High-level Design

Overview

• What is software architecture?
• Classic architecture styles
• UML Package Diagram
• How to do architecture Design?
What is Software Architecture?

• “The architecture of a system is comprehensive framework that describes its form and structure -- its components and how they fit together”
  --Jerrold Grochow

What is Architectural Design?

• Design overall shape & structure of system
  – the components
  – their externally visible properties
  – their relationships
• Goal: choose architecture to reduce risks in SW construction & meet requirements
SW Architectural Styles

• Architecture composed of
  – Set of components
  – Set of connectors between them
    • Communication, co-ordination, co-operation
  – Constraints
    • How can components be integrated?
  – Semantic models
    • What are the overall properties based on understanding of individual component properties?

Architecture Patterns

• Common program structures
  – Pipe & Filter Architecture
  – Event-based Architecture
  – Layered Architecture
  – Map-Reduce Architecture
Pipe & Filter Architecture

- A pipeline contains a chain of data processing elements
  - The output of each element is the input of the next element
  - Usually some amount of buffering is provided between consecutive elements

Example: Optimizing Compiler

Compiler Structure

Compiler Optimization

[Engineering a Compiler, K. D. Cooper, L. Torczon]
Pros and Cons

- Other examples
  - UNIX pipes, signal processors
- Pros
  - Easy to add or remove filters
  - Filter pipelines perform multiple operations concurrently
- Cons
  - Hard to handle errors
  - May need encoding/decoding of input/output

Event-based Architecture

- Promotes the production, detection, consumption of, and reaction to events
- More like event-driven programming
Example: GUI

Pros and Cons

- Other examples:
  - Breakpoint debuggers, phone apps, robotics
- Pros
  - Anonymous handlers of events
  - Support reuse and evolution, new consumers easy to add
- Cons
  - Components have no control over order of execution
**Layered/Tiered Architecture**

- Multiple layers are defined to allocate responsibilities of a software product
- The communication between layers is hierarchical
- Examples: OS, network protocols

**Variant architectures**

- 2-layer architecture
  - Client-Server Architecture
  - Data-centric Architecture
- 3-layer architecture
  - Model-View-Controller
**Client-Server Architecture**

- Partition tasks or workloads between the providers and consumers of service or data
- Same system, different hardware, network communication
- Thin or thick clients

**Data-centric Architecture**

- A data store resides at the center to be accessed frequently by agents
- Blackboard sends notification to subscribers when data of interest changes
- Compared with event-driven architecture?
2-layer: Examples, Pros and Cons

• Examples
  – Web-based applications, Distributed file system, version control system

• Pros
  – Low requirements for agents
  – Easy to add/change agents

• Cons
  – Blackboard can be a bottleneck
  – Data integrity

3-layer Architecture

• Presentation: UI to interact with users
• Logic: coordinate applications and perform calculations
• Data: store and retrieve information as needed
Example: Online Ordering System


Model-View-Controller

Design of Finite State Machine Drawing Tool
Key Points about MVC

- View layer should not handle system events
- Controller layer has the application logic to handle events
- Model layer only respond to data operation

3 layer: Pros and Cons

- Pros
  - Clear separate concerns
    - Easy to develop, change & reuse
- Cons
  - Hard to maintain when changes in one layer can affect other layers
Layered Architecture: Pros and Cons

• Pros
  – Support increasing levels of abstraction during design
  – Support reuse and enhancement
• Cons
  – The performance may degrade
  – Hard to maintain

Hadoop Map-Reduce

• Open source project written in Java
• Large scale distributed data processing
• Based on Google’s Map Reduce framework and Google file system
• Work on commodity hardware
• Used by Google, Yahoo, Facebook, Amazon, and many startups

http://www.slideshare.net/acmvnit/hadoop-map-reduce
Hadoop Core

• Hadoop Distributed File System (HDFS)
  – Distributes and stores data across a cluster
• Hadoop Map Reduce (MR)
  – Provides a parallel programming model
  – Moves computation to where the data is
  – Handles scheduling, fault tolerance
  – Status reporting and monitoring

Word Count Problem

• Find the frequency of each word in a given corpus of documents
• Trivial for small data
• How to process more than a TB of data?
  – Doing it on one machine is very slow
• Good news: it can be parallelized across number of machines
• Strategy: Divide-and-conquer
**Map Reduce Architecture**

- Programmer submits job (mapper, reducer, input) to Job tracker
- Job tracker, splits input data, schedules and monitors various map and reduce tasks
- Task tracker executes map and reduce tasks

**Map Reduce Programming Model**

- Inspired by functional language primitives
- **map** f list : applies a given function f to each element of a list and returns a new list
  
  map square [1 2 3 4 5] = [1 4 9 16 25]
- **reduce** g list : combines elements of list using function g to generate a new value
  
  reduce sum [1 2 3 4 5] = [15]
- Map and reduce do not modify input
Mapper and Reducer

- **Mapper**
  - Input: records (database rows etc.) represented as key/value pairs
  - Output: one or more intermediate key/value pairs for each input
- **Reducer**
  - Input: intermediate key/value pairs
  - Output: final key/value pairs based on combination of input pairs

Word Count Map Reduce Job

- **Mapper**
  - Input: <key: offset, value: a line of a document>
  - Output: <key: word, value: count in the line>
- **Reducer**
  - Input: <key: w, value: count>
  - Output: <key: w, value: \( \Sigma \text{count} \)>
**Map, Shuffle & Sort, Reduce**

- **Partition Map-output by hashing the key**
  - Same keyed pairs are put together
- **Number of partitions is equal to number of reducers**
- **Partitions are sorted by keys**
Revisit Map Reduce Architecture

- **Job tracker**
  - Splits input and assigns to tasktrackers
  - Schedules and monitor map tasks (heartbeat)
  - On completion, schedule reduce tasks

- **Task tracker**
  - Execute map tasks
  - Partition and sort map outputs
  - Execute reduce tasks

Usage

- Map-Reduce greatly simplifies writing large scale distributed applications
- Used for building search index at Google, Amazon
- Widely used for analyzing user logs, data warehousing and analytics
- Also used for large scale machine learning and data mining applications
Pros

- **Locality**
  - Job tracker divides tasks based on location of data
- **Parallelism**
- **Fault tolerance**
  - Job tracker maintains a heartbeat with task trackers
  - Failures are handled by reexecution

Cons?
How to Do Architecture Design?

• When decomposing a system into subsystems, take into consideration
  – how subsystems share data
    • data-centric or data-distributed
  – how control flows between subsystems
    • as scheduled or event-driven
  – how they interact with each other
    • via data or via method calls