NYIT

**Fall 2021**

Homework No: 12

Title: Distributed Application Architecture and Java Web Apps and Server.

**Name:** Patade, Yash Arun

**Class ID#:**  11

**School ID#:** 1284979

**Course:** Java Networking

**Course ID:** CSCI 725

**Date:** 12/10/2021

Assignment Contents

[Question: Fundamental SE Principle. 3](#_Toc90768328)

[1) What is the most fundamental SE principle (SE slogan)? 3](#_Toc90768329)

[2) Distributed Software Engineering. 3](#_Toc90768330)

[3) RPC Stubs. 3](#_Toc90768331)

[4) Marshalling/Serializing Argument Values. 4](#_Toc90768332)

[5) What are the basic distributed application architectures that benefit from reuse? 4](#_Toc90768333)

[6) Thread as CPU Representative. 5](#_Toc90768334)

[7) What does one thread of instructions need from the CPU to run? 5](#_Toc90768335)

[8) How does thread use ALU (Computation-hardware module) in the CPU? 6](#_Toc90768336)

[9) Multithreading as an Imperative. 6](#_Toc90768337)

[10) Multithread advantages. 7](#_Toc90768338)

[11) Multithreading disadvantages. 7](#_Toc90768339)

[12) What is the difference between concurrent and parallel threads? 7](#_Toc90768340)

[13) Which part of the kernel’s executive is in charge of the process dispatching on the CPU, (Moving process from the Ready to Running state)? 8](#_Toc90768341)

[14) How can two threads pass to each other messages? 8](#_Toc90768342)

[References. 9](#_Toc90768343)

# Question: Fundamental SE Principle.

# 1) What is the most fundamental SE principle (SE slogan)?

Answer) Reuse, reuse, reuse

# 2) Distributed Software Engineering.

• Distributed software/applications are application programs made of communicating partsthat we may call modules.

• Distributed SE illustrates the best how to simplify complex engineering problems by reuse-principle application.

# 3) RPC Stubs.

• A client-side stub is a function that looks to the client as if it were a callable server function

– I.e., same API as the server’s implementation of the function

• A server-side stub/skeleton looks like a caller to the server

– I.e., like a hunk of code invoking the server function

• The client program thinks it’s invoking the server

– but it’s calling into the client-side stub

• The server program thinks it’s called by the client

– but it’s really called by the server-side stub

• The stubs send messages to each other to make the RPC happen transparently (almost!)

# 4) Marshalling/Serializing Argument Values.

• Marshalling or serializing is the packing of function parameter values into a message packet series of bits.

– the RPC stubs call type-specific functions to marshal bit stream or unmarshal (receive the bit streamand recreate parametr values) the parameters of an RPC

• Client stub marshals the arguments into a message bit stream

• Server stub unmarshals the arguments and uses them to invoke the service function, (Pass them to the called function)

– on return:

• the server stub marshals return values into the bit stream

• the client stub unmarshals return bit stream, and returns to the client program the result as the called function was right in the same memory space.

# 5) What are the basic distributed application architectures that benefit from reuse?

• 2-layer API reusing only, 3-layer RPC and API reusing,

• 4-layer message protocol, RPC with API reusing, and

• 5-layer Web-Client/Web-Server framework based architecture.

# 6) Thread as CPU Representative.

• A thread is an execution path through the sequence of JVM object code (Java byte-code) or identifiable execution context in which variables may change values.

• Each thread has its own Program Counter (PCr) and RTS (with RTS tracking registers).

• Having own registers (register values) a thread appears like a virtual CPU on which registers change values.

• With a single processor, only one thread runs at a time.

Diagram

Description automatically generated with low confidence

# 7) What does one thread of instructions need from the CPU to run?

Answer)

• CPU registers only

– To track its loaded memory image components (CS, DS, PCB)

# 8) How does thread use ALU (Computation-hardware module) in the CPU?

Answer)

• By using instruction register (Ir).

– Thread of a running program simply only-needs, claims and uses CPU register set.

• Very rarely all of them, but always:

– Ir, PCr, SPr, SBPr, DSr, CSr, PCBPr, and a few of many data registers Dr’s, (Pr’s are pointer registers)

# 9) Multithreading as an Imperative.

• Modern user-interaction intensive client side and serviceintensive back-end server side programs are heavily multi-threaded.

Diagram

Description automatically generated

# 10) Multithread advantages.

• Easier to program.

– 1 thread per task.

• Can provide better performance.

– thread only runs when needed.

– no polling to decide what to do.

• Multiple threads can share resources.

• Utilize multiple processors if available.

# 11) Multithreading disadvantages.

• Multiple threads can lead to deadlock “jam”.

– Much more on this later.

• Overhead of switching between threads.

– Still smaller/faster than Interrupt-based multiple-process switching.

# 12) What is the difference between concurrent and parallel threads?

• Concurrent threads are seemingly-parallel (virtually parallel).

– Time sharing one CPU.

• Take turns on the same CPU.

• Parallel threads run in true-parallelism.

– Using multiple CPUs.

• Each thread uses own CPU.

– Allocated by multithreaded OS kernel.

# 13) Which part of the kernel’s executive is in charge of the process dispatching on the CPU, (Moving process from the Ready to Running state)?

Diagram

Description automatically generated

Answer)

• CPU time manager, also known as CPU scheduler, (part of the process manager or executive supervisor).

# 14) How can two threads pass to each other messages?

Answer)

• If threads belong to two distinct processes than via kernel IPC services and kernel’s buffer variables.

– Could slow application with massively exchanging messages between module-processes.

• If threads belong to the same process than via shared “mailbox” variables.

– The fastest possible.

• Instead of being distributed (many processes) stand-alone application should be single process multi-threaded (multiple threads)

# References.

[1] [Microsoft PowerPoint - \_725\_10\_Lecture\_02\_DistSE\_B\_.pptx (tfbor.com)](http://tfbor.com/02_725/12_DistAppArchitecture/725_12_Lecture_02_DistSE_B_.pdf)

[2] [Microsoft PowerPoint - \_\_725\_12\_Lecture\_02\_JavaThreadsRev\_v2.pptx (tfbor.com)](http://tfbor.com/02_725/12_WebServer/725_12_Lecture_01_JavaThreadsRev_v2.pdf)

[3] [Microsoft PowerPoint - \_\_725\_12\_Lecture\_03\_ServerThreading\_v1.pptx (tfbor.com)](http://tfbor.com/02_725/12_WebServer/725_12_Lecture_02_ServerThreading_v1.pdf)