Agenda

Introductions  Game Rules  Game  Discussion
Agenda

Introductions

Game Rules

Game

Discussion
Instructions

This game intends to illustrate the essentials of architecture design using an iterative method such as ADD.

You will be competing against other software architects (or other teams) from rival companies, so you need to make smart design decisions or else your competitors will leave you behind!
Introduction

ADD Step 1: Review Inputs

Let’s start by reviewing the inputs to the design process...

Smart Decisions

Review inputs → Iteration 1 → Iteration 2 → Iteration 3 → Iteration 4 → Iteration 5 → Finish

Step 1: Review Inputs

- Design objectives
- Primary functional requirements
- Quality attribute scenarios
- Constraints
- Concerns

Step 2: Establish iteration goal and select inputs to be considered in the iteration

Step 3: Choose one or more elements of the system to decompose

Step 4: Choose one or more design concepts that satisfy the inputs considered in the iteration

Step 5: Instantiate architectural elements, allocate responsibilities and define interfaces

Step 6: Sketch views and record design decisions

Step 7: Perform analysis of current design and review iteration goal and design objectives

Software architecture design
Functional drivers

Web Servers
- Hundreds of servers
- Massive logs from multiple sources

Real-time monitoring
- Full-text search

Historical static reports
- Available through BI corporate tool

Raw and aggregated historical data
- Ad-hoc analysis
- Human-time queries

24/7 Operations, Support Engineers, Developers

Management

Data Scientists/Analysts
Quality attributes

- **Performance**
  - Q1: The system shall collect 10000 raw events/sec in average from up to 300 web servers
  - Iteration 2
  - Q2: The system shall provide static reports over historical data (<5 sec report load time) for Product and IT Managers
  - Iteration 4
  - Q3: The system shall provide ad-hoc analysis over historical data with human-time queries (<1 min query execution time) historical for Data Analysts
  - Iteration 5
  - Q4: The system shall provide full-text search and ad-hoc analysis with human-time queries (<20 seconds query execution time, last 48 hours data) for on-duty Operations, Developers and Support Engineers
  - Q5: The system shall automatically refresh real-time monitoring dashboard with new data (<1 min data latency, last 48 hours data) to on-duty Operations, Developers and Support Engineers
  - Iteration 2

- **Compatibility**
  - Q6: The system shall be composed of components that preferably integrate to each other with no or minimum custom coding
  - Iteration 2

- **Reliability**
  - Q7: The data collection and event delivery mechanism shall be reliable (no message loss)
  - Iteration 2

- **Extensibility**
  - Q8: The system shall support adding new data sources by just updating configuration/metadata with no interruption of ongoing data collection
  - Iteration 2-5

- **Scalability**
  - Q9: The system shall store raw data for the last 60 days (~1 TB of raw data per day, ~60 TB in total)
  - Iteration 3

- **Availability**
  - Q10: The system shall survive and continue operating if any of its node or component is failed
Constraints

C1: The system shall be composed primarily with open source technologies for cost saving. For those components where value/cost of using proprietary technology is much higher proprietary technology should be used.

C2: The system shall support two deployment environments – Private Cloud and Public Cloud. Architecture and technology decisions should be made to keep deployment vendor as agnostic as possible.

C3: The system shall use corporate BI tool with SQL interface for static reports (e.g. MicroStrategy, QlikView, Tableau)
Agenda

Introductions  Game Rules  **Game**  Discussion
Game Rules

ADD Step 2: Review iteration goal and select inputs
ADD Step 3: Choose one or more elements of the system to decompose

The game is played in rounds which represent the iterations.

The goal for the iteration is provided:
- Drivers to be considered
- Element to decompose
Instructions

Smart Decisions

Review inputs

Iteration 1

Iteration 5

Iteration 4

Iteration 3

Iteration 2

Finish

© 2015 H. Cervantes, S. Haziyev, O. Hrytsey, R. Kazman,
Iteration 1 goal: Logically structure the system

Drivers for the iteration:
- Ad-Hoc Analysis
- Real-time Analysis
- Unstructured data processing
- Scalability
- Cost Economy

Element to decompose:
Big Data System
ADD Step 4: Choose one or more design concepts that satisfy the inputs considered in the iteration

Make the design decision of selecting design concepts:
- Reference architectures
- Patterns (including technology families)
- Tactics
- Externally developed components
Game Rules: Design Concepts Cards

**Extended Relational**
*Reference Architecture for Data Analytics*

**Description:** Although this reference architecture is completely based on relational model principles and SQL-based DBMS, it intensively uses MPP and In-Memory techniques to improve scalability and extensibility.

**Functionality:**
- Ad-hoc analysis — supports complex ad-hoc real-time read queries
- Real-time analysis — near-real-time with micro-batching technique
- Unstructured data processing — supports ingesting and querying semi-structured data such as JSON/XML

**Quality attributes and constraints:**
- Scalability — can run terabytes with MPP and clustering capabilities
- Extensibility — extending data model is possible but not as flexible as in non-relational architectures
- Data quality — relational model is integrated and consistent
- Cost economy — MPP DBMS license cost is quite expensive

**Sample implementations:** Business Reporting, Enterprise Data Warehousing, Data Discovery

**Patterns**
- Reference Architectures
- Families

**Technologies**
- Reference Architectures
- Families
Time to make your first smart decision!

Drivers for the iteration:
- Ad-Hoc Analysis
- Real-time Analysis
- Unstructured data processing
- Scalability
- Cost Economy

Element to decompose:

To Do:
Select 1 Reference Architecture Card

Possible alternatives:
- Extended Relational
- Pure Non-Relational
- Data Refinery
- Lambda Architecture

Disqualified alternatives:
- Traditional Relational
Fill the scorecard

<table>
<thead>
<tr>
<th></th>
<th>Iteration #1</th>
<th>Iteration #2</th>
<th>Iteration #3</th>
<th>Iteration #4</th>
<th>Iteration #5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(a) Design Decisions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>(Names of selected design concept(s))</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(b) Driver selection points</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>(from cards)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(c) Instantiation points</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>(from dice)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(d) Analysis bonus points</strong></td>
<td></td>
<td></td>
<td></td>
<td>Final score:</td>
<td></td>
</tr>
<tr>
<td><em>(from review)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(e) Iteration total</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>(b + c + d)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fill (b) by adding the points for the drivers considered for the iteration, in this case:
- Ad-Hoc Analysis
- Real-time Analysis
- Unstructured data processing
- Scalability
- Cost Economy

= 1 Point
Introduction

ADD Step 5: Instantiate elements, allocate responsibilities and define interfaces.
ADD Step 6: Sketch views and record design decisions

Record the design decision and throw two dice to simulate how well you instantiate your design concept
Fill the scorecard

<table>
<thead>
<tr>
<th></th>
<th>Iteration #1</th>
<th>Iteration #2</th>
<th>Iteration #3</th>
<th>Iteration #4</th>
<th>Iteration #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Design Decisions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Names of selected design concept(s))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b) Driver selection points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(from cards)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c) Instantiation points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(from dice)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d) Analysis bonus points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Final score:</td>
</tr>
<tr>
<td>(from review)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e) Iteration total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b + c + r)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Roll the dice and add or subtract points according to the following table, fill (c).

<table>
<thead>
<tr>
<th>Dice result (1)</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 - 3</td>
<td>-2</td>
</tr>
<tr>
<td>4 - 9</td>
<td>0</td>
</tr>
<tr>
<td>10 - 12</td>
<td>+2</td>
</tr>
</tbody>
</table>

Record design decisions in (a)
Review design decisions and score iteration.

We will review the first iteration together, but the rest will be reviewed at the end.
# Iteration 1: Scoring

Score Ad-Hoc Analysis, Real-time Analysis, Unstructured data processing, Scalability, Cost Economy

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Relational</td>
<td>3+2+2+2+1=<strong>10</strong></td>
<td>-4</td>
<td>This reference architecture is less appropriate for this solution mostly because of cost and real-time analysis limitation</td>
</tr>
<tr>
<td>Pure Non-Relational</td>
<td>2+2.5+3+3+3=<strong>13.5</strong></td>
<td></td>
<td>This reference architecture is closer to the goal than the others except Lambda Architecture</td>
</tr>
<tr>
<td>Lambda Architecture (Hybrid)</td>
<td>2.5+3+3+3+3=<strong>14.5</strong></td>
<td>+2</td>
<td>This is the most appropriate reference architecture for this solution! From the provided reference architectures Lambda Architecture promises the largest number of benefits, such as access to real-time and historical data at the same time.</td>
</tr>
<tr>
<td>Data Refinery (Hybrid)</td>
<td>3+1+3+2+1=<strong>10</strong></td>
<td>-4</td>
<td>This reference architecture is less appropriate for this solution mostly because of cost and real-time analysis limitation</td>
</tr>
</tbody>
</table>
Fill the scorecard

<table>
<thead>
<tr>
<th></th>
<th>Iteration #1</th>
<th>Iteration #2</th>
<th>Iteration #3</th>
<th>Iteration #4</th>
<th>Iteration #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Design Decisions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b) Driver selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c) Instantiation points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d) Analysis bonus</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Final score:</td>
</tr>
<tr>
<td>(e) Iteration total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add bonus points, if any and fill (d)

Sum the points and calculate the total for the iteration in (e)
Lambda Architecture Logical Structure

- **Batch Layer**
  - Master Dataset
  - Pre-Computing
  - Batch Views

- **Serving Layer**
  - Query & Reporting

- **Data Stream**
  - Source: http://lambda-architecture.net/
Big Data Analytics Reference Architectures Trade-off

Legend:
- Unstructured data processing capabilities (the larger the better)
- Real-time analysis capabilities (more saturated the better)
Instructions

Smart Decisions

Review inputs

Iteration 1

Iteration 5

Iteration 4

Finish

Iteration 2

Iteration 3

© 2015 H. Cervantes, S. Haziyev, O. Hrytsey, R. Kazman,
Iteration 2: Design Data Stream Element

Drivers for the iteration:
- Performance (for Family and Technology)
- Compatibility (for Family)
- Reliability (for Technology)

Element to decompose:

Batch Layer
- Master Dataset
- Pre-Computing
- Batch Views

Serving Layer
- Query & Reporting

Speed Layer
- Real-time Views

Select 1 Family card and 1 Technology card

Possible alternatives:
- Data Collector
  - Apache Flume
  - Logstash
  - Fluentd
- Distributed Message Broker
  - Apache Kafka
  - RabbitMQ
  - Amazon SQS
  - Apache ActiveMQ

Disqualified alternatives:
- ETL Engine (lack of real-time data stream support and no need for complex data transformations)

Tip:
- Look for an option that can be deployed on-Premise and on-Cloud
Iteration 3: Design Batch Layer

Drivers for the iteration:
• Scalability
• Availability

Element to decompose:

Batch Layer
- Master Dataset
- Pre-Computing
- Batch Views

Serving Layer
- Batch Views
- Real-time Views
- Query & Reporting

Speed Layer
- Data Stream
- Real-time Views

Data Stream

Possible alternatives:
- Distributed File System
- NoSQL Database/Key-Value
- NoSQL Database/Graph-Oriented
- Analytic RDBMS
- Distributed Search Engine

Disqualified alternatives:
- Document-Oriented
- Column-Family

Tip:
- Look for an option with better extensibility (easy storing of new data formats)

To Do:
Select 1 Family card
Iteration 4: Design Serving Layer

Drivers for the iteration:
• Ad-hoc Analysis (for Family)
• Performance (for Family and Technology)

Element to decompose:

Batch Layer
- Master Dataset
- Pre-Computing
- Batch Views

Serving Layer
- Query & Reporting

Speed Layer
- Real-time Views

Data Stream

To Do:
Select 1 Family and 1 Technology card

Possible alternatives:
- NoSQL Database
  - Document-Oriented
    - MongoDB
    - CouchDB
  - Column-Family
    - HBase
    - Cassandra
- Interactive Query Engine
  - Impala
  - Apache Hive (Stinger)
  - Spark SQL

Disqualified alternatives:
- NoSQL Database/Key-Value
- NoSQL Database/Graph-Oriented
- Analytic RDBMS
- Distributed Search Engine

Tip:
• Look for an option that provides ad-hoc analysis and still good performance for static reports
Iteration 5: Design Speed Layer

Drivers for the iteration:
• Ad-hoc Analysis (for the family)
• Real-time Analysis (for the technology)

Possible alternatives:
NoSQL Database
- Document-Oriented
  - MongoDB
- CouchDB
- Column-Family
  - HBase
  - Cassandra

Search & Query
- Distributed Search Engine
  - Splunk
- Elasticsearch
- Apache Solr

Disqualified alternatives:
- NoSQL Database/Key-Value
- NoSQL Database/Graph-Oriented
- Analytic RDBMS

Possible alternatives:
- Look for an option that provides full-text search capabilities and extensibility (new data formats and dashboard views)

Select 1 Family and 1 Technology card
**Iteration 2: Design decisions analysis and scoring**

**Family card: score Performance and Compatibility**

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Collector</td>
<td>2+3=5</td>
<td>+2</td>
<td>Additional bonus is added for extensibility</td>
</tr>
<tr>
<td>Distributed Message Broker</td>
<td>3+1=4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Technology card: score Performance and Reliability**

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Flume</td>
<td>2+2=4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logstash</td>
<td>2+2=4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fluentd</td>
<td>2+3=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RabbitMQ</td>
<td>2+2=4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache Kafka</td>
<td>3+2=5</td>
<td>+2</td>
<td>Additional bonus for easier deployment and configuration comparing with other alternatives</td>
</tr>
<tr>
<td>Amazon SQS</td>
<td>0</td>
<td></td>
<td>Disqualified due to deployment constraint (support On-premise and Cloud)</td>
</tr>
<tr>
<td>Apache ActiveMQ</td>
<td>2+2=4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Iteration 3: Design decisions analysis and scoring

### Family card: score Scalability and Availability

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoSQL Database/Column-Family</td>
<td>3+3=6</td>
<td>-1</td>
<td>Column families must be defined up front and require modification when log format is changed – extensibility disadvantage</td>
</tr>
<tr>
<td>NoSQL Database/Document-Oriented</td>
<td>3+3=6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributed File System</td>
<td>3+3=6</td>
<td>+2</td>
<td>Bonus for extensibility (log format changes do not require any changes in DFS cluster) and easier deployability/maintainability compared with NoSQL databases</td>
</tr>
</tbody>
</table>

Note: If you selected FluentD during the previous iteration and DFS at this iteration you receive **-1 performance bonus** (FluentD uses WebHDFS which pays a little performance cost due to HTTP)
## Iteration 4: Design decisions analysis and scoring

### Family card: score Ad-hoc Analysis, Performance

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive Query Engine</td>
<td>3+2=5</td>
<td>+2</td>
<td>Extensibility bonus because this approach does not require complex tuning of schema for introducing new reports and data types</td>
</tr>
<tr>
<td>NoSQL Database/Column-Family (+ SQL connector)</td>
<td>1+3=4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NoSQL Database/Document-Oriented (+ SQL connector)</td>
<td>1.5+3=4.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Technology card: score Performance

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impala</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache Hive</td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spark SQL</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache Cassandra</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache HBase</td>
<td>2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache CouchDB</td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Iteration 5: Scoring

### Family card: score Ad-hoc Analysis

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Search Engine</td>
<td>2</td>
<td>+2</td>
<td>Full-text search is out of the box + bonus for extensibility (adding new log formats and report views requires minimum changes in search engine)</td>
</tr>
<tr>
<td>NoSQL Database/Column-Family</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NoSQL Database/Document-Oriented</td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Technology card: score Real-time Analysis

<table>
<thead>
<tr>
<th>Design decision</th>
<th>Driver points</th>
<th>Bonus points</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elasticsearch</td>
<td>2.5</td>
<td>+2</td>
<td>Elasticsearch easily integrates with Kibana – an open source interactive dashboard</td>
</tr>
<tr>
<td>Apache Solr</td>
<td>2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Splunk (Indexer)</td>
<td>2.5</td>
<td>-2, +2</td>
<td>-2 penalty for cost and +2 bonus (Splunk offers end-to-end solution including powerful visualization tool)</td>
</tr>
<tr>
<td>Apache Cassandra</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache HBase</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apache CouchDB</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fill the scorecard

<table>
<thead>
<tr>
<th>(a) Design Decisions</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Final score:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Names of selected design concept(s))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b) Driver selection points (from cards)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c) Instantiation points (from dice)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d) Analysis bonus points (from review)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e) Iteration total ((b + c + d))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Calculate the final score
- Add 2 to the player who finished first
- Add 1 to the player who finished second
Agenda

- Introductions
- Game Rules
- Game
- Discussion
Thank You