SE 464 - Guest lecture study 1

October 1, 2023

1. Q: What is the ISO/IEC/IEEE 42010 standard definition of system architecture?

   A: Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.

2. Q: What is the practical definition of system architecture?

   A: A system's architecture codifies a set of decisions that are both hardest to change and have the most significant impact on the way the system manifests its quality attributes.

3. Q: Why doesn't the academic (ISO/IEC/IEEE 42010) definition of system architecture work well in practice?

   A: It doesn't give support in making practical decisions and implementing based on architectural reasoning.

4. Q: Why is observability important?

   A: With how complex software systems are today, with numerous services and interconnected components, observability plays a key role in keeping this service healthy and ensuring minimal downtime for its users.

5. Q: Observability is about providing tools to help monitor and detect real-time issues, guiding you to the roots of problems.

   A: It can also help you scale and optimize performance, either running more machines when demand is high, or lowering the count during off-peak.

6. Q: What is distributed tracing in the context of observability?

   A:

   (a) Distributed Tracing is a method to trace a request or call as it traverses through the various services and subcomponents of a distributed system.
   (b) It is used to provide an end-to-end view of how a specific request is handled by the system.
   (c) It can be used for performance metrics (eg. latency), debugging, and identifying bottlenecks.
   (d) It is commonly used in cloud native, and serverless applications.

7. Q: How are "clusters managed"?

   A: A cluster refers to a group of interconnected computers, servers, or nodes that work together to perform tasks or process data as a single system. Clusters are managed using cluster management and orchestration software such as Kubernetes to maximize the work a given cluster can achieve by reducing bottlenecks (load balancing) and monitoring system health.

8. Q: What is system architecture?

   A: A system's architecture codifies a set of decisions that are both hardest to change and have the most significant impact on the way the system manifests its quality attributes.

9. Q: Why is it important to intentionally choose architecture that maximizes specific quality attributes?

   A: It's hard to find an architecture that's perfect. Often there will be design tradeoffs. In Jeromy's case with his first business, this was flexibility vs load capabilities. Intentionally choosing architecture to maximize the most important quality attributes of a system will reduce the likelihood that the system will need to be redesigned.

10. Q: What is one way startups and established companies differ in terms of what quality attributes matter?

    A: Startups generally need to move fast so an easily extendable architecture is important. If you're at a big established company, other quality attributes like scalability, reliability, and performance are important.

11. Q: Name 4 quality attributes of a system.

    A: Reliability, scalability, security, testability, maintainability, extendibility, etc.

12. Q: What issue does Mr. Carriere take with the structural/classical definition of system's architecture?

    A: It is an academic definition, and does not give practical support while trying to make decisions with architectural reasoning.

13. Q: According to his own definition of systems architecture, system's architecture codifies a set of decisions that has two characteristics. What are these characteristics?

    A: 1) Hardest to change, 2) Have the most impact on the way the system manifests its quality attribute

14. Q: Mr. Carriere mentioned that he designed his first startup, Quack.com, very specifically. Which trait did he prioritize?

    A: He designed the system to be highly flexible and to prioritize flexibility above all other quality attributes.

15. Q: Definition of system architecture

    A: A system's architecture codifies a set of decisions that are both hardest to change and have the most significant impact on the way the system manifests its quality attributes.

16. He puts emphasis on the significance of architecture in a project by telling about his personal experience as a start-up founder. The founders didn't put a lot of intention in making the architecture of their product scalable. When they deployed it for AOL, they were crushed because their system could not handle their traffic. They had to redesign everything to manifest the new desired quality attributes.

17. As Jeromy talks about his startup's architecture's initial lack of scalability, no definition of scalability has been provided so the following provides more context. Definition of Scalability: The capability of a system to be adapted to meet new size / scope requirements. (Source: Week 2 Slides, slide 12)

18. Software Engineers should give decisions that are harder to reverse more thought and consideration than reversible decisions. Engineers should try to transform irreversible decisions to be reversible whenever possible.

19. Q: Are there any drawbacks to designing a system that can easily pivot/change?

    A: It may be difficult to scale if the system was built to pivot and change.

20. Datadog offers a suite of products to help customers monitor the health of their systems across infrastructure and applications.

21. Q: What are kinds of information that businesses may be interested in observing, to understand the state of their s/w system?

    A: business metrics application performance (latencies, memory and cpu consumption) incident management (how incidents occur, how long it takes for them to be managed)

22. Q: what is the old-fashioned way to capture state information about an application? printf - debug logs
    A: The major responsbilities of cloud providers are infrastructure management (physical equipment), resource allocation (computing resources), security, data storage

23. Q: What are some attributes that one would use to measure the health of a system?

    A: Availability, Performance, Efficiency, Business Performance, Security

24. Offers a suite of products and services to monitor and maintain the health (availability, performance, efficiency, business performance, secuirty, etc.) of infrastructure and application portfolios.

25. Q: If an implemented system that is not scalable suddenly has a spike of new users, which performance attribute would be most useful? Why?

    (a) Maintainability
    (b) Flexibility
    (c) Testability
    (d) None of the above would help

    A: b) Flexibility - This trait allows the team to quickly scale the system

26. Q: What is the observability problem and why is solving it important?

    A:

    (a) The observability problem, in the context of software engineering, refers to the accuracy that external outputs of a system can be used to measure the system's internal states.
    (b) Some examples of how companies can benefit from proper observability may include:
        i. Insights in real-time (Datadog offers low enough latency to make this feasible)
        ii. Better Bug Detection, Troubleshooting
        iii. Improved User Experience (Datadog is all about pleasing the user)
        iv. Optimizing Performance and Resources
        v. Application performance (uptime/downtime)

27. Q: If you were the software architect at early Datadog aiming to address the broader observability problem, what components would you design to build a system that offers more insightful metrics than, say, CloudWatch? Name and explain one such component.

    A:

(a) The question is open-ended, but explanations and examples for any one of the following components would be accepted: Log management, Application Performance Monitoring, Real User Monitoring, Synthetic Testing, Flows and Business Metrics, Workflow Automation, Incident Management, Security - Cloud / Application.

(b) Examples of direct applications of the components could be CPU/GPU consumption, disk utilization, other hardware utilization (reference: Jeromy showing his personal home computer metric usage)

28. Q: Name 3 different cloud providers that Datadog uses. What is the benefit of diversifying cloud providers? Feel free to use examples from Datadog's outage (or other companies) to illustrate those benefits.
A: Any three of AWS, GCP, Azure, Oracle b. While a bit harder to manage, diversification removes a single point of failure with regards to the cloud. During Datadog's outage, different providers had different ways of handling database clusters and some were easier/harder to recover than other (which was unknown in advance)

29. Q: What does the ingestion block in Datadog's architecture do?

30. Q: What are some of the functionalities/services that Datadog provides? Give 2 examples.

31. Q: What are some unconventional statistics that Datadog can monitor?

32. Q: Describe the purpose of the Monitoring/Alerting component of Datadog's system and how it is used by customers.
AThe purpose of the monitoring component is to make observations on the customer's data and ensure that if an anomaly occurs, the customers are notified in a timely manner through the appropriate channels such as Slack, or emails. Customers heavily depend on this component as they need to know if their operations are behaving as usual, and they should be notified instead of having to discover issues by logging into Datadog to manually check.

33. Q: What are 2 of the processing operations that Datadog performs on stored data and why (what benefits do these operations have)?
A: Indexing the information to facilitate faster searching. Generating aggregate metrics to eliminate redundant computation that needs to be performed on the fly for that same data. Note - aggregate metrics take multiple input values and then use them to produce a single output.

34. Q: One benefit of storing the ingested data is that it can be used for future querying by customers. How does the Query component of Datadog's system allow the user to interact with its stored data?

    A: Customers want to access their data in flexible ways, and Datadog provides a bespoke language for specific purposes such as metrics queries as well as a SQL-like language to allow its clients to search for highly specific information.

35. Q: Why is it important to store pre-aggregated metric data?

    A: To avoid the need for repeatedly computing metric aggregates on the fly.

36. Q: What is the purpose of monitoring?

    A: Monitoring is essential because we are observing metrics, logs, and data to identify customer needs. Customers want assurance that the data performs as expected and notifications for the same, which is critical.

37. Q: What benefits does indexing provide to data?

A: Data, on its own, is not very useful as it can be difficult to read and navigate. Standardizing it through indexing makes the data more accessible and understandable. This can be achieved by creating an indexing method, allowing us to search for and utilize the data effectively.

38. Q: Datadog distributes its observability system:

    A:

    (a) vertically by maintaining different implementations across different data-type layers.
    (b) horizontally by replicating the same set of layers across all its geographical regions.

39. Q: Different observability data types that are processed in the observability system

    A:

    (a) metrics
    (b) timestamps with numeric measurements as labels representing quantitative data on a web app
    (c) logs
    (d) traces
        i. distributed tracing through an application
        ii. a data type that essentially tracks the path requests will travel through in the web application
        iii. for example, a request enters the web app, goes through multiple meshes of microservices, goes through persistence layers, goes through some queues etc.

40. what is a persistence layer?

41. Q: Give an example of a quality property and an application where that property would be essential. A: An example of a quality attribute is durability. For example, in Datadog, durability is essential since losing customer data should never happen.

42. Q: Give one advantage and one disadvantage of having a detailed architecture diagram.

    A: Advantage: It can give more useful context to developers. Disadvantage: It can be harder to maintain to ensure it is still correct.

43. Q: Explain why you might want to host services in multiple regions, and why sometimes you might not want to do this.

    A: The advantage of hosting in multiple regions is two-fold: better reliability when one region goes down; and improved latency when users are closer to one region then another. The main disadvantage is added cost and engineering complexity.

44. Q: What does it mean for a system to have reliability? Why might this property be desirable for companies such as Datadog?

    A: A reliable system is one that is expected to be and is available. High availability is desirable for datadog because real people / companies rely on Datadog to support and observe their own products (some of which could be safety critical).

45. Q: Suppose you're designing a system that stores important files for a client. What is one quality attribute that may be desirable and why?

    A: Durability is one quality attribute that is likely to be desirable. A durable system is one that doesn't lose data (or very very rarely loses it). If we are asked to store important files for a client on our system, we will obviously not want our system to lose them.

46. Q: What is an ACK?

    A: Short-form for the term "acknowledgement", this is a signal passed between computers or processes to signify successfully receiving a message or data.

47. Q: what are the qualities attributes that are critical for data monitoring?

    A: Durability, reliability, and latency:

    (a) Durability: we should make sure that we don't lose the data.

    (b) Reliability: We can extract useful information from it.

    (c) Latency: how fast is it to access and present data to be used.

48. Data ingestion refers to the process of importing data from its source (in the current case, the customer).

49. Data metrics are values tracking something about the system over time (eg. latency, error rates, user actions, etc.) consisting of a name, a timestamp, and labels. Logs are a record of something that happened in a system (eg. events, warnings, errors, state info, anything else the user wants recorded about the system) Traces show the path a request follows during its lifecycle spanning different components in a system.

50. Q: What is Kubernetes?

    A: Kubernetes is a platform for container management. It provides functionality for deploying and monitoring containerized applications.

51. Q: Why is shared-use metadata tricky to manage?

    A: It can come in through many channels and it can be hard to fuse all these data points together to correctly retrieve the metadata. It is also very important to get correct or it will cause confusion.

52. In terms of reliability of data monitoring for datadog, ingestion is the most important property and ingestion has its own hierarchy with metrics being most important, followed by logs and other data points.

53. The second most important reliability property for datadog is notifications and monitoring. This is more important than being able to access the data because at least customers know something is wrong and do not have to dig around for it. This also explains why ingestion is the most important because ingestion is necessary for this property to work.

54. Q: Explain in detail what a reliable software architecture entails. What are the 3 different measures one should take to prioritize reliability of software, in order of their importance?

55. Q: What does ingest-to-serve time mean in terms of latency?

56. Q: What is the difference between durability and reliability, and their importance to software architecture design?

57. Q: Why is testability an important quality property?

    A: It allows for entities to scale fast (high velocity) as it ensures that the software you write actually works. You can write a lot of code fast, but without testing you can't have the needed confidence in your software to know if it works to be able to scale quickly. This is especially true in an iterative-based development lifecycle, where you are making small changes frequently.

58. Q: Why is choosing what software language to use an important architectural design decision?

    A: Choosing what language to use to build a particular architectural component is important because each language has its own advantages and disadvantages. For instance, one might choose C when designing architecture that has speed and memory as critical factors, even though it may have a higher learning curve and doesn't support object-orientated programming, unlike Python. That being said, there must be a clear purpose for adding a language because you don't want to end up having a zoo of languages that becomes hard to maintain.

59. Q: Why are non-operational quality properties also critical to the success of a business?

    A: Non-operational quality properties (eg. extensibility, maintainability, testability, modifiability) allow for a company to have velocity, which in turn allows them to adapt to their customers' needs better and more quickly.

60. Q: What is meant by service level indicators and what kind of service level objectives should we set relative to the respective indicators?

61. Q: What is are some examples of non-operational qualities and operational qualities? Should we focus more on operational quality properties of a software more than non-operational quality?

62. Q: What is the priority when it comes to developing software features or focusing on perhaps testability(non-operational) and hence reliability(operational)?

63. Q: What is the relationship between service level indicators and service level objectives?

    A: Service level indicators are measurements that track user-facing operational quality attributes, while service level objectives set the target goal for service level indicators.

64. Q: Explain what a service level agreement is, and what a common consequence for violating it is.

    A: A service level agreement is a contractual commitment between a service provider and receiving customer. It defines that the provider will guarantee that their service meets certain service level objectives. If a service violates an SLA, the service's provider may incur a financial penalty.

65. Q: Explain how service level indicators help in achieving service level objectives, and what roles do service level agreement play in this process?

    A: Service level indicators (SLIs) are used to measure service quality attributes. Service level objectives (SLOs) are used to set target quality goals such as the uptime of the service. A service level agreement, (SLA), is a contract linking SLOs together. If SLOs aren't met, such as the uptime of a service was less than the agreed amount in our SLA, the service provider may incur penalties.

66. Q: What 2 things should be top of mind when defining a Service Level Objective (SLO)?

    A: The bounds of the Service Level Agreement (SLA) and the customer experience at a given level of performance. Lower performance in certain aspects may be acceptable, while in others the preferred level of performance may be much higher than in the SLA.

67. Q: How does a Service Level Agreement (SLA) differ from a Service Level Objective (SLO)?

    A: A SLA is a SLO but contains contractual agreements between the provider of a service and its customer.

    Typically contains more business impact than a SLO (e.g. penalty or fines for non-compliance)

68. Q: How is an error budget typically used?

    A: An error budget defines an acceptable level of failures a system can have before some recovery action should be started, often taking the form of an alert or automated rollback.

69. Q1: What are the features of a good SLO?

    A:

    (a) Meaningful: when defining SLOs, the objectives must line up with client needs and experiences. Choosing SLOs that don't align can lead to wasted time and resources

    (b) User-oriented: SLOs should reflect the way clients perceive the non-functional requirements of the system

    (c) Actionable: SLOs should provide insights on why the system did not meet certain metric goals so that the team can act on it

    (source: https://www.usenix.org/system/files/nsdi20spring_hauer_prepub.pdf)

70. Q: What are compliant and non-compliant events?

    A: Compliant event: an event that occurs in the system measured by an SLI, that meets the corresponding SLO set for this SLI

    Non-compliant event: an event that occurs in the system measured by an SLI, but that does not meet the corresponding SLO set for this SLI

71. Q: You notice your system's error budget is -25%. What does this tell you about the number of compliant and non-compliant events?

    A: This tells you that the system has exceeded the error budget. You cannot determine the exact number of compliant or non-compliant events since you do not know what the original error budget was (and whether it was defined as a number of allowed non-compliant events or an allowable ratio of non-compliant events to total events) nor do you know how many events occurred. All this tells you is that there are too many noncompliant events to achieve the SLO.

72. Q: Why did some customers of Datadog not use the service for the length of the full outage (two whole days), when they were communicated to that there was a partial recovery beforehand?

    A: Always double check every update, as though it can be compatiable with everyone else, sometimes it won't be for you.

    Always roll out updates in incriments, as if there is an issue you can spot it before it corrupts the whole network.

73. Q: What is durability? Give an example of how high durability can be achieved.

    A:

    (a) Durability is an operational quality property that refers to the ability of a system to keep data once it has been ingested.

    (b) For instance, DataDog can be considered to typically exhibit very high durability, since data, once received by a customer and acknowledged by DataDog, is very rarely lost.

    (c) High durability can be achieved through the use of multi-layer persistence, which emphasizes the concept of defense in depth by storing data in multiple levels - first in a local disk, then into a queue to be stored in a database.

74. Q: What is reliability? Give some examples of how reliability can be measured and affected.
A:

    (a) Reliability is an operational quality property that pertains to a system's availability to customers. It's important for good software to be available to customers almost all the time, and therefore be very reliable.

    (b) For example, if DataDog were to become unavailable, its customers would lose visibility into the behaviour of their production applications.

    (c) One measure of reliability is host availability, which can be affected by factors such as a loss of server capacity, as seen in the case of DataDog.

75. Q: Which quality property was primarily affected in DataDog's service outage, and why?

    A:

    (a) The primary quality property affected during DataDog's service outage was reliability (the availability of services to customers).

    (b) This outage prevented users from effectively monitoring their production applications.

    (c) Although DataDog was able to progressively recover their systems in the days following the outage, the predictability and visibility remained insufficient. As a result, customers were unsure about the trustworthiness of the data that they were seeing.

76. Q: What's the difference between the durability and the reliability of a software system?

    A: The durability of the system refers to how well it preserves the data associated with it. In the case of DataDog's outage, the system was durable because the majority of the data that was ingested into the system was preserved, and past data stayed preserved. In contrast, the reliability of a system refers to the system being consistently available to users and its ability to perform the tasks that it is designed to do and that users expect.

77. Q: How has multi-layer persistence helped to increase durability of software systems?

    A: The proliferation of multi-layer persistence has, as a side effect, considerably increased the durability of software systems. In a traditional service without multi-layer persistence, all of the data in the system is stored directly by some monolithic subsystem. However, with multi-layer persistence, incoming data is often stored in a hierarchy of queues and ingestion systems that tend to be separate systems. These systems are often run on different servers, in different geographical locations, which reduces the probability that an outage in one component will cause data loss across all of these systems.

78. Q: How and in what cases might some quality properties of a system be more important to preserve in an outage?

    A: This depends heavily on the type of system under consideration. In the case of DataDog, persistence was more important than availability as the crucial data flowing in is used for analysis and metrics. While the service was unreliable during the outage, data that was generated was preserved even though it was not available. In some other cases it might be much more important to preserve reliability. For example, an aircraft controller that logs its actions might prefer to have the logs fail/not persist before its reliability was impacted. These priorities can guide system architecture and help the system fail gracefully depending on the context of its use and role.

79. The outage was caused by Kubernetes' networking stack. At Datadog, Kubernetes is organized hierarchically, there is a master, a control plane and children. During the outage, Kubernetes gave good management; however, made recovery difficult. The bulk of the effort in fixing the outage was spent on getting Kubernetes back online.

80. Kafka is an event streaming software. It enlists sharding which involves partitioning smaller instances from a larger instance. Kafka is an edge software and handled the outage well, which makes sense as it is farther from the source of the outage - Kubernetes.

81. What is a quorum-based system? A distributed system that requires a minimum number of votes from nodes (this vote threshold is the "quorum") to make a decision (usually regarding the performance of some operation, such as writing to a database). In Datadog's architecture, Cassandra was used as its database and did not perform well during the outage.

82. Q: What does the term "spooling" mean?

    A: In the context of the outage Jeromy was explaining, spooling is the process of saving / holding data temporarily before sending it to a queue for processing.

83. Q: What are the pros / cons of hierarchically organized (parent/child) Kubernetes control planes?

    A: Pro: Nice manageability properties Con: Hard to recover. Have to be done progressively with the recovery nodes in the right order.

84. Q: At what point does an entire quorum-based system die?

    A: Once slightly over 50% of the system dies.

85. Q: Do all cloud providers (GCP, Azure, AWS, etc) provide the same failure modes / failure processes?

    A: No ; there is no standard shared across cloud providers that dictates how nodes should proceed in the event of a failure. Specifically from Jeromy's lecture, AWS nodes running Cassandra lost their disk, and thus their data ; Azure and GCP nodes could be rebooted to recover their data.

86. Q: Why were most Kafka nodes still up during the Datadog incident?

    A They were still in Ubuntu 20, so they were unaffected by the security update, which was the culprit of the outage.

87. Q: What are some advantages and disadvantages of using multiple service providers (e.g. cloud providers) with different failure modes?
    A: One advantage is that you can recover some services more easily than others. It also introduces redundancy/heterogeneity to the architecture. One disadvantage is that you must use different operational procedures/playbooks for each service.

88. Q: How can one reinitialise a system with circular dependencies

    A: Through manual intervention. You have to let the system know to skip certain parts of their boot up process, get to a stable state so that other programs can start up and then finally come back to all the skipped setup.

89. Q: What are the advantages and disadvantages of having different software architectures and service providers?
    A: Pro: Redundancy, if one architecture or service provider fails, others are still up. Con: Maintenance is more difficult. Recovery is harder.

90. Q: What is a circular dependency in software architecture?

    A: Two or more modules depend on each other to function properly. Example, Google BigTable and Colossus, class A contains class B and class B contains class A.

91. Q: When you need to start the recovery process, how do you handle circular dependencies in the system?

    A: Break them. Manually inject data and/or short-circuit dependencies to start the process.

92. Q: Why are circular dependencies dangerous?

93. Q: What are some examples of failure modes?

94. When you try to recover a system, it is important to create a plan for booting up the most useful components first.

95. Cloud providers may not be able to handle the load of booting up too many instances at once. Hence prioritize useful components to help.

96. Multi-layered persistence is useful, but not without a plan to recover that data back into the system.

97. Q: What is a sharded system, and what quality of sharded systems helped them recover most easily?

    A: A sharded system typically refers to a distributed computing architecture where data is partitioned into smaller pieces and distributed over multiple servers or nodes; individual pieces are called shards and are independent of each other.

    Since shards do not have interdependencies, even when the control plane goes down (e.g. Kafka in this lecture), the individual shards can continue running. Hence the sharded system was most easily recovered.

98. Q: What is a quorum-based system?

    A: A quorum-based system is a type of distributed computing or decision-making framework where a specific minimum number of participants or nodes must be present and agree on a certain action or decision for it to be considered valid. This minimum number, known as a quorum, is essential to ensure the system's reliability, fault tolerance, and consistency.

    Often used in distributed databases, consensus algorithms to maintain data integrity and prevent issues like data inconsistencies, conflicts, and unauthorized operations.

99. Q: When does recovering a quorum-based system become problematic?

    A: From the lecture: problematic when a system needs all data to be queryable before it can do anything. Data integrity cannot be guaranteed if participants are inactive and the quorum cannot be reached.

100. Q: What is Apache Kafka and how does it help in dealing with user surges?

     A: Apache Kafka is a distributed event-streaming and processing platform that allows developers and companies to store and manage interactions between various services in their technology stack. Kafka is mainly used for service-to-service requests, acting like a bus that also automatically handles failures and retries, along with distributing the event load. During user surges, Kafka can help with making sure that events are spread evenly and provides temporary storage in case servers aren't able to immediately cope with the current event volume but will be able to in the near future.

101. Q: What are some considerations to take into account when deciding between local disk storage and cloud storage solutions?

     A: When choosing between local disk storage and cloud storage, it's essential to weigh several critical factors. Some factors include cost, security measures, data accessibility and mobility, scalability options, redundancy and backup strategies, reliability of your internet, and data volume. By evaluating

these factors, you can make an informed decision on whether to use local, cloud, or even hybrid storage to meet your specific storage needs.

102. Q: What are some negatives of utilizing homogeneity in your system?

    A: Utilizing homogeneity in a system can have several drawbacks. First and foremost, it can stifle diversity and innovation. When all components or elements in a system are too similar or identical, there is a lack of fresh perspectives and novel ideas, which are often born from differences and diversity. This can lead to stagnation and hinder the system's ability to adapt to changing circumstances. Furthermore, homogeneity can increase vulnerability to systemic failures. If all elements are built or configured in the same way, they are more likely to share the same vulnerabilities. A single weakness or point of failure can potentially lead to widespread issues or catastrophic failures across the entire system.

103. Q: What are the trade-offs associated with using a local disk for storage?

    A: Local disks offer better reliability, generally are more cost-efficient, and are more performant than using a hosted solution such as AWS EBS. However, if multiple members in a quorum-based system lose their data, using the local disk can be catastrophic.

104. Q: What is the most common reason for global outage?

    A A human or a system configuration change.

105. Q: What is a major drawback of having homogeneity in your system?

    A Homogeneity induces common failure nodes because a trigger would cause failure all over the system even if it affects a single one.

106. Q: According to the customer perceived value v.s. capacity graph, What is the capacity range associated with the highest customer perceived value during DataDog's outage?

    A: According to the graph, 80-100% is the range for the highest customer-perceived value.

107. In an ideal world, customers' perception of the company's value should vary linearly with the available capacity of the company. In reality, this is rarely true. Below a certain operating threshold A, the perceived value goes exponentially to zero. Above a certain threshold B, perceived value remains relatively stagnant and there's no point adding resources to boost server capacity. The goal is to achieve threshold B, or at least maintain it.

108. Q: What is the ideal server degradation to be expected?

109. Q: What are the three phases of customers' perception of company's value?

110. Q: What were the main issues with the recovery from the outage?

111. Q: How was the system architecture modified to provide customers value in times of low available capacity?

112. A vulnerability was accepted as a calculated risk; that not all regions will be unserviceable at a single given point of time.

113. Q: What is journaling and why is it useful?

    A: Journaling is a technique used to ensure data integrity and recoverability in the case of some failure/crash by keeping a chronological record of all transactions made to the database. In the event

of a failure/crash, the journal is consulted to bring the database back to a consistent state. It is crucial that before any transaction occurs, it is recorded in the journal - as opposed to performing it, then recording it as a failure could occur in between these steps. In this case, reverting to a state based on the now-compromised journal would likely ruin the integrity of the database.

114. Q: What is graceful degradation of an operating system? Why is it important?

    A: Graceful degradation is a design principle whose aim is to make sure that basic parts of the system still remain functional even if parts of it are failing. Graceful degradation is essential in preventing "Catastrophic Failure" where one failure leads to a chain-reaction of other failures resulting in system-wide shutdown. A system that implements Graceful Degradation ensures that users can still do essential tasks and access limited features in times of failure.

115. Big concepts mentioned in our time frame:

    (a) Journaling and sharding are patterns that worked.

    (b) Quorum-based stores did not Control plane architecture graceful degradation "If you don't test it, it doesn't work". They would do 'chaos-type' testing e.g. shutting down 50% of capacity to see how they react.

    (c) Losing customer percieved value of their data is much faster than recovering it.

    (d) There should be a mechanism in the architecture that since a percentage of available capacity (30%), the user must be able to choose to move away from the fail region.

    (e) The key is having a gradient zone which goes from high customer percieved value to lower values.

    (f) Backups are only useful if you can restore them.

116. Q: What is an example of large-scale chaos testing?
    A: Turns off 50% of capacity across AZs in one region set up in production-reflective configuration.

117. Q: How did Datadog use infrastructure decisions to recover customer trust?
    A Improved on sharding, journaling and reassessed things that did not work such as quorum-based stores.

118. Q: What product improvements did Datadog make that are customer-visible?
    A: Implemented graceful degradation, multi-region high availability.

119. Q: What are some issues that can arise when you focus solely on being "the best" when launching a product?
    A:

    (a) Spending too much time trying to perfect the product can decrease focus on other important aspects of your overall business.

    (b) May have market or investment opportunities taken by a product that beats yours to being the first-to-market.

    (c) May cause a reluctance to try and experiment with more creative ideas out of fear for making mistakes.

120. Q: What aspects beyond the quality attributes of the product are required in order for a business to recover after disaster and why?

    A:

    (a) Customer trust: Customers are the end users of any business's product, so even with the best product, if customers do not trust or have faith in the company, it is hard to get them to continue business with the company.

(b) Human connection: Communicating directly and honestly with the customers (especially if they are enterprises that need a clear picture of the situation, because it impacts their own operations) can give them reason to keep using your product.

(c) Reputation and Speculation: Some clients (like other enterprises) may be very risk-averse in nature, so having a product that has historically performed consistently can make occurrences of problems such as data loss be perceived as a standalone event rather than the beginning of a pattern of vulnerabilities.

121. Q: How does a company stay relevant and avoid stagnancy after it has been established?

A:

(a) Establish identity: Once your product/company has been brought to market, an important aspect that will both attract customers and maintain relevance in your given market is defining what gives your company recognition makes it unique.

(b) Evaluate whether the aspects of the service provided by the existing product need to be rebalanced, depending on target audience and market circumstances.

(c) Diversify into new areas that are able to be integrated into the product/company's identity, such as data security for a data monitoring company.

(d) Detecting failure is looking for anomalies and then working your way back. This is applicable for all types of systems. Since failure at Datadog is observed by Datadog monitoring Datadog they did not have the proper telemetry to monitor their information. In Datadog's case the failed nodes were obvious and they were able to solve the problem by tracing these nodes to the root of the problem, loss from an engineer's local disk.

122. Q: What are some possible setbacks for having a system monitor itself for failure?

A: Lack of proper telemetry which makes it harder to debug failures.

123. Q: What is a process of debugging failure within a system given limited information?

A: Monitioring the information given and tracing the routes.

124. Q: Beyond auditing the Datadog system to ensure future automated reboots did not occur, why was no other action taken to anticipate future global outages? What strategies were already in place to prevent global outages?

A: The risk of a global outage across 3 cloud providers was small enough that the cost-benefit did not warrant the large investment necessary for prevention. Strategies such as staggered deployments, integration tests, and different operating systems on different cloud providers further reduced the risk that direct human changes would cause global outages.

125. Q: What steps can you take to prevent data loss when using Cloud providers?

A: Redundancy. Data should be stored in multiple places until it is safe to delete copies. At Datadog, this was done by saving records on the local disk upon receipt. Frequent backups, with the ability to restore the state to a previous working state, is necessary to minimize lost data in the event of an outage. It is especially important that these backups have the ability to restore the state in a timely manner, otherwise a backup by itself is useless.

126. Q: Why is the perceived value vs compute capacity curve non-linear?

A: The loss of compute is unlikely to impact all components of a system equally - one component is likely to suffer an outsize loss of resources and could become the bottleneck. What a customer perceives

is also important - even if 20% of the system is down, if the customer relies heavily on that 20% of service it will appear as though most of the system is down. A customer might also believe that other related but functional components are unreliable or out-of-date.

127. Is it worth building your architecture to withstand the most catastrophic types of failure (global etc failure.)? Consider the cost-benefit ratio.

128. Customer experience is the most critical criteria in any software system.

129. Chaos testing allows for locating the problem in specific regions however it needs to be used for planning recovery in the worst case scenarios, it cannot stand on its own.