<https://www.youtube.com/watch?v=jdR9kT2xkzw&index=5&list=PL5BFBFE156D50B07E>

**FrontEnd**

1. save image as compressed. to load the page fast.
2. Jobs like microservice to compress images.

**Performance Measures**

1. Startup Time
2. Response Time
3. Throughput🡪 For how many request can a system respond back for a given amount of time.
4. ***Scalability*** is the capability of a system to handle a growing amount of work,

**How will you define performance?**

1. The User hits a web-page, how much time the server is taking time to respond.
   1. JSP to servlet conversion takes time for the first user. Load at startup.
2. After serving 1000 users or so, the server response time slows down.
   1. Because of open connection left out on JMS/DB/Files/Network calls etc…
   2. Heap Size full, Deadlock occurred.

**How to improve performance?**

1. User Profiler in VisualVM to analyse the application work load.
2. Use VisualVM to analyse thread dump snapshot, heap dump snapshot, memory profiling, CPU profiling, out of memory issue, deadlocks, Hard Disc Usage etc.
3. Logging can be done asynchronously to save precious time of a method call.
4. Improve upon the concurrency of software piece. Use all the cores of CPU
   1. Use Atomic utilities provided in java. util.concurrent.atomic package, or Fork & Join to achieve higher throughput in concurrent applications.
   2. Use NIO wherever possible. Ex: Socket Connections, WebService Calls. Hystrix is one such example
   3. Use Transaction less DB like Mongo for storing reports like data structure.
   4. We should try holding the shared locks for as little time as possible.
   5. Eliminate Synchronized variables...Ex: Use arrayList instead of Vector.
5. Picking up appropriate algorithm and data structure for a given scenario can help optimize the processing Ex: Choosing ArrayList over LinkedList
6. Use right design pattern like “fly weight”, “Transfer Object”.
7. Use Caching in scenario where data is stale.
8. Choose Hibernate on Large project with more than 100 tables. Else choose plain JDBC or Spring JDBC.
9. Consider using Weak Reference Object which can increase memory on demand
10. If we are using SQL in our application then we should tune the SQL, create indexes on the essentials table columns for faster retrievals.
11. We should tune our JVM for optimum memory settings (Heap, PermGen, etc) and Garbage collection settings.
    1. For example if we do lot of text processing in our application with big temporary objects being created, then we should have larger Young Generation defined so that frequent gc run does not happen.
    2. Selecting the appropriate Garbage Collector like G1 or CMS
12. avoid catching throwable --> catch appropriate exceptions
13. start your transaction from facade. Rather from controller, service or DAO layer
14. compare like null==a and not like a==null. "null==a" will make the compiler to assign some undefined value to the program at compile time. So this increases the speed of programming
15. Dont Release non-memory resources like Connection Object,Socket inside finalize() method of object. Because it is not guaranteed when the method will be called.
16. Avoid the Overuse of Exceptions. Exception handling is costlier operation.
17. Keep up to date with new technologies for performance benefits.
18. Design level: The architectural design of a system affects the overall performance of the system. Specify efficient algorithms to carry out the task. Specify efficient design patterns like below.
    1. Algorithm Strategy Pattern.
    2. Execution Pattern
    3. Implementation Strategy Pattern like program organization and data structures to use.
    4. Structural Design Patterns which explains the structure of application being developed.
       1. Creational Pattern
       2. Structural Pattern
       3. Behavioral Pattern
19. Source Code level:
    1. Use VisualVM to test the performance of system, for analysing the threads, memory and cpu.
    2. Use Java Coding java.util.concurrent, caching,
    3. From database tune sql, apply index,
    4. Tools like PMD, CheckStyle can be used to improve the efficiency of code.
20. Assembly level: Use Native Coding (JNI) in case of hardware level interaction is needed, which will boost the speed of program.