CptS 487
Software Design and Architecture

Lesson 18 (pt2)

Quality Attributes 2

Availability & Performance

WASHINGTON STATE UNIVERSITY

World Class. Face to Face.

Instructor: Bolong Zeng

Outline

- Availability
- Performance

Availability

- “Availability refers to the ability of a system to mask or repair faults such that the cumulative service outage period does not exceed a required value over a specified time interval.”

- Closely related to:
  - Security
  - Performance
  - Safety

- Minimizing service outage time by mitigating faults.

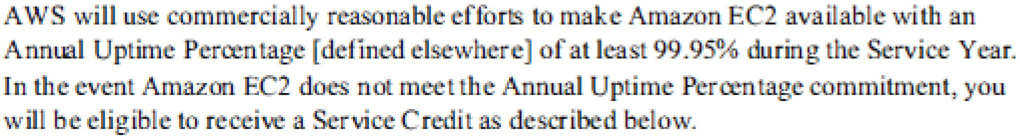
Failure and Fault

- Failure
  - The deviation of the system from its specification
  - and it’s externally visible.

- Fault
  - The cause of the failure
    - Can be either internal or external.
  - Distinction between fault and failure also allows “automatic repair strategies”: recover from the fault before a failure happens.

Calculate Availability

- The probability that the system will provide the specified services within required bounds over a specified time interval.
  - E.g. “Downtime no longer than 1 hours per 30 days of continuous operating”
- In hardware: $\frac{MTBF}{MTBF + MTTR}$
  - (mean time between failures and mean time to repair)
- In software:
  - It should mean the likelihood of faults occurring, and mean time required for repair.
  - Typically referred in Service-level agreements (SLAs)

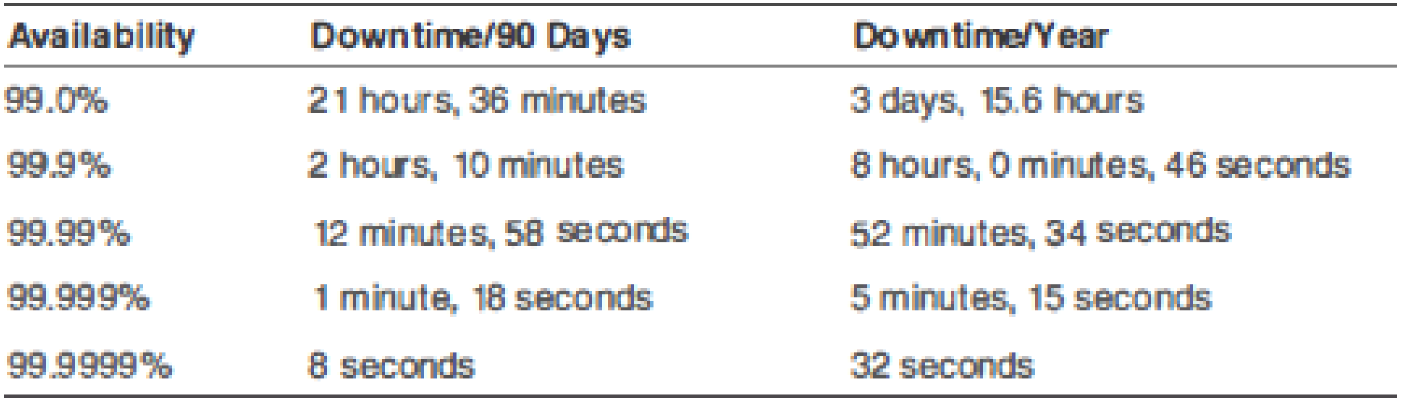


AWS will use commercially reasonable efforts to make Amazon EC2 available with an Annual Uptime Percentage [defined elsewhere] of at least 99.95% during the Service Year. In the event Amazon EC2 does not meet the Annual Uptime Percentage commitment, you will be eligible to receive a Service Credit as described below.

Calculate Availability

- Example
  - 99.0% sounds good, but…

Table 5.1. System Availability Requirements



Availability	Downtime/90 Days	Downtime/Year
99.0%	21 hours, 36 minutes	3 days, 15.6 hours
99.9%	2 hours, 10 minutes	8 hours, 0 minutes, 46 seconds
99.99%	12 minutes, 58 seconds	52 minutes, 34 seconds
99.999%	1 minute, 18 seconds	5 minutes, 15 seconds
99.9999%	8 seconds	32 seconds

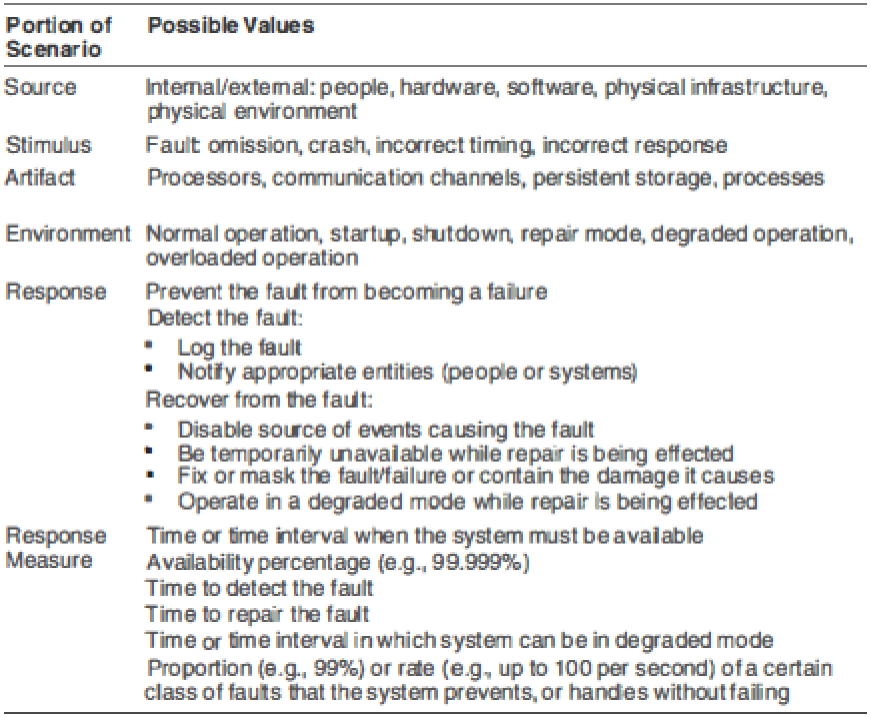
Recommended Reading

- [BASS] page 82-85
  - Side bar: Planning for Failure
  - Fault tree and calculation of fault probability

- To proceed, have a software system in mind while learning about the QAs.
  - Use a real example to relate to the contents.
  - Prepare for your final take home exam too.
  - Most importantly, try to look at your example from a new “design” and “quality” perspective.

Discussion of Availability

- General Scenario



Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	Processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: ▪ Log the fault ▪ Notify appropriate entities (people or systems) Recover from the fault: ▪ Disable source of events causing the fault ▪ Be temporarily unavailable while repair is being effected ▪ Fix or mask the fault/failure or contain the damage it causes ▪ Operate in a degraded mode while repair is being effected
Response Measure	Time or time interval when the system must be available Availability percentage (e.g., 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

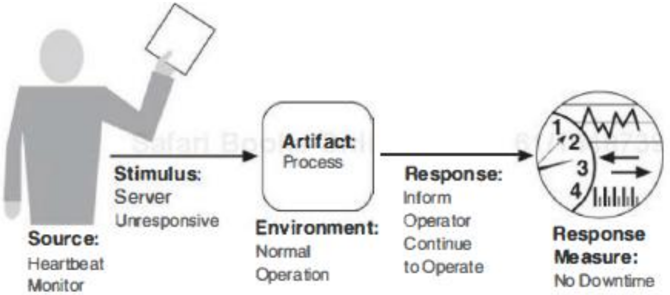


Figure 5.3. Sample concrete availability scenario

- Concrete Scenario
  - Example

Discussion of Availability

- Tactics
  - Could be categorized as addressing one of 3 categories:
    - Fault detection
    - Fault recovery
    - Fault prevention

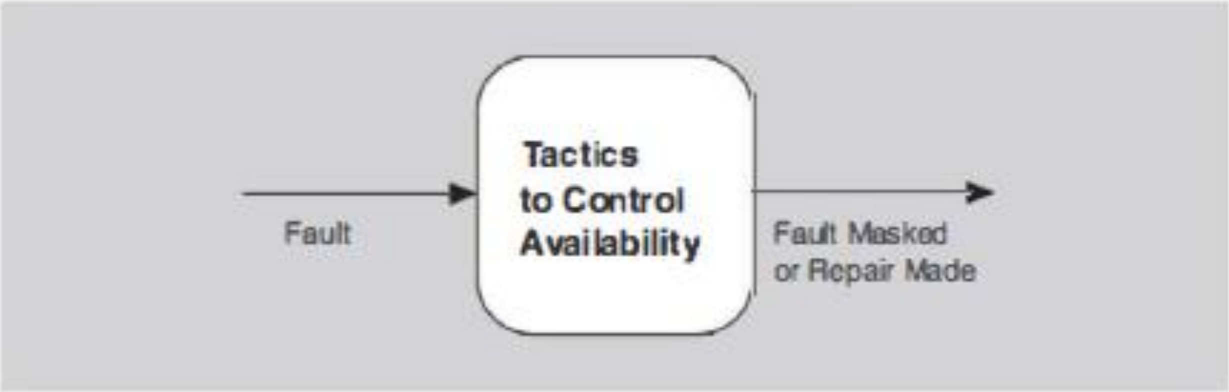


Figure 5.4. Goal of availability tactics

Detect Faults

- Ping/Echo
- Monitor
- Heartbeat
- Timestamp
- Sanity checking
- Condition monitoring
- Voting
- Exception Detection
- Self-test

Recover from Faults

- **Preparation-and-repair**
- Active redundancy
- Passive redundancy
- Spare
- Exception handling
- Rollback
- Software upgrade
- Retry
- Ignore faulty behavior
- Degradation
- Reconfiguration

- **Reintroduction**
- Shadow
- State resynchronization
- Escalating restart
- Non-stop forwarding

Prevent Faults

- Removal from service
- Transactions
- Predictive model
- Exception prevention
- Increase competence set

Summary on Tactics

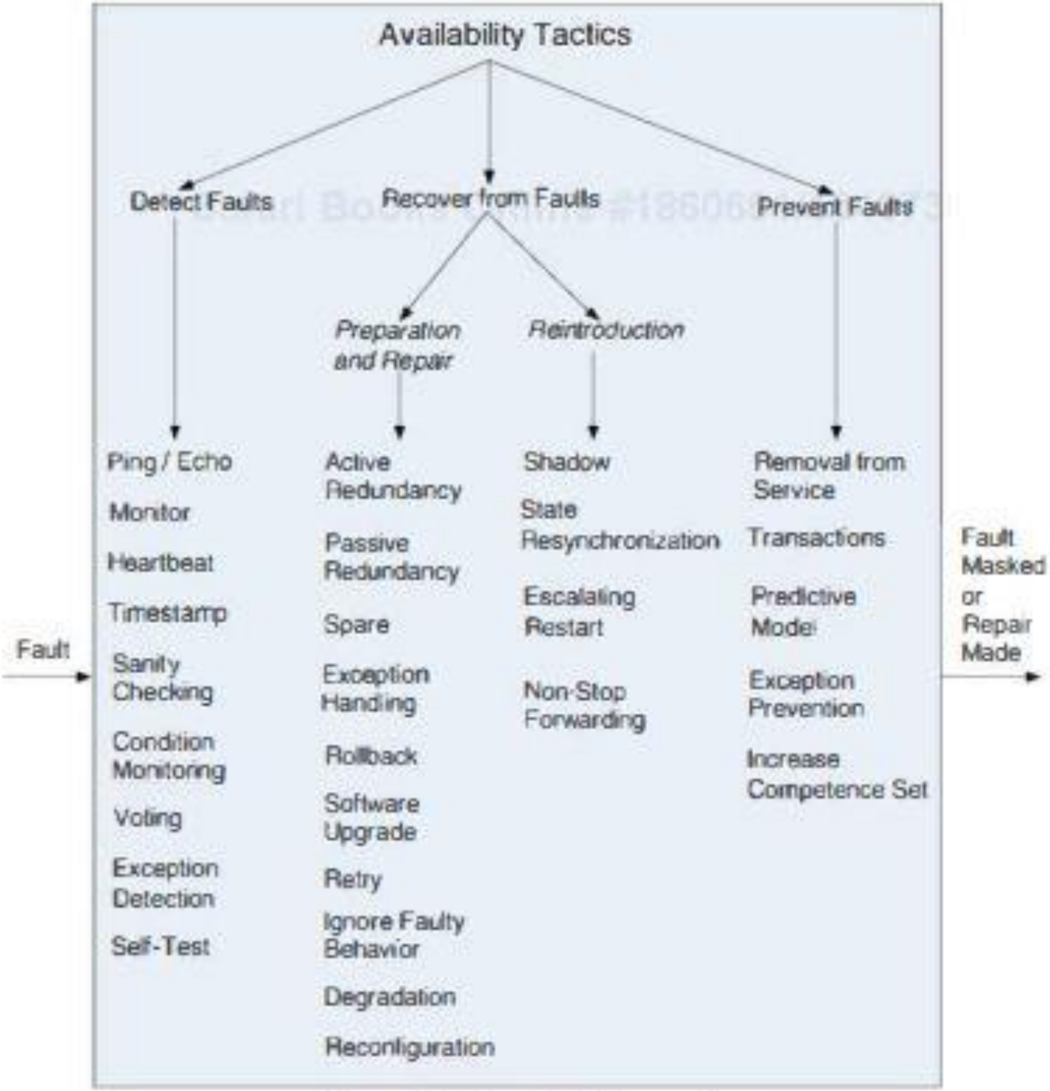


Figure 5.5. Availability tactics

Design Checklist for Availability

- Determine the system responsibilities that need to be highly available.
  - Allocation of responsibilities
  - Coordination model
  - Data model
- Determine which artifacts are prone to producing fault
  - Mapping among architectural elements
- Determine critical resources necessary to continue operating when there is a fault
  - Resource management
- Determine how and when elements are bound
  - Binding time
- Determine on available technologies that help
  - Choice of technology

Performance

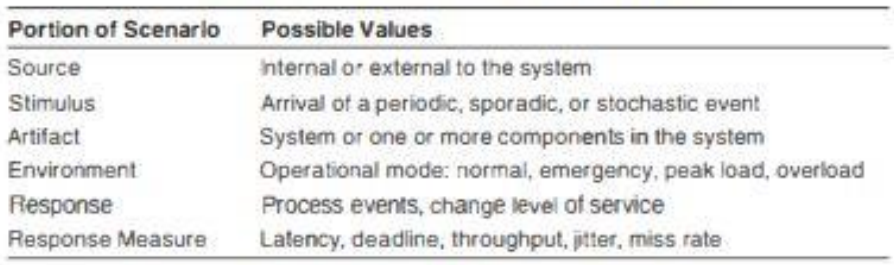
- “It’s about time”
  - And the software system’s ability to meet timing requirements.

- Closely related to:
  - Scalability
    - “making your system easy to change” – hence a kind of modifiability.

Discussion of Performance

- General Scenario
  - Begins with an event arriving
    - Periodic
    - Stochastic
    - sporadic
  - Responding correctly requires TIME (and other resources)
    - Latency
    - Deadlines in processing
    - Throughput
    - Jitter
    - Number of events not processed

Table 8.1. Performance General Scenario



Portion of Scenario	Possible Values
Source	Internal or external to the system
Stimulus	Arrival of a periodic, sporadic, or stochastic event
Artifact	System or one or more components in the system
Environment	Operational mode: normal, emergency, peak load, overload
Response	Process events, change level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate

Discussion of Performance

- Concrete Scenario
  - Example

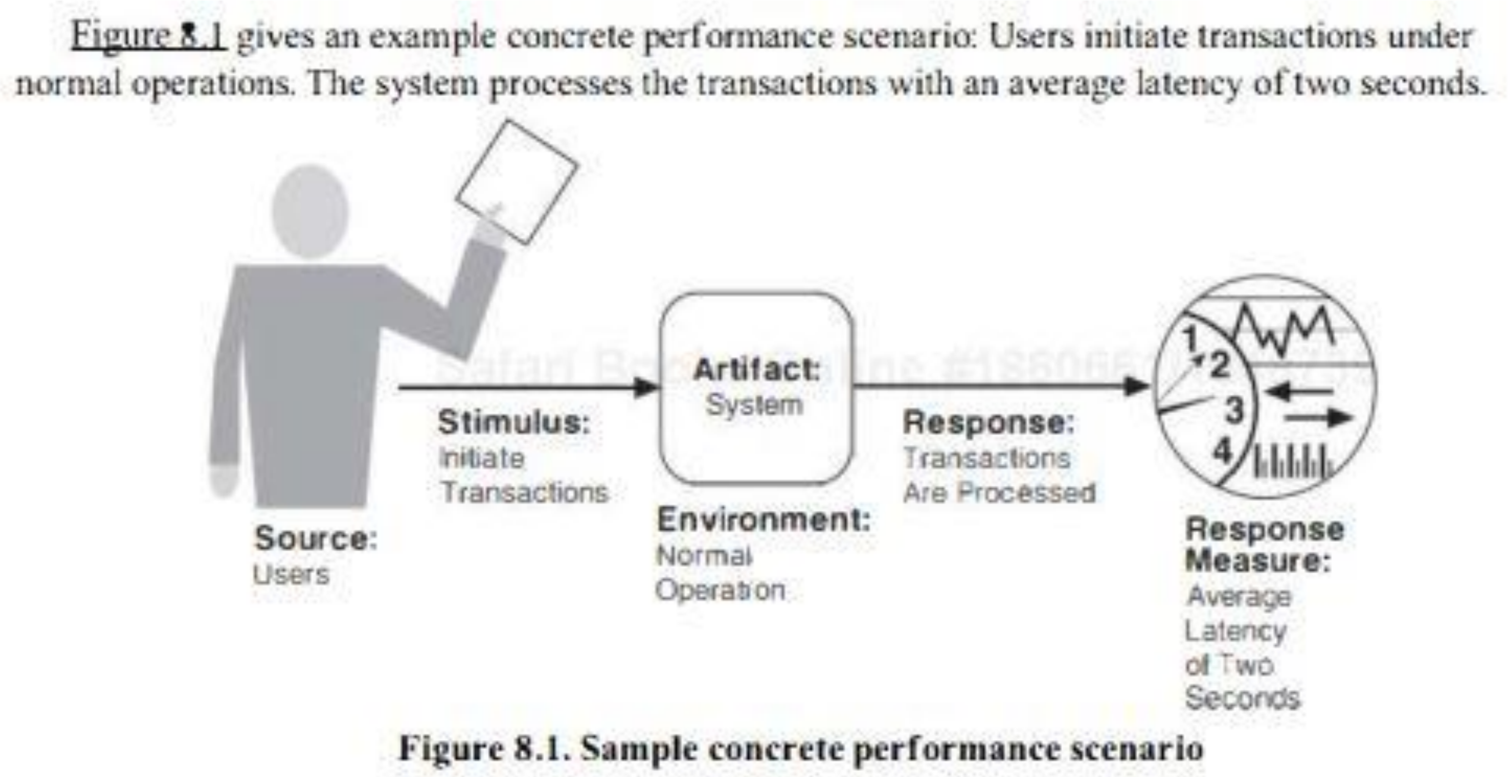


Figure 8.1 gives an example concrete performance scenario: Users initiate transactions under normal operations. The system processes the transactions with an average latency of two seconds.

Source:
Users
Stimulus:
Initiate Transactions
Artifact:
System
Environment:
Normal Operation
Response:
Transactions Are Processed
1
2
3
4
Response Measure:
Average Latency of Two Seconds

Figure 8.1. Sample concrete performance scenario

Discussion of Performance

- Tactics
  - Processing time
  - Blocked time
    - Contention for resources; Availability of resources; Dependency on other computations
  - 2 categories:
    - Control resource demand
    - Manage resources

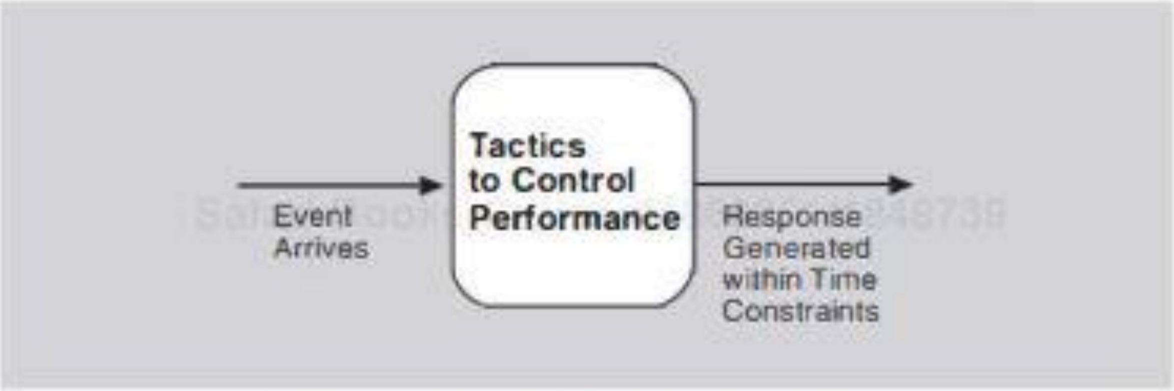


Figure 8.2. The goal of performance tactics

Control Resource Demand

- Manage sampling rate
- Limit event response
- Prioritize events
- Reduce overhead
- Bound execution times
- Increase resource efficiency

Manage Resources

- Increase resources
- Introduce concurrency
- Maintain multiple copies of computations
- Maintain multiple copies of data
- Bound queue sizes
- Schedule resources

Design Checklist for Performance

- Determine the responsibilities that will be heavily used
  - Allocation of responsibilities
  - Data model
- Determine heavy coordinate and communication
  - Coordination model
  - Mapping among architectural elements
- Determine resources critical for performance
  - Resource management
- For element bound after compilation, determine binding time and overhead
- Determine if the technology allows you to set and meet deadlines, and give you ability to execute the tactics.