# 1. System Design: Distributed Monitoring

1. **Introduction to Distributed Monitoring:**

Discusses the critical nature of monitoring in preventing costly downtimes.

Explores the different types of monitoring necessary for comprehensive oversight.

1. **Prerequisites for a Monitoring System:**

Details essential concepts such as metrics and alerting, fundamental for any monitoring system.

1. **Monitoring Server-side Errors:**

Defines the requirements and high-level architecture for monitoring server-side operations.

Provides a detailed breakdown of system components and data visualization techniques.

1. **Monitoring Client-side Errors:**

Highlights the importance of capturing client-side errors.

Describes the design of a system specifically for monitoring client-side issues.

### Additional Features

* **Rate Limiter, Blob Store, and Distributed Search:** Briefly touches on other distributed system components that ensure efficient operation and scalability.
* **Distributed Logging and Task Scheduler:** Discusses the roles these components play in maintaining system health and ensuring tasks are performed timely.
* **Sharded Counters:** Explains their use in managing count-based data in a distributed environment.

### Case Studies

The document also includes design scenarios for various applications to provide practical examples of applying distributed monitoring principles:

* **Design YouTube, Quora, Google Maps, Yelp, Uber, and Twitter:** Each case study discusses specific challenges and design considerations relevant to these platforms.

# 

# 2. Introduction to Distributed Monitoring

### Key Concepts of Distributed Monitoring

* **Importance of Monitoring**: Monitoring helps prevent cascading failures in distributed systems by providing early warnings and identifying the root causes of faults.
* **Scenarios**: An example scenario is given where a failure in one service affects others, demonstrating the interconnectedness and potential points of failure in distributed systems.

### Costs and Implications of Downtime

* **Downtime Costs**: Significant financial losses can occur due to downtime, with examples like Meta's outage in October 2021 costing approximately $13 million per hour, and an AWS outage on December 7, 2021, which cost around $66,240 per minute.
* **Global Infrastructure**: Monitoring must encompass globally distributed data centers, as modern IT infrastructure spans across the world, linked through various networks.

### Types of Monitoring

* **Service-Side Errors**: These are server-level issues that can be directly monitored and are usually indicated by server error responses (HTTP 5xx codes).
* **Client-Side Errors**: Errors that originate from the client side, often invisible to servers unless specific monitoring is in place (e.g., failed connections or HTTP 4xx errors).

### Monitoring Strategies

* **Automated Monitoring**: Due to the vast scale of modern distributed systems and the impracticality of manual monitoring, automated systems are crucial for efficient operation and cost reduction.
* **Examples of Error Monitoring**: The document outlines various potential errors, such as issues in container allocation and UI updates, and discusses strategies for monitoring these.

### Application Design Examples

The document also discusses system designs for various applications as examples of how monitoring integrates into larger systems:

* **Designing Newsfeeds, Instagram, and more**: Each example includes considerations for how monitoring systems would interact with and support the application architecture.

### Educational Approach

* **Interactive Learning**: The document is part of an educational course aimed at teaching system design through interactive examples and real-world scenarios.

# 3. Prerequisites of a Monitoring System

### Monitoring: Metrics and Alerting

* **Importance of Metrics**: The document emphasizes that a robust monitoring system must define clear metrics (what to measure and in what units) and establish threshold values. Metrics should give insights into the system’s health at any point in time, helping the support team respond swiftly and accurately to issues.
* **Examples of Metrics**: Metrics might include the number of requests a server handles per second, network throughput, and latency measurements. Metrics are collected with minimal impact on performance, using indicators like user-perceived latency or computational resource usage.

### Populating and Persisting Metrics

* **Data Collection**: Metrics can either be pushed by servers or pulled by the monitoring system based on configured intervals. This design choice affects how data overloads are managed and how firewalls are navigated.
* **Data Storage**: The document discusses storing metrics data centrally, possibly in memory for smaller setups or using time-series databases for larger operations. This aids in maintaining a historical view of the metrics, crucial for backtracking issues and understanding system trends over time.

### Application-Specific Metrics

* **Code Instrumentation**: For application-specific monitoring, additional code or APIs may need to be embedded within applications. This is referred to as code instrumentation, which helps in collecting detailed data about application performance and issues.
* **Utility of Application Metrics**: These metrics provide a holistic view of application health and are used for making informed decisions about resource allocation, troubleshooting, and predictive maintenance.

### Alerting Mechanisms

* **Alert Triggers**: The monitoring system must be configured to trigger alerts when metrics exceed or fall below predefined thresholds. This function is critical for proactive problem resolution.
* **Alert Actions**: Actions may include notifications to the support team, automated system adjustments, or triggering more detailed diagnostics.

### Interactive Elements

* **Terminal Commands**: The document includes interactive terminal commands like top in Linux to view real-time CPU usage and system processes, demonstrating practical ways to engage with the monitoring system.

### Strategic Questions

* **Failure Handling**: It poses strategic questions regarding conventional approaches to IT failures, promoting a deeper understanding of system resilience and the role of monitoring in minimizing disruptions.

### Practical Applications

* The prerequisites outlined are applicable to various system designs such as those for social media platforms, URL shorteners, and more, illustrating the broad utility of effective monitoring systems.