*Performance is part is the non-functional requirements of a software, it can be defined as* ***how efficiently a software can accomplish its tasks****. When we talk about performance as part of the software engineering process, we are talking about a* ***set of methods*** *which are applied with the goal of getting guarantees that the* ***system and infrastructure are stable, scalable, resilient, and reliable*** *to give customers a good experience when using your application, using the* ***least amount of your computational resources*** *possible.*

*Performance Optimization of a programs and software is the process modifying a software system to make it work* ***more efficiently and execute more rapidly****. Performance optimization is key in having an efficiently functional application and is done by* ***monitoring and analysing the performance of an application and identifying ways to improve it****. Performance optimization focuses on improving just one or two aspects of the system’s performance, e.g* ***execution time, memory usage, disk space, bandwidth etc.*** *This will usually require a trade-off where one aspect is implemented at the expense of others. For example, increasing the size of cache improves run-time performance, but also increases the memory consumption.*

*common problems that affect application performance –*

1. *Poor database design and optimization: If the database is not properly designed, it can lead to slow performance and a lack of scalability. How to avoid Poor database design and optimization*

* *Normalize data.*
* *Create indexes.*
* *Use suitable data types.*
* *Use stored procedures -Stored procedures are pre-compiled SQL statements that can be called and executed by an application.*
* *Monitor performance.*
* *Optimize Queries.*
* *Test and tune the database.*

1. *Network latency and bandwidth issues: Applications that rely on network communication can be affected by latency and bandwidth issues. Network congestion, routing problems, and limited bandwidth can lead to slow performance. How to avoid network latency and bandwidth issues*

* *Monitor network performance - monitoring is important to identify and troubleshoot any issues. This can be done using tools such as network monitoring software, network analysers, or simple ping and traceroute commands.*
* *Optimize routing: Optimizing routing can be done using routing protocols such as OSPF or BGP or load balancers to distribute traffic across multiple routes.*
* *Minimize the use of remote connections: This can be done using local storage, caching, data replication, or mirroring techniques.*
* *Use Quality of Service (QoS): Quality of Service (QoS) is a set of technologies and techniques used to manage network bandwidth and ensure critical applications and services receive the bandwidth they need.*
* *Use Content Delivery Networks (CDNs): CDNs are networks of servers that are distributed around the world. They are used to distribute content to users based on their geographic location. This can help to reduce network latency and improve the performance.*
* *Use compression: Use compression techniques to compress the data sent over the network. This can help to reduce the amount of data that needs to be sent, which can help to reduce network latency.*
* *Use network segmentation: Use network segmentation to separate several types of network traffic and apply different QoS policies to them.*

1. *Memory leaks and resource constraints: Memory leaks occur when an application consumes memory even after it’s no longer needed. Resource constraints occur when an application cannot access the resources it needs to function correctly.*

*How to avoid -*

* *Use memory management tools - profilers, memory leak detectors, garbage collectors.*
* *Monitor resource usage: Regularly monitoring memory, CPU, and disk space can help to identify and troubleshoot any issues.*
* *Use caching.*
* *Optimize Algorithms - This can be done by using data structures that are more efficient.*
* *Use memory-efficient data structures: Use data structures that are memory-efficient.*
* *Release resources when they’re no longer needed: Release resources such as memory, file handles, and database connections.*
* *Use memory pools: Memory pools are pre-allocated blocks of memory that are used to reduce the overhead of allocating and deallocating memory.*

1. *Inefficient algorithms and code: Applications that use weak algorithms or code can run slowly. This can be caused by issues such as poor data structures, unnecessary computations, and a lack of code optimization.*

*How to avoid –*

* *Use efficient algorithms: Use efficient algorithms that have a good time and space complexity.*
* *Use data structures appropriate for the task.*
* *Profile and optimize code: Using a code profiling tool, such as FusionReactor, can rapidly speed up issue identification and help optimize performance bottlenecks in the code.*
* *Minimize the use of unnecessary computations: Minimize the use of extreme analyses.*
* *Avoid using global variables.*
* *Use design patterns.*
* *Use code reviews.*

1. *Lack of caching and overuse of resources: Caching is a technique that stores frequently used data in a location faster to access than the original storage location. Applications that lack caching or overuse resources can run slowly.*

*How to avoid –*

* *Use caching.*
* *Use caching libraries or frameworks.*
* *Use Content Delivery Networks (CDNs).*
* *Monitor resource usage.*
* *Optimize algorithms.*
* *Use memory-efficient data structures.*
* *Release resources when they’re no longer needed.*

1. *Overloaded servers and limited scalability: Applications that run on overloaded servers can run slowly. This can be caused by issues such as a lack of resources, a lack of failover mechanisms, and limited scalability.*

*How to avoid –*

* *Monitor server performance.*
* *Use load balancers.*
* *Use auto-scaling.*
* *Use cloud computing: Cloud computing can help improve an application’s scalability by allowing you to quickly add or remove servers as needed.*
* *Optimize resource usage: Optimizing resources such as memory, CPU, and disk space can help reduce the server load.*
* *Use a microservices architecture.*
* *Use containerization: Containerization allows you to package an application and its dependencies into a container.*

1. *Incorrect configuration settings: Applications that are misconfigured can run slowly. This can be caused by issues such as incorrect settings for memory, CPU, and other resources.*

*How to avoid –*

* *Document configuration settings - Document all the configuration settings including the default values and any recommended or required settings.*
* *Use configuration management tools: Use tools such as Ansible, Puppet, or Chef to automate the configuration of your servers and applications.*
* *Use environment variables.*
* *Use version control.*
* *Test the configuration.*
* *Monitor the system.*
* *Keep the system updated: Keep the system updated with the latest security patches and software updates.*

1. *Integration issues with external systems: Applications that rely on external systems can be affected by integration issues. This can be caused by poor API design, lack of standards, and documentation.*

*How to avoid –*

* *Understand the external system: Understand how the external system works, including its data structures, APIs, and any limitations or constraints that it may have. This will help to ensure that the integration is done correctly and that any issues can be identified and resolved quickly.*
* *Use well-defined interfaces: Use well-defined interfaces when integrating with external systems. This will make it easier to understand how the systems are communicating and will make it easier to troubleshoot.*
* *Test the integration: Test the integration between the systems to ensure that it is working as expected.*
* *Use logging and monitoring: Use logging and monitoring to track the systems’ interactions and identify any issues.*
* *Use a buffer: Use a buffer when integrating with external systems. This can help to reduce the load on the external system.*
* *Use a message queue: Use a message queue when integrating with external systems.*
* *Use a hybrid approach: Use a hybrid approach to integration.*

1. *Concurrency and synchronization problems: Applications that use multiple threads can run slowly due to concurrency and synchronization problems. This can be caused by issues such as race conditions, deadlocks, and poor synchronization design.*

*How to avoid –*

* *Understand the problem: Understand the problem you are trying to solve and the specific requirements for concurrency and synchronization. This will help to ensure that the correct solution is used.*
* *Use synchronization objects: Use synchronization objects such as semaphores, mutexes, and monitors to synchronize access to shared resources.*
* *Avoid using global variables.*
* *Use thread-safe data structures.*
* *Use lock-free data structures.*
* *Use atomic operations: Use atomic operations, such as compare-and-swap, to avoid issues such as race conditions and deadlocks.*