Principles of Software Construction: Objects, Design and Concurrency

Distributed System Design, part 1

Jonathan Aldrich

Charlie Garrod

© 2012 C Garrod, J Aldrich, and W Scherlis
Adminstrivia

- Homework 8 due Thursday night
  - Don't forget to turn in work from Lab 8
Last time: Software product lines

- Correctness, verifiability, clarity in the context of large configuration spaces
Recall: Network communication in Java

• Abstractions of a network connection
• Packet-based communication
• Stream-based communication
• Remote Procedure Call
  ▪ Java Remote Method Invocation
Recall: Distributed systems

- Multiple system components (computers) communicating via some medium (the network)

- Challenges:
  - Heterogeneity
  - Scale
  - Geography
  - Security
  - Concurrency
  - Failures

- The redundancy/reliability curve:
Today: More distributed systems

- Motivation: reliability and scalability
- Failure models
- Techniques for:
  - Reliability (availability)
  - Scalability
  - Consistency
You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。
A case study: Passive primary-backup replication

- Architecture before replication:

  - Problem: Database server might fail
  - Solution: Replicate data onto multiple servers

- Example architecture:

  - Client
  - Front-end
  - Database server

  - Primary:
    - {alice:90, bob:42, ...}
  - Backup:
    - {alice:90, bob:42, ...}

- Example data:

  - {alice:90, bob:42, ...}
Passive primary-backup replication protocol

1. Front-end issues request with unique ID to primary DB

2. Primary checks request ID
   - If already executed request, re-send response and exit protocol

3. Primary executes request and stores response

4. If request is an update, primary DB sends updated state, ID, and response to all backups
   - Each backup sends an acknowledgement

5. After receiving all acknowledgements, primary DB sends response to front-end
Issues with passive primary-backup replication
• Many subtle issues with partial failures

• If primary DB crashes, front-ends need to agree upon which unique backup is new primary DB
  ▪ Primary failure vs. network failure?

• If backup DB becomes new primary, surviving replicas must agree on current DB state

• If backup DB crashes, primary must detect failure to remove the backup from the cluster
  ▪ Backup failure vs. network failure?

• If replica fails* and recovers, it must detect that it previously failed

• ...

Issues with passive primary-backup replication
More issues…

- **Concurrency problems?**
  - Out of order message delivery?
    - Time...

- **Performance problems?**
  - 2n messages for n replicas
  - Failure of any replica can delay response
  - Routine network problems can delay response

- **Throughput problems?**
  - All replicas are written for each update, but primary DB responds to every request
  - Does not address the scalability challenge
Aside: Facebook and primary-backup replication

- Variant for scalability only:
  - Read-any, write-all
  - Palo Alto, CA is primary replica

- A 2010 conversation:
  Academic researcher: What would happen if X occurred?
  Facebook engineer: We don't know. X has never happened...but it would be bad.
Types of failure behaviors

- Fail-stop
- Halting failures
- Communication failures
  - Send/receive omissions
  - Network partitions
  - Message corruption
- Performance failures
  - High packet loss rate
  - Low throughput
  - High latency
- Data corruption
- Byzantine failures
Common assumptions about failures

- Behavior of others is fail-stop (ugh)
- Network is reliable (ugh)
- Network is semi-reliable but asynchronous
- Network is lossy but messages are not corrupt
- Network failures are transitive
- Failures are independent
- Local data is not corrupt
- Failures are reliably detectable
- Failures are unreliably detectable
Some distributed system design principles

- **The end-to-end principle**
  - When possible, implement functionality at the end nodes (rather than the middle nodes) of a distributed system

- **The robustness principle**
  - Be strict in what you send, but be liberal in what you accept from others
    - Protocols
    - Failure behaviors

- **Benefit from incremental changes**

- **Be redundant**
  - Data replication
  - Checks for correctness
A case of contradictions: RAID

- **RAID**: Redundant Array of Inexpensive Disks
  - Within a single computer, replicate data onto multiple disks
  - e.g., with 5 1TB disks can get 4TB of useful storage and recover from any single disk failure
A case of contradictions: RAID

- **RAID**: Redundant Array of Inexpensive Disks
  - Within a single computer, replicate data onto multiple disks
  - e.g., with 5 1TB disks can get 4TB of useful storage and recover from any single disk failure

- Does Google use RAID?
  - On a piece of paper, argue from the distributed systems design principles (1) why Google should use RAID, and (2) why Google should not use RAID.