

Testing Distributed System

Azize Ezgi Ersoy | 11011013
Buğra Cansın Göz | 12011014
Hande Gül | 11011036
Sezgin Ege | 12011007

What We Will Discuss?

- Distributed Systems
- Testing Distributed Systems
 - Test Scenarios
 - Test Methodologies
 - Unit Testing – Component Testing
 - Integration Testing – System Testing
 - Main Problems of Testing Distributed Systems
- Conclusion

Definition

- Collection of computers which are remote from a central computing
- Collection or network of loosely coupled, autonomous, computers
- Computing facility built with many computers that operate concurrently

According to Tanenbaum and Steen

Collection of independent computers that appears to its users as a single coherent system

- hardware aspect
- software aspect

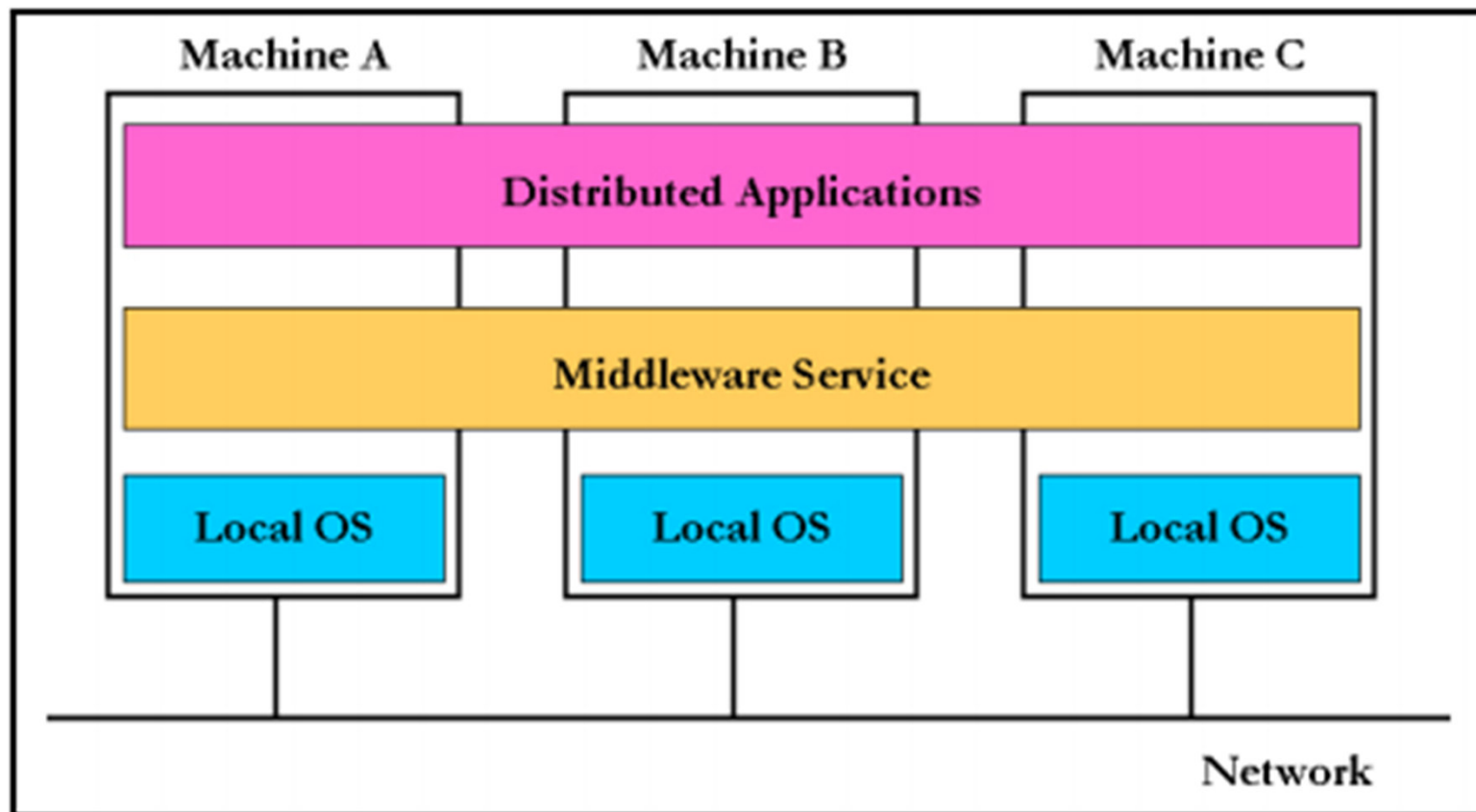




Figure 1: A Distributed System Organized as Middleware

(Source: Tanenbaum & Steen, 2002)

Characteristics

- Resource Sharing
 - Openness
 - Concurrency
 - Transparency
- 

Types of Transparency

- Access
 - Location
 - Concurrency
 - Replication
 - Failure
 - Migration
 - Performance
 - Scaling
 - Relocation
 - Persistence
- 

-
- Scalability
 - Fault tolerance
 - Heterogeneity
 - Security
- 

Testing Distributed Systems

A brief introduction about testing and its essential role in constructing distributed systems are discussed in conjunction with different techniques that can be applied to each level of testing.

These include

- Unit testing
- Integration testing
- System testing
- Acceptance testing
- Regression testing.

In addition, a list of important features that needs to be taken into consideration when designing distributed system is categorized and different solutions are discussed.

Today, testing is a crucial activity in software development process and should be designed before the actual code is written.

Additionally, the role of testing is depicted throughout a number of System Development Life Cycle Models (SDLC) that can be applied during the software development process.

The most common SDLC models are,

- Waterfall model
- V software life-cycle model
- Spiral model.

Three main types of tests:

- Black-box Testing
- White-box Testing
- Gray-box Testing

1. Black-box testing, is the technique in which the testers do not have access to the source code.

- Tester and programmer are independent of one another, this helps to prevent the programmer's bias towards his/her own work.
- Tests are often done from the user's point of view and can be designed when the specifications are completed.

2.White-box testing,is the technique which requires testers to have access to the logic and structure of the source code when performing testing tasks.

- White-box testing is used when designing unit test, integration test, and regression test.
- A white-box tester can design test cases that exercise independent paths.


3.Gray-box testing,is used to test a system by directly targeting the various components and levels of the system.therefore gray-box testing requires the knowledge of the system's internal structure as well as the requirements from the user's point of view.

Test Scenarios


		APPLICATION	
		Single Node	Multiple Nodes
TEST	Single Node	Traditional Testing	Client/Server
	Multiple Nodes	Write once, test everywhere- client side apps	Full distributed application and tests

Test Methodologies

Project managers should choose suitable levels of testing depending upon the status length

- Unit Testing
 - Integration Testing
 - Regression Testing
 - System Testing
 - Acceptance Testing
- 

Unit Testing

- Lowest testing level
 - Written before the actual codes derived
 - Ensure that each function in the program works correctly before integrating them
 - Performed during the project's development cycle | project begins
- 

Integration Testing

- Performed by combining two or more units into one component


Regression Testing

- Is done whenever the program is modified
- Rerunning the existing test against change along with new tests

System Testing

- 'conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements'
- Measure the final system to its original objectives
- Two implications
 - Program do not meet the original objectives
 - Should be a set of measurable objectives

Acceptance Testing

- Generate tests from the end-user's perspective
 - System performance correctly? and Meet all the required feature?
 - Written before the features are fully implemented
 - 'Contract' between the developer and the customer
- 

Main Problems Of Testing

- Resource Sharing
- Node failure
- Asynchronous Data Delivery