Carnegie Mellon University

Availability

Outline

- **Introduction**
- Dealing with failure

Availability

- Availability is defined as the degree to which a system or component is operational and accessible when required for use
- Typically measured in terms of percentage. E.g. 99.9999% uptime.

Strategy to achieve high availability

- Anticipate causes of unavailability
- Prepare masking strategy for each cause
- Detect failure
- Execute masking strategy.

Causes of unavailability

- A failure occurring within the software. E.g. incorrect output of a component.
- A failure occurring in the environment. E.g. a VM running an instance of a component of the system fails or an availability zone in a region becomes inaccessible.
- A mismatch between the software and its environment. E.g. misinterpreting a sensor reading.

Masking a failure

- A masked failure is a failure that occurs in some portion of the system and the user continues to see the system as operational and accessible for use
- Masking a failure means
  - The failure has been anticipated and preparations made to deal with it.
  - The failure has been detected and the preparations that were made are executed

Redundancy

- Most preparations involve some type of redundancy
  - Duplicate code
    - Possibly independently developed
    - Possibly alternative method of achieving function
  - Duplicate executable
  - Duplicate data
  - Duplicate hardware
  - Duplicate requests
- Each of these forms of redundancy has its own considerations

Outline

- Introduction
- **Dealing with failure**

Detecting failure

- Two categories of symptoms
  - No response from a request within specified time.
  - Erroneous output of a component.
- For each symptom, there are questions
  - How is the symptom manifested?
  - What preparations will keep the symptom from causing a failure?
  - How is the symptom masked to keep the system from becoming unavailable?

Outline

- Introduction
- **Dealing with failure**
  - **Symptom - Slow or no response**

Detecting slow response

- In a distributed system, components communicate through messages.
  - Requestor sends a message
  - Recipient acts on message
- Requestor, when sending a message, sets a time out interval.
- If the interval times out, then the response is slow.
- Distinguishing between a slow response no response is not possible in a distrubted system.

Outline

- Introduction
- **Dealing with failure**
  - **Symptom - Slow or no response**
  - Possible causes
    - **Congestion**
    - Scheduling problem
    - Failure of an instance (stateful or stateless)
    - Failure in the environment

Congestion

- Some portion of the system is too busy
  - Network
  - Computational component
- Determined by examining
  - utilization measures (CPU or network) or
  - queue lengths within a computational component

Masking techniques for congeston

- Retry
  - Works if the problem is transitory.
  - The requestor must be prepared to retry.
- Scale hardware
  - works if the problem is overloaded instance.
  - The instance must be under control of autoscaling infrastructure, or the delay will be observable. Pre-allocating additional instances will shorten delay.

Masking techniques for congestion

- Proceed without requested data. E.g. in a GPS system, if the data from the satellites are not received, proceed with dead reckoning based on other data.
- Over-request. If multiple resources are being allocated, request more than needed and cancel the remaining requests once the necessary amount is reached.

Outline

- Introduction
- Dealing with failure
  - Symptom - Slow or no response
  - Possible causes
    - Congestion
    - **Scheduling problem**
    - Failure of an an instance (stateful or stateless**)**
    - Failure in the environment

Scheduling problem

- In real time systems, threads are prioritized.
- Priority inversion causes lower priorities to be run ahead of higher priorities

Priority inversion

- Priority inversion is detected by looking at priorities of various threads vs execution of those threads
- Priority inversion is prevented by using appropriate scheduling mechanism.

Outline

- Introduction
- Dealing with failure
  - Symptom - Slow or no response
  - Possible causes
    - Congestion
    - Scheduling problem
    - **Failure of an instance (stateless or stateful)**
    - Failure in thevironment

Failure of an instance

- Detected by performing health checks
  - Heartbeat
  - Ping/echo
- Requires a monitor that receives heartbeat or performs ping/echo.

Outline

- Introduction
- Dealing with failure
  - Symptom - Slow or no response
  - Possible causes
    - Congestion
    - Scheduling problem
    - **Failure of an instance (stateless)**
    - Failure in the environment

Masking failure of a stateless instance

- Allocate new instance on failure detection.
  - Health check monitor determines failure
  - Monitor initiates allocation of new instance
- Pre-allocate additional instances.
  - Keep spare instances available as back up.
  - Monitor
    - informs routing mechanism to stop sending traffic to failed instance and begin sending traffic to back up instance.
    - Allocates new back up instance

Routing traffic

- Whether a new instance is created or has been pre-allocated, requests must be routed to it.
- Routing mechanism uses a discovery service – DNS, Service Mesh, load balancer, or specialized type of discovery.
- New instance must be registered with discovery service and failed instance de-registered.

Masking failure of a stateless instance

- If all instances of a component have failed, have a fall-back computation available.
- E.g. if a personalized recommendation system fails, display popular choices.

Outline

- Introduction
- Dealing with failure
  - Symptom - Slow or no response
  - Possible causes
    - Congestion
    - Scheduling problem
    - **Failure of an instance (stateful)**
    - Failure in the environment

Stateful instance failure

- Masking a stateful instance failure is the same as masking a stateless instance failure plus state for the replacement instance must be acquired.
- A copy of the state for stateful instances must be kept external to the instance.
- Issues:
  - Performance costs of keeping copy of state
  - Consistency between state in the instance and the copy.

Outline

- Introduction
- Dealing with failure
  - Symptom - Slow or no response
  - Possible causes
    - Congestion
    - Scheduling problem
    - Failure of an instance
    - **Failure in the environment**

Failure in the environment

- A failure in the environment is masked by maintaining a copy of the components in the environment in a separate environment.
- The issues are
  - Performance cost of keeping the copy
  - Financial costs of keeping the copy
  - Data consistency
  - Distinguishing between the failure of an instance and the failure of the environment in which that instance exists.

Outline

- Introduction
- Dealing with failure
  - **Symptom – Erroneous output**

Erroneous output

- Detected by having validity checks on output of a component
- Validity checks may come from
  - Independent generators of the output. Compare the outputs of the independent generators.
  - *A priori* specification of reasonable ranges of output
  - Consistency checks of different variables.

Masking erroneous output

- Disable erroneous component.
  - Stop sending it traffic
  - Maintain fall back computation
- Roll back to prior, correct version of component
- If erroneous component does not send response on detection of error, then requestor will treat it as a failed instance.

Summary

- Anticipate causes of unitability
- Prepare masking strategy for each cause
- Detect failure
- Execute masking strategy.