources
  - Control resource demand
  - Manage resources

Performance Tactics Hierarchy

Events Arrive

Performance Tactics

- **Control Resource Demand**
  - Manage Sampling Rate
  - Limit Event Response
  - Prioritize Events
  - Reduce Overhead
  - Bound Execution time
  - Increase Resource Efficiency
- **Manage Resources**
  - Increase Resources
  - Introduce Concurrency
  - Maintain Multiple Copies of Computations
  - Maintain Multiple Copies Of Data
  - Bound Queue Sizes
  - Schedule Resources

Responses Generated within Time Constraints

Control Resource Demand: Manage Sampling Rate

- Reduce the resource demand by reducing the data sampling frequency (in signal processing systems)
  - May result in loss of fidelity
  - E.g. using different codecs to ensure smooth video playback
    - Tradeoff: latency VS. fidelity

Control Resource Demand: Limit Event Response

- When discrete events can not be "downsampled", process events only up to a set maximum rate
  - Used when a queue size or processor utilization measure exceeding some warning level
- Two situations for this tactic
  - Unacceptable to lose any events: use a queue large enough
  - When choose to drop events, corresponding handling policy: log & notify?

Control Resource Demand: Prioritize Events

- Use a priority scheme to rank events according to their importance
  - Ignore low-priority events when the system is overloaded

- E.g. fire alarms VS. informational alarms such as a room is too cold in a building management system

Control Resource Demand: Reduce Overhead

- Reduce computational overhead
  - ***Use of intermediaries*** increases resource consumption in processing events: the classic modifiability/performance trade-off
  - ***Separation of concerns*** increase the processing overhead in that an event is serviced by a chain of components instead of a single component: modifiability/performance trade-off
- Reduce communication overhead
  - Co-locate resources: hosting cooperating components on the same processor to avoid the time delay of network communication

Control Resource Demand: Bound Execution Times

- Place a limit on how much execution time is used to respond to an event
  - E.g., limit the number of iterations for iterative, data-dependent algorithms:
    - performance/accuracy tradeoff
- Frequently paired with the manage sampling rate tactic

Control Resource Demand: Increase Resource Efficiency

- Improving the algorithms used in critical areas will decrease latency

Manage Resources: Increase Resources

- Faster processors, more processors
- Additional memories
- Faster network
- …
- Cost/performance trade-off

Manage Resources: Introduce Concurrency

- Processing requests in parallel to reduce the blocked time.
  - Multiple threads
  - Paired with scheduling policies (maximize fairness, throughput etc.)

Two threads execute the following statements currently. What is the value of `x` after both threads have executed the same statements?

```
x :=1;
x++;
```

Manage Resources: Maintain Multiple Copies of Computations

- E.g. multiple servers in a client-server pattern
  - A *load balancer* will assign tasks with varying criteria such as round-robin or assigning the next request to the least busy server.

Manage Resources: Maintain Multiple Copies of Data

- ***Caching*** (memory hierarchy) : same data replicated on different repositories
  - Data synchronization and consistency is an important issue here
  - Another responsibility is to choose the data to be cached
- ***Data replication*** involves keeping separate copies of the data to reduce the contention (e.g. P2P downloading)

Manage Resources: Bound Queue Sizes

- Put a limit on the maximum number of queued events
  - Need a policy for what happens when the queues overflow

- This tactic is frequently paired with the limit event response tactic

Manage Resources: Schedule Resources (1)

- Whenever there is contention for a resource, the resource must be scheduled
- Two parts of a scheduling policy: priority assignment & dispatching
- Scheduling criteria
  - Optimal resource usage, request importance, minimizing latency, maximizing throughput, preventing starvation to ensure fairness…
- Preemption
  - Anytime
  - Only at special moment
  - Not preempting executing processes

Manage Resources: Schedule Resources (2)

- First-in/First-out: treats all requests for resources as equals and satisfy them in turn
  - Request priority is not considered
  - A request might be stuck for a long time
- Fixed-priority scheduling: a fixed priority is assigned to each request and that assignment will remain unchanged.
  - High-priority requests are better served
  - Low-priority requests may wait for an arbitrarily long time
  - Common prioritization strategies:
    - Semantic importance
    - Deadline monotonic – higher priority to streams with shorter deadlines
    - Rate monotonic - higher priority to streams with shorter periods

Manage Resources: Schedule Resources (3)

- Dynamic-priority scheduling: the priority of the event requests are calculated during the execution of the system
  - Round robin
    - Event requests are refreshed after the requests are served one term
  - Earliest deadline first
    - Refresh priority according to the remain deadline of the events
- Static Scheduling: pre-emption points and the sequence of assignment to the resource are determined offline.
  - E.g. a cyclic executive schedule

Architectural Design Support for Performance

- We check the architectural design and analysis process for performance from the following 7 aspects:
  1. Allocation of responsibilities
  2. Coordination model
  3. Data model
  4. Management of resources
  5. Mapping among architectural elements
  6. Binding time decisions
  7. Choice of technology

Allocation of Responsibilities

- Identify heavy loading, time-critical and heavily used responsibilities
- Responsibilities to support:
  - Thread control
  - Resource scheduling
  - Managing queues, buffer, caches etc.

Data Model

- Determine the portion of data model that are heavy loading, time-critical and heavily used and consider the following for this portion:
  - Maintain multiple copies of data
  - Data partitioning

Mapping among Architectural Elements

- Co-locating: runtime components → hardware infrastructure
- Resource allocation
- Introduce concurrency: a piece of functionality → multiple copies of a component running simultaneously

Resources Management

- Monitor and manage performance-critical resources
  - Prioritization
  - Access
  - Scheduling
  - Locking
  - On-demand deploying

Binding Time Decisions

- Time necessary to complete the binding
- Overhead of using late binding

Choice of Technology

- Does the tech. meet real-time requirements?
- Does it support:
  - Scheduling policy
  - Priorities
  - Allocation of portions of the technology to processors
- Does your choice of technology introduce excessive overhead for heavily used operations?

Assignment

- Identify architectural tactics
  - The assignment description and the related paper have been uploaded to the course FTP

Reading Assignment

- Read Chapter 9 & 10 of the textbook.