CS 551 – Final Exam
Ishaan Roychowdhury

By submitting the following answers to the exam, I attest that the answers are mine alone, that I did not communicate directly or indirectly with any others about the exam, and that the following lists all the external materials I used beyond the textbook and class notes.

External links I used:

1) https://www.projectpro.io/article/aws-vs-gcp-which-one-to-choose/477
2) https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
3) https://phoenixnap.com/kb/hadoop-mapreduce
4) https://cronitor.io/guides/kubernetes-cron-jobs
5) https://www.baeldung.com/ops/kubernetes-run-cron-jobs
6) https://phoenixnap.com/blog/how-to-choose-cloud-provider
7) https://www.ziffity.com/blog/how-to-choose-the-right-cloud-service-provider/
8) https://about.gitlab.com/topics/ci-cd/
9) https://www.redhat.com/architect/cloud-industrial-edge-architecture

* Used ChatGPT to an extent for framing sentences, diagrams, and cross-checking sources.

Problem 1 (20 pts)

You have been hired by an enterprise that is moving all its computing to the cloud. They signed a contract with a cloud provider the day before you started, and the CEO announced a company-wide mandate: no matter how you do it, get to the cloud in less than two months. They make it clear that "getting to the cloud early" is essential (presumably to impress investors and keep stock prices high), and they also say that they are willing to wait until later for high performance.

Your team has started to make a list of applications they will disaggregate over the next two months because they will need to turn each application into a set of microservices. They are demanding that the company hire additional software engineers for two months. Is that approach the best way to achieve the company mandate? If so, tell why and explain what the engineers will need to accomplish. If not, tell what approach you would recommend instead

While a 2-month timescale for cloud migration is hypothetically doable, especially with the flood of more skilled expertise and resources, a more methodical and efficient strategy is likely to offer stronger long-term advantages. Quickly disaggregating systems into microservices and onboarding new workers incurs considerable expenditures, including recruiting and training.

Such a rapid change raises the danger of neglecting crucial security, compliance, and architectural factors, which can have long-term consequences. Hence, while aggressive recruiting and investment may make a quick changeover possible, it's more practical to take a more measured, stepwise approach. This helps with broad examination and adjustments of cloud infrastructure and security measures, as well as ensuring that the transition matches corporate goals without sacrificing long-term performance, security, or financial prudence.

Here are the main steps that one should take:

1) **Prioritization and Evaluation**: This first step should entail doing a thorough inspection of the existing application to determine which services (if disaggregated), will provide the most business value when transferred to the cloud. The cloud readiness of these services, their importance in routine operations, and the complexity of migration should be considered. This delivers a strategic approach in which the most advantageous migrations take place first.

2) **Gradual Migration and Training:** A staggered migration reduces disturbance and facilitates the gradual upskilling of current employees on cloud technology. Starting with simple and isolated apps, not only teaches key lessons that can be used to succeed in future migrations, but also helps the organization learn in a stepwise manner.

3) **Security and Compliance:** Strict security and compliance tests must supplement each phase of the migration. Given the cloud's changing security landscape, it's critical to build new security standards and guarantee that data is encrypted and that we have robust security protocols.

4) **Cost Management and Infrastructure Evaluation:** A comprehensive analysis of the total cost of ownership in the cloud should be performed. This covers both direct expenses such as storage and computation resources and indirect costs such as network bandwidth. The decision between a single cloud and a multi-cloud approach will be influenced by considerations such as cost optimization, risk mitigation, and the requirement for specialized services from several providers. As we have seen from our project as well, different cloud providers excel in different areas. (For example, AWS offers a broad set of scalable and secure services, whereas GCP draws on Google's experience in data analytics and machine learning.)[1]

5) **Feedback with 3rd party cloud contractors:** Gather feedback and execute optimizations after gaining feedback from 3rd party cloud contractors, rather than hiring developers for 2 months. This approach also contributes to the refinement of the migration strategy, resulting in smoother transitions for more complex and crucial sections later on.

To summarize, while the desire to transfer to the cloud rapidly is reasonable, especially in the context of impressing investors and market dynamics, a more careful and organized strategy is not only easier to execute but also more sustainable in the long run.

[1] https://www.projectpro.io/article/aws-vs-gcp-which-one-to-choose/477

Problem 2 (20 pts total)

You are working for a company that sells cloud transition services — helping enterprises move to the cloud. A customer in the aerospace industry wants to move from a supercomputer to the cloud. Your manager met with them, took notes on their application, and told your team it's all gibberish. His notes are phrases:

> *the digital wind tunnel, analysis involving computational fluid dynamics, massive amounts of data for high resolution (many points on an airfoil), half a petabyte at least, even more data for the new airplane design, uses matrices of double-precision floating-point values to give more accuracy, software inverts, and multiplies matrices, displays results in 3-D graphical form, uses colors to represent pressure at each point on the airfoil, extrapolates results to higher speeds of airflow, compare eigenvalues derived from the matrices, state-of-the-art computations are much better than competitors.*

Your manager says that the problem has enough data to justify using the cloud vendor's MapReduce software and wants the team to help move the application to MapReduce in the cloud.

A. (8 pts.)
Explain the basic steps MapReduce uses, name a technology that can handle most of MapReduce, and tell what pieces of software will need to be written.

MapReduce is a programming model for processing big data sets on a cluster using a parallel, distributed method, and it is especially well-suited for situations needing complicated processing on extremely large amounts of data. MapReduce's fundamental steps are as follows:

The MapReduce methodology starts with the **Split phase**, in which enormous data sets are separated and given to Map processors. Each CPU handles its allocated data during the **Map phase**, conducting computations to build key-value pairs. Following that, the **Shuffle phase** redistributes these data items, ensuring that all pairs with the same key are routed to the same Reduce processor. These pairings are processed in the **Reduce stage** to aggregate, reduce, or combine data, resulting in a smaller set of results. Finally, the **Merge results** stage integrates all of the results from the Reduce phase into the final output, bringing the process to a close. This concept is effective for large-scale data processing, providing an organized technique for dealing with massive volumes of data using distributed computing.

A technology that can handle most of MapReduce is **Apache Hadoop.**

Apache Hadoop is an open-source platform that is primarily designed to effectively store and handle big data volumes across computer clusters. Hadoop is made up of two primary components:

Processing Component: It leverages the MapReduce programming paradigm for distributed processing of massive data volumes. This technique is exceptionally scalable and fault-tolerant, allowing for quick data processing and analysis.

Hadoop Distributed File System (HDFS): HDFS is designed to reliably store large data sets across cluster nodes. It breaks files into large chunks and distributes them among cluster nodes to provide great data speed and resilience.

Hadoop's architecture enables it to manage terabytes of data, making it a great choice for operations such as the aerospace company's computational fluid dynamics study, which needs to process enormous quantities of data and conduct complex computations.

In automating MapReduce with Hadoop, the programmer's responsibilities include:

- **Map Function Software:** This component converts the basic data into key-value pairs using Map Function Software. It's critical for dealing with application-specific data kinds and structures, such as the aerospace company's fluid dynamics data.

- **Reduce Function Software:** The Reduce Function Software collects the key-value pairs produced by the Map function. It runs the necessary computations and outputs the results.

- **Configuration File:** This is an important component that describes how the MapReduce process should be conducted, including task and resource allocation.

These are the three most significant things for which a programmer would have to build software.

B. (12 pts.)
If you agree that the problem is appropriate for MapReduce, explain why it is appropriate; if you disagree, explain why it is inappropriate.

MapReduce can efficiently handle and process the massive data quantities necessary for aerodynamic simulations in the aerospace industry. It is well-suited for distributed storage and other data-intensive applications which require half a petabyte of data.

However, when it comes to the sophisticated and recurrent computations of CFD, such as manipulating matrices for accuracy and executing operations like inversion and multiplication, MapReduce may fall short. These operations need real-time, high-speed computing and low-latency inter-node communication, which are characteristics of HPC settings rather than batch-oriented MapReduce.

Furthermore, displaying CFD findings in 3D and interpreting them interactively necessitates graphics processing skills that are often above MapReduce's data processing capacity. While MapReduce is useful for some portions of the process, a combined approach with HPC may be required to address all of the computational needs of cutting-edge aircraft designs.

Problem 3 (20 pts total)

You have been hired to disaggregate a large monolithic application into microservices. Your team has identified nine modules and placed each in a separate microservice. The solution seems to work, and your team has been asked to monitor the resulting system. Here are the basics:

- *Module A contains the main program, and all users execute A.*
- *60% of the users also invoke module B, 20% invoke module C, and the remaining 20% invoke module D.*
- *Of the users invoking module B, 40% also invoke module E and 60% also invoke module F.*
- *Of the users invoking module C, half also invoke module G and half invoke module H, Among users invoking module H, 10% invoke module E and 90% also invoke module I.*

A. (10 pts)
Suppose N users are currently using the service. Write nine equations that tell the maximum number of containers of each of the nine types that are expected to be running when N users are accessing the service.

Module A: Every user executes Module A. So, the number of containers for Module A is equal to the number of users.
A(N) = N

Module B: 60% of the users invoke Module B.
B(N) = 0.60N

Module C: 20% of the users invoke Module C.
C(N) = 0.20N

Module D: The remaining 20% of the users invoke Module D.
D(N) = 0.20N

*** Module E**: 40% of the users invoking B also invoke Module E (Check below)
E(N) = 0.4 * 0.6N = 0.24N

Module F: 60% of the users invoking B also invoke Module F
F(N) = 0.6 * 0.6N = 0.36N

Module G: Half the users invoking Module C also invoke Module G
G(N) = 0.5 * 0.2N = 0.1N

Module H: Half the users invoking Module C also invoke Module H
H(N) = 0.5 * 0.2N = 0.1N

*** Module E Final Updated:** Among users invoking module H, 10% also invoke module E
E(N) = 0.24N (previously calculated above) + 0.1*0.1 N
E(N) = 0.24N (previously calculated above) + 0.01 N
E(N) = 0.25N

Module I: Among users invoking module H, 90% invoke module I
I(N) = 0.9 * 0.1N = 0.09N

Hence final equations are:

1) **A(N) = N**

2) **B(N) = 0.60N**

3) **C(N) = 0.20N**

4) **D(N) = 0.20N**

5) **E(N) = 0.25N**

6) **F(N) = 0.36N**

7) **G(N) = 0.1N**

8) **H(N) = 0.1N**

9) **I(N) = 0.09N**

Here, I assume that the application will scale out to 1 container per user. (If I have 1200 users, and my application takes 1 minute to load, I will not make 1200 users wait, I will automatically scale out to 1 container per user)

B. (10 pts)
According to your equations, what is the maximum number of containers of each type that will be running if N = 1200?

From my equations above, I substitute N as 1200.

1) *A(N) = N = 1200*
2) *B(N) = 0.60N = 0.6 * 1200 = 720*
3) *C(N) = 0.20N = 0.2 * 1200 = 240*
4) *D(N) = 0.20N = 0.2 * 1200 = 240*
5) *E(N) = 0.25N = 0.25 * 1200 = 300*
6) *F(N) = 0.36N = 0.36 * 1200 = 432*
7) *G(N) = 0.1N = 0.1 * 1200 = 120*
8) *H(N) = 0.1N = 0.1 * 1200 = 120*
9) *I(N) = 0.09N = 0.09 * 1200 = 108*

Hence my final answer for a maximum number of containers for each type that will be running if N =1200:

- **Module A: 1200**
- **Module B: 720**
- **Module C: 240**
- **Module D: 240**
- **Module E: 300**
- **Module F: 432**
- **Module G: 120**
- **Module H: 120**
- **Module I: 108**

  Here, I assume that the application will scale out to 1 container per user. (If I have 1200 users, and my application takes 1 minute to load, I will not make 1200 users wait, I will automatically scale out to 1 container per user)

  Problem 4 (20 pts)

  You are working for a company that uses edge computing to operate and monitor a factory. The edge data center adjacent to the factory is quite powerful — it has twelve racks of servers, has wired connections to the machinery in the factory, and uses containerized microservices. Most of the applications use Kubernetes to scale a microservice to activate additional assembly lines when a large number of orders arrive. You have been asked to add a new monitoring application. The specification calls for the monitor to check a piece of equipment every 2 minutes. An engineer suggests running a script on a server and having the script repeatedly sleep for 2 minutes and then check the machine. Your manager worries about what might happen if the server running the script fails, and the engineer suggests building yet

another script to monitor the monitor script. Unfortunately, your manager's manager rejects the approach, saying that too many applications already have extra applications that merely monitor monitors. Something is needed that doesn't require an extra "monitor monitor." At the team meeting, someone laments, "If we only had a standard way to run a container every 2 minutes without building yet another specialized application script." What approach do you suggest that uses existing software?

Given the context and needs, leveraging Kubernetes' **CronJob** resource might be a potential solution. CronJobs is a native component of Kubernetes, which is already in use in your environment for scaling microservices. This allows users to plan a container to run at certain periods or intervals, in this example, every 2 minutes, without the need for extra monitoring scripts or any other extra-special applications.

It runs exactly like Unix cron and has the same specification format. A CronJob in Kubernetes functions similarly to a line in a crontab or cron table file. It arranges for Kubernetes jobs to execute at specific times. When it's time to execute, Kubernetes constructs a Job object, which schedules the formation of a Pod to carry out the job described in the CronJob.[2]

It is important to use the right syntax and understand the various elements in the YAML configuration when establishing a Kubernetes CronJob. These include the schedule, concurrency policy, suspend option, history limitations for successful and unsuccessful jobs, and the job execution start time in seconds. This adaptability enables exact control over when and how activities are completed. Managing these CronJobs is simple, thanks to kubectl commands that allow you to create, list, describe, and remove CronJobs as needed.[3]

In conclusion, this is the approach that I would suggest that uses existing software to achieve running a container every 2 minutes without building yet another specialized application script.

[2] https://cronitor.io/guides/kubernetes-cron-jobs
[3] https://www.baeldung.com/ops/kubernetes-run-cron-jobs

Problem 5 (20 pts total)

You are working for a large enterprise company that uses cloud services. The cloud provider has pitched serverless computing services, and your company has decided to use it for customer-facing applications. The provider offers three tiers of service that depend on use. Of course, you expect the number of instances to vary over time, but the provider says that they will compute an average number of instances in daily use, ADU, and then they will base the monthly charge on the ADU for that month. The provider rounds up the computed ADU to the nearest integer, so don't worry about fractions. There are three tiers of service and you have been asked to assess each application and choose a tier for each application.

Tier 1 No minimum, the monthly cost is $30.00 times the ADU
Tier 2 A minimum ADU of 10, and the cost is $28.00 times the ADU
Tier 3 A minimum ADU of 1000, and the cost is $22.00 times the ADU

The minimums mean that for Tiers 2 and 3 if the customer's ADU is less than the minimum in a given month, the provider rounds up to the minimum that month.

A. (10 pts)
In addition to cost, what factor(s) do you need to consider before deciding which tier to choose for a given application?

In addition to cost we should consider the following:[4] [5]

1) **Scalability, performance, and compliance requirements**: Determine each tier's capacity to manage increased demand without degrading performance. Consider any special compliance or security requirements your application may have. Higher tiers may provide improved performance and compliance features, which are critical for specific types of applications, such as those that handle sensitive data or require high dependability.

2) **Unused Capacity:** This is the expense of cloud resources that have been assigned but are not used. Tiers with minimum usage criteria may result

[4] https://phoenixnap.com/blog/how-to-choose-cloud-provider
[5] https://www.ziffity.com/blog/how-to-choose-the-right-cloud-service-provider/

in us paying for wasted capacity, so we have to make sure that we have planned, and it closely matches the tier's minimum requirements (especially for Tier 2 and Tier 3).

3) **Availability:** Examine the service level agreement (SLA) for each tier, paying special attention to the availability guarantee. This is critical for high-availability applications.

4) **Reliability:** Consider each tier's track record of dependability. Higher tiers may provide more stable infrastructure, resulting in fewer service interruptions.

5) **Uptime Guarantee:** Examine the uptime assurances offered by each tier. Applications that are vital to corporate operations or customer-facing services may necessitate a higher uptime guarantee tier.

6) **Hot Cold Starts:** Cold starts can cause delays in function execution, negatively impacting user experience, particularly in time-sensitive applications. Hot starts, on the other hand, offer faster reaction times but are dependent on the frequency with which the program is used. Understanding and planning for these elements is therefore crucial in selecting the best service tier to balance cost, performance, and user satisfaction.

7) **Long-term plan to consider:** This includes potential changes in application usage or growth. A tier that seems more expensive in the short term might offer better cost efficiency in the long run due to factors like better performance, scalability, or included services. Also, consider different tiers across different cloud providers. Hence this also makes us aware of vendor lock-in.

B. (10 pts)

Suppose a given application could use any tier. Considering only the cost, at what values of estimated ADU will each tier be best? (Remember to explain how you arrived at your answers.)

1. **Tier 1 vs Tier 2 Analysis:**

    - Tier 1 cost is $30.00 times ADU.
    - Tier 2 cost is $28.00 times ADU, with a minimum ADU of 10.

      The break-even point occurs when the costs for both tiers are equal. So, we set $30.00 × ADU = $28.00 ×ADU (for ADU >= 10).

      Solving this equation, the break-even point is when $30.00 × ADU = $280.00 (the cost for the minimum ADU in Tier 2).

      Therefore, ADU = $280.00 / $30.00 ≈ 9.33

2. **Tier 2 vs Tier 3 Analysis:**

    - Tier 2 cost is $28.00 times ADU, for ADU >= 10
    - Tier 3 cost is $22.00 times ADU, with a minimum ADU of 1000.

      The break-even point occurs when the costs for both tiers are equal. So, we set $28.00 × ADU = $22.00 × ADU (for ADU >= 1000).

      Solving this equation, the break-even point is when $28.00 × ADU = $22,000.00 (the cost for the minimum ADU in Tier 3).

    - Therefore, ADU = $22,000.00 / $28.00 ≈ 785.71

Hence my final answer regarding the best tier for ranges of ADU is:

- **For ADUs below 10, Tier 1** is the most cost-effective.
- **For ADUs from 10 to 785**, **Tier 2** is the most cost-effective.
- **For ADUs higher than 786: Tier 3** is the most cost-effective.

Problem 6 (20 pts)

A software engineering team at your company has been using traditional software development for many years. Now that your company has adopted hybrid multi-cloud to handle all their computing, the CIO says that everyone must shift to using DevOps. When your manager explains DevOps to your team, one of your fellow engineers become upset and say, "We've been doing software engineering for twenty-seven years, we have a great track record, and no one else around here knows anything about software development. Why should the operations guys be telling us how to build software when they've never done it?" Find five examples where the intended deployment methods used for cloud-native software add requirements that limit the choices software engineers can make when designing and building software.

These are the main examples that limit the choices software engineers can make when designing and building software:

1) Cloud-native apps frequently use a microservices design, which divides programs into smaller, independently deployable services. This calls for thinking in terms of linked, independently scalable components. Furthermore, this design imposes network configuration limitations, since services must interact properly over a network, and concerns like shared IP addresses and port conflicts (as seen in Kubernetes) must be controlled.

2) Cloud-native apps are often built to be stateless, which aids scalability. This has an impact on data and session state management, which often requires the usage of external storage and databases. Engineers used to design stateful applications must modify their ideas to achieve statelessness, which may be a major conceptual change.

3) Working in a hybrid multi-cloud environment adds new security and compliance concerns. Software must be developed that meets the security standards of various cloud providers as well as to comply with various regulations. Depending on the compliance and security standards of the various cloud environments involved, this might limit options for data management, storage, and even functionalities.

4) DevOps emphasizes CI/CD pipelines to automate testing and deployment. This demands designing software with automation in mind, which influences the coding, testing, and deployment processes. Furthermore, cloud-native

deployment frequently entails the creation of templates or manifests (as in Kubernetes), which further specify how applications are deployed and maintained.

5) There are specialized needs for data management and storage in distributed systems, especially when adopting paradigms like MapReduce in Hadoop. Because technologies such as the Hadoop Distributed File System (HDFS) are suited for these contexts, software programmers must build their data handling and processing methods to be compatible with these systems.

Problem 7 (20 pts total)

You have been hired to work as a software engineer for a company that has adopted the CI/CD approach to software development and deployment.

A. (4 pts)

Another new hire tells you that he is prepared to jump in because he has used a text editor and gcc extensively to create software in grad school. What do you tell the new hire?

I would tell him that knowing gcc and text editor experience is great for software development. However, in our CI/CD environment, he would also need to familiarize himself with the following: [6]

1) Tools like Jenkins or Travis CI automate the building, testing, and deployment processes, ensuring consistent and frequent integration of code changes.

2) Systems like Git allow developers to collaborate on code by tracking changes and supporting branch management for features and fixes.

3) Automated tests are run as part of the CI pipeline to ensure code changes do not introduce new bugs. (Example: CircleCI)

4) Tools like Maven or Gradle compile code, manage dependencies, and package the software into deployable artifacts.

[6] https://about.gitlab.com/topics/ci-cd/

5) Techniques such as blue-green or canary deployments are used to release applications with minimal downtime and risk.

6) Continuous monitoring tools provide real-time performance data, and feedback loops help teams respond quickly to issues.

I would also help him in any way and provide support at work if he needs it.

B. (6 pts)
Your company is considering the sandbox, canary, and blue/green approaches for cautious deployment. You have a meeting with the CFO, who is focused entirely on cost. The CFO asks you to give a 1-sentence description of each option, and briefly explain the financial costs for the option.

1) **Sandbox Trial**: Sandbox trial involves creating an isolated, replica environment to test new software modules without jeopardizing the production system.
   Financial Cost: Due to the requirement to duplicate the whole production environment, this solution incurs additional expenditures, potentially raising infrastructure and maintenance charges.

2) **Canary Deployment:** Canary deployment progressively deploys the new version to a limited fraction of customers first, acting as an early warning of potential difficulties before full deployment.
   Financial Cost: Because it leverages the current production infrastructure, it is typically cost-effective, although there may be fees associated with monitoring and administering the segmented user groups.

3) **Blue/Green Deployment:** Blue/Green deployment maintains two identical production environments: one (Blue) for testing the new version and one (Green) for running the current version, with a switchover once the new version is ready.
   Financial Cost: Because this strategy necessitates the upkeep of two full-scale settings, the expenses are comparable to Sandbox trials, particularly in terms of resources and operational overhead.

C. (10 pts)
After you meet with the CFO, you have another meeting, this time with the CIO, who is focused entirely on finding out which option absolutely will be the safest (so a customer never has a bad experience with a new version of the software). For each of the three approaches, explain to the CIO what can be done to ensure that new versions of software do not make customers unhappy. Be careful to remember all aspects and their consequences.

1) **Sandbox Trial:**
   a) Having a completely separate environment allows us to extensively test new versions without influencing the actual production environment, lowering the possibility of defects or issues affecting consumers.

   b) We can conduct rigorous functional, integration, and stress testing to identify any potential mistakes before they reach the production environment.

   Major Consequences:
   a) Even if the version has been thoroughly tested in a controlled environment, real-world issues may still surface when it is put into production owing to changes in data, user behavior, or external integrations.

   b) Higher Costs: The financial and resource expenditure required to replicate the manufacturing environment might be substantial.

2) **Canary Deployment:**
   a) Begin by introducing the new version to a limited, carefully selected set of 'canary' users. Before wider deployment, constantly monitor their experience to discover and rectify any difficulties.

   b) Implement real-time monitoring and a quick rollback mechanism. If an issue emerges, return to the previous version as soon as possible to minimize consumer damage.

   Major Consequences:

   a) Limited User Exposure: Because the new version is initially seen by a limited fraction of users, it may delay the identification of some faults that are only obvious in larger-scale usage.

b) Variable User Experience: During the canary phase, various users may have access to different versions, resulting in inconsistent user experience.

3) **Blue/Green Deployment:**

a) Run two identical environments to allow full-scale testing of the new version in a production-like scenario without interfering with the present user experience.

b) If the new version in the Blue environment is judged stable, switch traffic to it immediately. If problems emerge, you may immediately go back to the Green environment.

Major Consequences:

a) Resource-intensive: You'll need twice as many resources to sustain two full-scale environments.

b) Environment Drift: The Blue and Green environments may drift apart in terms of setup and data over time, which might create concerns when switching over.

In conclusion, while each strategy has advantages, the Blue/Green deployment provides the most robust means to test new versions in a production-like environment and immediately switch back if problems develop, reducing the chance of customer displeasure. This strategy, however, is resource-intensive and needs careful administration to guarantee that both environments remain consistent with one another.

Problem 8 (20 pts)

A company that runs automated factories uses monitoring and control software to ensure a given factory is running smoothly. The monitoring software watches the status and production levels at every factory. At any time, a company executive can ask the monitoring software to display a web page that shows a graph of the output of each factory over the past hour, 24 hours, and past year along with an indication of whether the output is normal, low, or high (the calculation must take into account the time of day, weekends, and planned shutdowns (e,g., for maintenance). The executive can zoom in on one factory and see more detail. In addition to monitoring, the software needs to handle problems and emergencies. Each piece of machinery in each factory has an electronic monitor and control system attached. The electronic monitor must be polled over the network every 12 milliseconds. On a given poll, the electronic monitor can report: normal operation, an emergency jam that requires the machine to be shut down immediately (within 15 milliseconds), or a minor problem that allows the machine to work but makes it run at a slower rate. When a problem causes a machine to slow down or an emergency causes a machine to be stopped, the software must activate a backup assembly line, either in the same factory where the problem occurred or in another factory. The idea is to compensate for lower output and keep the total production stable. To minimize shipping costs, if no more backup lines are available in a given factory, the ideal solution choose a nearby factory in which to activate a backup assembly line. All the factories are in the US, and they each have a miniature edge computing data center located on the factory property. In addition, the company has a set of regional fog data centers, where each regional center covers a geographical region that contains at least three factories. Finally, the company runs a private cloud data center located in the middle of the US. You have been asked to create software that handles monitoring and control of the factories. Describe the software you would run in the edge data centers, regional data centers, and cloud data centers, and tell how you would arrange for the pieces to interact.

I would run the following software: [7]

1) **In Edge Data Centers**: Ideally, perform low latency tasks here. So, I would run advanced ML models for immediate analysis of the sensor data. Every 12 milliseconds, I would use real-time data processing to check machine status. This would also include software for immediate reaction systems for emergencies, such as shutting off a machine in the event of a jam within 15

[7] https://www.redhat.com/architect/cloud-industrial-edge-architecture

milliseconds. Based on the severity of the problem and the availability of backup assembly lines, the software would activate backup assembly lines within the same facility or neighboring factories, with local decision-making capabilities. I would also accommodate software here that helps the onsite workers to see real-time updates and visualization tools. Lastly, this would reduce the strain on central data centers by processing and summarizing data for higher-level analysis.

2) **In Regional Data Centers:** This would contain software for advanced analytics platforms. It would aggregate data from multiple edge centers in that region and provide a regional overview. This would mean it would consider multiple factories and then optimize resource allocation. It would also include AI/ML software for predictions based off of aggregated regional data. This software would also help in backup control coordination when local edge centers can't handle the load, or when communication is needed between factories. Lastly, software will be included to handle scheduled maintenance downtime across the region.

3) **In Cloud Data Centers:** This would finally be somewhere in the center of all factories ideally. This would include software for purposes where latency of some milliseconds isn't critical. So, it would have big data computational abilities for an in-depth analysis of all the factories. It would have graphs for different time durations with status indicators so data analysts can zoom in on specific details. In some cases, it would intervene with local factories if there is a large-scale emergency. Since this is at the top of the hierarchy, it would have optimized software and hardware to store a lot of data for historical referencing, making predictions, and strategic decision-making.

Since this problem is of IIOT(Industrial Internet of Things), we should use OMG (Object Management Group). DDS(Distributed Data Service) is an OMG standard that I would use to arrange all the pieces to achieve our goal. In the setup, the Databus acts as the central communication hub with gateways to make sure data flows across different levels. This calls for efficient and selective communication in a secure fashion. Edge Data Centers will use DDS for real-time monitoring and control communication with machinery, making sure communication latency is low. Edge centers will publish everything on Databus, selecting relevant data streams. Regional Data Centers will utilize DDS to efficiently handle a variety of data streams and aggregate and analyze data from numerous edge centers. Then they will publish aggregated data and receive cloud center directions while managing

regional coordination. Finally, the cloud data center will conduct in-depth analysis and strategic decision-making while communicating with DDS. A cloud center can subscribe to aggregated data from regional data centers or particular data from edge data centers.

A diagram in the textbook:

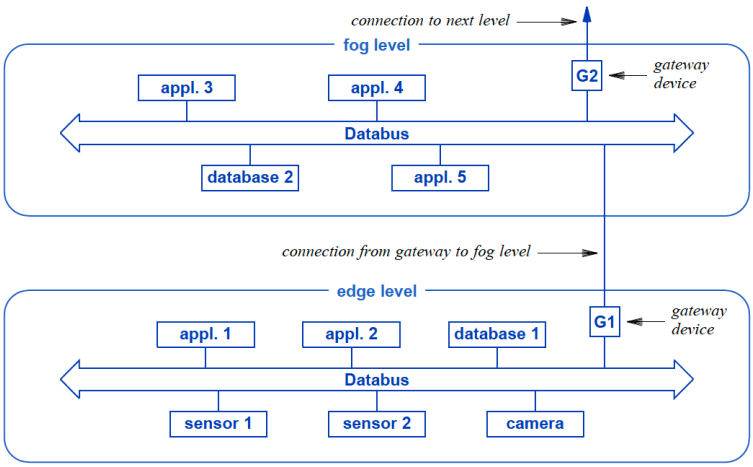


Figure 16.5 The concept of a single DDS Databus that spans multiple levels of a hierarchy. An application at any level can subscribe to receive data from an application at an arbitrary level.

Edge Level: At the edge level, such as in factories, applications and sensors would process data in real-time and handle quick control reactions (such as shutting down a machine). They would notify the Databus of their state.

Fog Level: Regional data centers that combine data from numerous edge centers and provide higher-order analytics and coordination are represented by the fog level. When compared to edge-level processing, this level would handle more complicated decision-making that is slightly less time-sensitive.

Gateways (G1 and G2): These serve as communication bridges between levels, enabling data and orders to move upward and downward. They efficiently filter and forward data, ensuring that only relevant data is delivered to preserve performance and reaction times.

Databus: The DDS-enabled communication architecture that connects all applications across tiers. It supports a publish-subscribe interaction architecture, in which different components of the system may publish data or subscribe to receive data, allowing interaction between monitoring software, control systems, and executive oversight tools.

Problem 9 (20 pts total)

A company operates a container shipping port where large ships dock and deliver containers. The company is designing a new automated system. Once a captain moves an incoming barge within sixty feet of a pier, robot tug boats will maneuver the barge close to the pier so an automated mooring system can lock onto the barge and pull it into place. After a barge moors, customs officials must approve unloading. If approved, automated cranes will begin to remove containers from the barge and place them on railroad cars for transport to their final destination. You have been hired to build software that handles the robot tug boats, the locking system, pulling the barge into place, and unloading. When moving a barge, your software must repeatedly read position sensors and respond quickly to ensure the barge reaches the pier without crashing into it. (The engineers who designed the tug boats and mooring system have specified that the software must respond within 100 milliseconds when an imminent crash is detected.) Once a barge has been locked in place, your software must send a message to a set of computers at the local port authority to inform the customs officials that the barge has docked and is waiting for their approval before initiating unloading. Each embedded processor in the system will have a backup that takes over automatically if the main processor fails. Your system must feed the same commands to the main processor and the backup. In an emergency, humans in the local port authority can send a message to all parts of the automated system that causes it to stop. Your software will work in many ports and must constantly upload status to the company's cloud software which will allow the company to monitor the status of each barge.

A. (14 pts)
Provide a brief sketch of pieces of the system and describe what each piece of software does

1) **Robot Tugboat Control Systems:** This would include software for managing robot tugboats that lead barges to the pier. Capabilities include receiving real-time location data from barge sensors, processing this data to identify needed motions, and sending directions to tugboats for precision maneuvering. Also, this system would have software for responding under 100 milliseconds to avoid collisions.

2) **Automated Mooring Systems:** When a barge is in place, it receives signals from the tugboat control system, engages the locking mechanism, and ensures the barge is firmly tied. This would include software for the automatic mooring system, which secures the barge upon arrival.

3) **Communication Systems**: This system should send notifications to port authority computers when a barge docks, awaiting approval from customs authorities, and initiating the unloading procedure after clearance is received. The software for this system would include methods for coordinating with local port authorities to gain customs clearance.

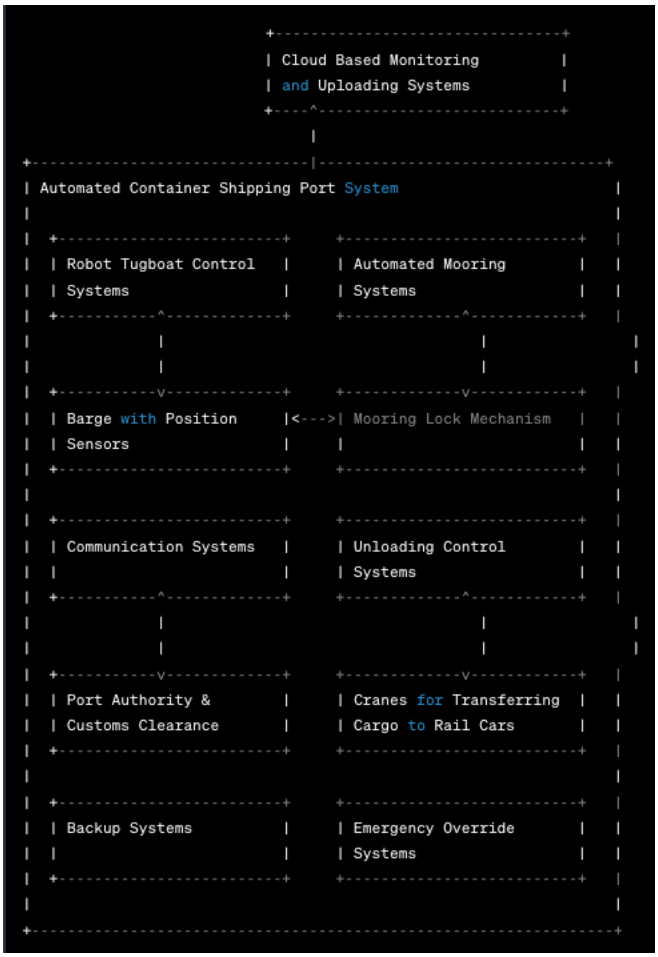
4) **Unloading Control Systems**: After seeking approval, this system will get activated. This system would be in control of the cranes that transfer cargo from barges onto train carriages. It would include software for the operations of cranes to grab and set containers, organize container sorting and placement with logistics, and guarantee safe, efficient offloading.

5) **Backup Systems:** This system would add a certain degree of redundancy and ensure that all the systems run continuously. The software would continuously feed identical commands to both the main and backup processors, smoothly switching to the backup in the event of a main processor failure.

6) **Emergency Override Systems:** This system would allow for human involvement in crises. The software would include the ability to receive stop instructions from the local port authority and quickly cease any automated activities.

7) **Cloud-Based Monitoring and Uploading Systems:** The system would provide real-time status updates and operation monitoring. It would be used to upload data to the company’s cloud as well. The software would have capabilities to continuously upload data to the company's cloud platform to monitor barge status, system performance, and operational efficiency across many ports, as well as provide real-time status updates.

Basic sketch for the following systems: [8]



The barge and mooring lock mechanism are linked by Robot Tugboat Control Systems and Automated Mooring Systems.
The port authority is linked with communication systems for customs clearance.
Cranes are used for freight transportation in Unloading Control Systems.
Backup and emergency override systems are essential components of overall safety and redundancy measures.

B. (6 pts)

Suggest a technology or technologies that you could use to implement the software and describe why you would choose it or them.

[8] ChatGPT for making a basic layout diagram

1) **Edge Computing**: The technique of processing data closer to where it is created rather than transferring it to centralized data centers is referred to as edge computing. I would choose edge computing in the context of an automated container shipping port system since it would include installing computing resources on or near the barges and tugboats. This method shortens the time it takes for data to travel between the sensor (source) and the processor, resulting in lower latency.

2) **Virtualization and Containers**: In the context of the port system, different system components can operate in independent virtual machines, guaranteeing isolation. Tugboat control software, mooring systems, and communication systems, for example, can all run on their virtual machines. Because VMs may be dynamically allocated resources based on demand, it enables effective resource usage. This adaptability is critical in a dynamic and limited-resource environment like a maritime port.

3) **RTOS (Real Time Operating Systems):** I chose this since it can deliver rapid and constant reaction times, which is crucial for jobs such as regulating tugboats and mooring systems in a shipping port. RTOS guarantees that the system responds quickly, preventing accidents and ensuring vessel operations are safe.

4) **IoT and Sensor Technologies**: To allow real-time data gathering and transmission from barges and tugboats, I would use the Internet of Things (IoT) and Sensor Technologies. This option increases safety by continually monitoring vessel movements and conditions, allowing proactive reactions to possible hazards, and improves operational efficiency through data-driven decision-making, adding to the port system's overall efficacy.

5) **Cloud Computing and Big Data Analytics:** I would use Cloud Computing for scalability, data storage, and processing capacity necessary to efficiently process large amounts of operational data from various ports. This functionality allows for real-time data management and analysis from a variety of sources, resulting in better decision-making and predictive maintenance. Big Data Analytics supplements this by collecting important insights from the vast amounts of data created in port operations, streamlining procedures, increasing operational

efficiency, and detecting trends that drive decision-making, eventually improving the overall performance of the port system.

6) **Machine Learning/Artificial Intelligence**: For improving crane operations during loading and unloading, machine learning and artificial intelligence are critical. Data may be analyzed by ML algorithms to discover the most effective ways to handle freight, decreasing operational time and increasing efficiency. Furthermore, AI-driven analytics may reveal useful patterns and trends in data, improving decision-making throughout the port system.

7) **Network Communication Protocols:** Robust network communication protocols are required to ensure dependable and secure data transfer and coordination between port system components such as tugboats, mooring systems, and control centers, and even central company cloud servers. These protocols provide seamless communication, real-time monitoring, and the transmission of crucial information, assuring the automated container shipping port system's efficient and safe functioning.

Problem 10 (20 pts total)

You have been hired by a company that has created 120 microservices. As expected, some microservices use other microservices, and the set of microservices has been built by dozens of software engineers, and the work has been incremental over many years. After the last major change, almost a year has passed and everything ran smoothly. Then, one day the number of instances of 67 of the microservices started to increase rapidly. The vice president who handles IT decided that they should all be shut down (to avoid high costs). Gradually, they restarted microservices, and each appeared to work without excessive instances. Although the crisis has passed, your manager remains worried. During a team meeting, your manager asks, "How can this happen?" Everyone on the teams starts to speculate about the cause. One of the engineers quips, "We don't know what happened, but it could have been a lot worse — at least we didn't create a deadlock."

A. (6 pts)
Explain what a deadlock is.

A deadlock in computing is a scenario in which numerous processes are trapped in a loop, unable to advance because one is awaiting resources or actions from the others. This happens when processes cling to resources while waiting for other resources, resulting in a vicious circle of reliance that stops any of them from proceeding. Deadlocks are especially problematic in distributed systems such as microservices, where services may wait for one other endlessly over a network, resulting in system inactivity.

To prevent or resolve deadlocks, system designers must use tactics such as timeout mechanisms, statelessness, and careful resource management. Because of the various interdependencies and communication layers in cloud-native and distributed systems where services are deployed across servers or regions, recognizing, and resolving deadlocks is difficult. To enable smooth operations and avoid these stalemate instances, efficient design patterns and effective monitoring tools are required.

B. (6 pts)
Explain why microservices that use a scale-out approach do not usually cause a traditional deadlock.

Because of their design and operational principles, microservices adopting a scale-out architecture often prevent classical deadlocks. Each microservice in this architecture is supposed to be self-contained and weakly connected with others. They perform specialized functions and interact via well-defined interfaces, eliminating the complex interdependencies that can lead to deadlocks in monolithic systems. Furthermore, the scale-out strategy enables dynamic resource allocation. Microservices may be scaled up or down based on demand without waiting for other services to release resources, reducing the possibility of stalemate. Traditional deadlocks are less common in such setups due to the flexibility of resource management and the independence of microservices.

C. (8 pts)
Your manager has heard about using models to understand behavior and assigns you to build "some sort of model" that can help. The manager says, "Even if a model can't prevent future problems, it would be great if the model could at least identify potential causes, giving the software engineers a place to start looking." Tell what type of model you suggest, sketch how it would work, and explain how it could help software engineers check for the underlying cause of the problem.

For modeling the behavior of microservices to identify potential causes of issues, I suggest creating a Dependency Graph Model. This model visually represents the dependencies between various microservices. Each node represents a microservice, while an edge connecting two nodes represents a dependence. This visual depiction makes identifying possible deadlock conditions, such as circular dependencies, easier. By studying this graph, software developers may identify services that may be causing deadlocks and alter their interactions or dependencies accordingly. This proactive strategy aids in the management of complexity and the avoidance of difficulties in cloud-native systems.

An example :

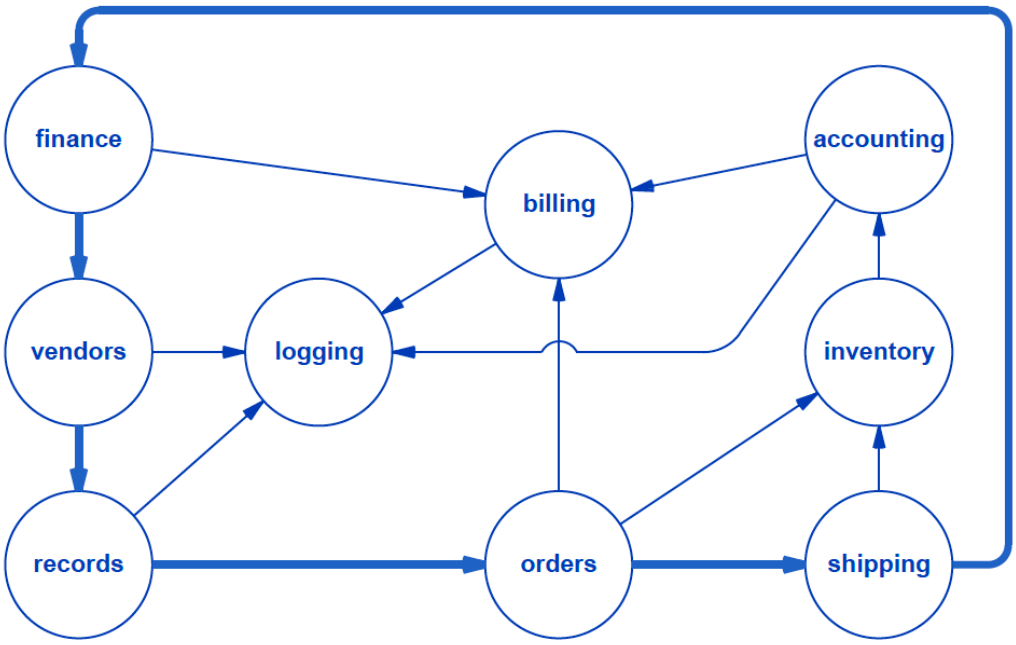


Figure 18.2 A pictorial representation of the graph for the dependencies from Figure 18.1 with a cycle shown highlighted.

In practice, this approach would aid software developers by explicitly demonstrating how services are linked. When a problem emerges, engineers may rapidly look at the graph to determine which services may be affected or contributing to the problem based on their dependencies. This focused approach to troubleshooting saves time and effort by narrowing down the places to check.

By regularly updating this model when new services are introduced or current ones are adjusted, the team may examine the impact of changes on the system's overall behavior, assisting in the prevention or at least anticipation of possible service dependency concerns.

THANK YOU ☺