Mindtree
A Larsen & Toubro Group Company

Executing Performance **Assessment Exercise**

Introduction

Application performance is a crucial success factor for the program. The overall performance impacts the end user experience. We execute performance assessment exercise to understand the overall performance of the application. The performance assessment exercise involves analyzing the performance bottlenecks at various layers and at various solution components. In this whitepaper, we discuss the details of performance assessment exercise.

Welcome to possible

A Mindtree Whitepaper

Layer-wise Performance Assessment

As the overall performance is dependent on performance at each of the layers, it is imperative that we assess the performance at various layers.

Web Layer

We measure and monitor the performance metrics for the web components such as web pages, views, and UI libraries. We can use the below given checklist to assess the web components:

Are JS and CSS files merged and minified?	Are images optimized using CSS sprites or impage compression methods?	Do we have too many chatty calls between web and the server?	Is the application using the browser cache to store the frequently used web components?	Are the JavaScripts files placed at bottom of the page and CSS files placed at the top of the page?	Is lazy loading used for non-critical page resources?

Web Performance Principles

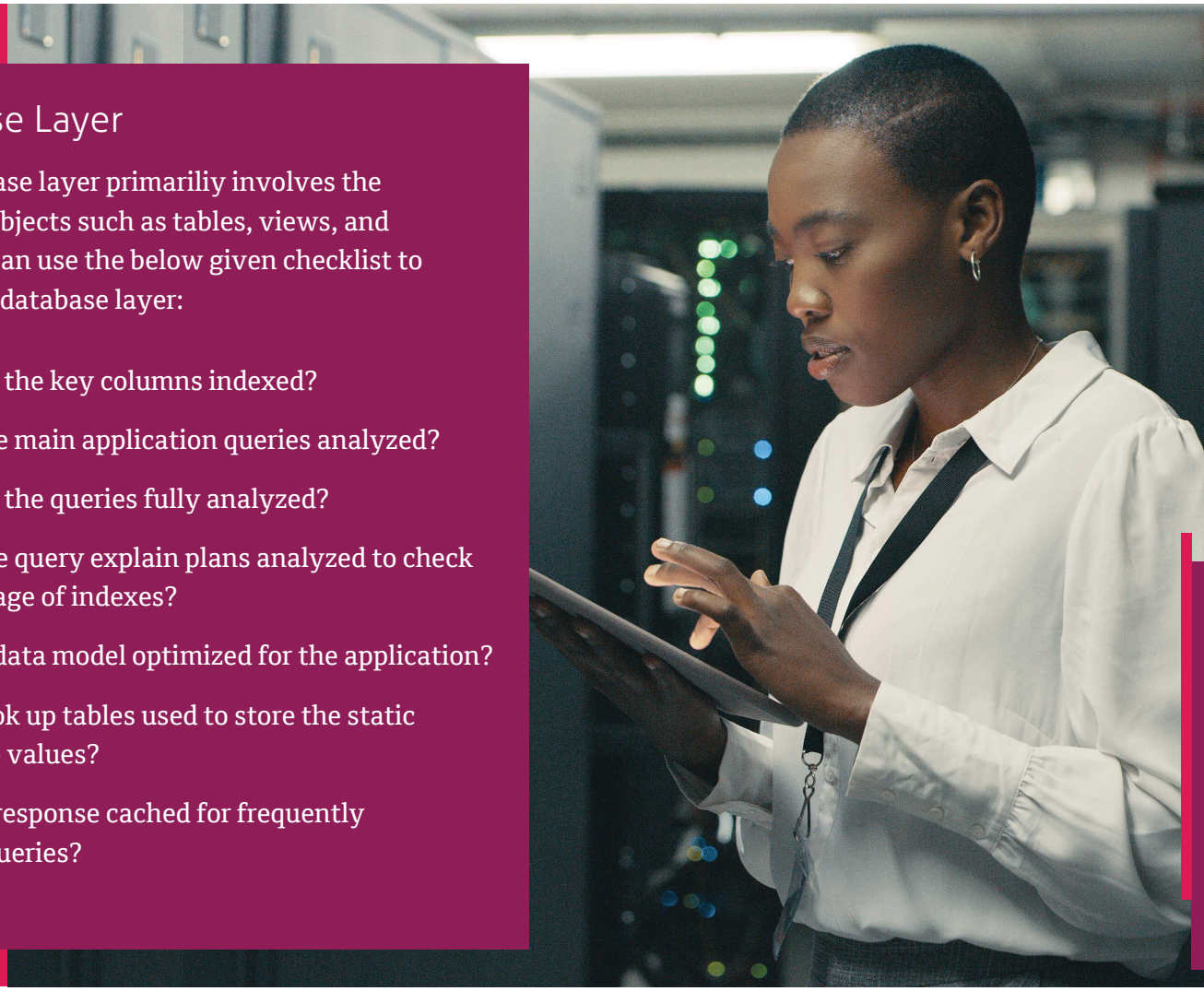
Check if the below given web performance principles are adhered to in the application:

1. Simple and lightweight: Keep the frequently used pages like home page and landing page simple yet effective. This would involve-
    a. Include only key functionalities to keep it light weight
    b. Have optimized marquee images
    c. Provide search feature to reach any page using keyword search
    d. Provide elaborate menu to allow the user to navigate to any sub level page.
2. Fine tune the key functionalities for performance.
3. Maximize client side components: Wherever possible employ partial page rendering to avoid the full page refresh.
4. Avoid 3rd party plugins unless absolutely required. Even when they are included, only load the scripts on-demand and keep the 3rd party scripts at the bottom of the page.
5. Think caching: Apply caching at all possible layers to get optimum performance. Caching can be applied at web server, server side, database layer and other possible integration layers
6. Performance guidelines: Come up with organization level performance guidelines involving images, JS/CSS coding and other aspects. Some of them could include-
    a. Always sse PNG format of image
    b. Use CSS sprites
    c. JS code validation with JSLint
    d. Use lazy loading of content wherever possible
    e. Avoid iframes and redirects to the best extent possible

Application Layer

The application layer mainly involves the server-side code that exposes the services for the web components. We can use the below given checklist to assess the application layer:

- Is the code reviewed from the performance standpoint?
- Are the duplicate code blocks removed?
- Is the server configured with appropriate memory and storage?
- Is layer-wise caching implemented?
- Is the API call tracing enabled to understand the end to end performance metrics?
- Is the application performance monitoring enabled?
- Is the error monitoring enabled?



Database Layer

The database layer primariliy involves the database objects such as tables, views, and such. We can use the below given checklist to assess the database layer:

1. Are all the key columns indexed?
2. Are the main application queries analyzed?
3. Are all the queries fully analyzed?
4. Are the query explain plans analyzed to check the usage of indexes?
5. Is the data model optimized for the application?
6. Are look up tables used to store the static lookup values?
7. Is the response cached for frequently used queries?

Performance Metrics

Given below are the key performance metrics which we can analyze for all the involved layers:

- System availability that depicts the overall availability of the system
- Response times include the metrics such as Time To First Byte (TTFB) and Average Response Time (ART) for the web pages. The response times are measured at average user load and at peak user load.
- Resource utilization provides the average utilization of resources such as CPU, memory, disk, and the bandwidth.

We also check the below given parameters at the server level to ensure that we have configured the optimal values to handle the peak user load:

- Heap size indicates the memory available to the application objects.
- Connection pool settings provide the details such as minimum pool size, maximum pool size and such
- Thread parameters indicate the thread-related values such as timeouts, maximum allowed thread and such
- Session parameters indicate the session related parameters such as session timeout values

Performance Aiding Tools

As part of performance assessment, we can identity the existing performance aiding tools and recommend the required tools. We have given the main performance aiding tools in table 1.

Table 1: Performance Aiding Tools

Tool Category	Open Source / Commercial Tool(s)
Web page analysis tools (HTML analysis, performance benchmarking, improvement guidelines)	Yahoo YSlow, Google PageSpeed, HTTPWatch, Dynatrace AJAX Edition Google Chrome Lighthouse
	Google PageSpeed Insights
Page development tools (analysis of page load times, asset size, asset load times and such)	Firebug, Google Chrome Developer toolbar, Fiddler, HTTP Archive WEB PAGEiddle, CSSLint, JSLint, W3 CSS Validator, W3 HTML validator
Asset merging and minification tools (JS/CSS minification)	Yahoo UI (YUI) minifier, JSMini, JSCompress

Page performance testing tools (load simulation)	JMeter, LoadUI, Grinder, Selenium
Synthetic monitoring (transactions simulation and performance statistics)	Web Page test, DynaTrace Synthetic monitoring
CDN	Akamai, CloudFlare, KeyCDN,
Web analytics (track user behavior, performance reporting)	Google Web Analytics, Omniture, Piwik
CSS optimization tools	CSS Sprites, SpriteMe, SpritePad
Bottleneck analysis (dependency and bottleneck analysis)	WebProphet, WProf
Real User Monitoring (RUM) (monitoring and bottleneck analysis)	New Relic, Dynatrace, Gomez
Network analysis (network traffic, HTTP headers, request/responses, protocol analysis)	Wireshark, Charles Proxy
Application performance monitoring (APM) (Layer wise monitoring of application code)	New Relic, AppDynamics Dyna Trace Monitoring, Nagios
Error monitoring	Splunk, DataDog
Healthcheck monitoring	Prometheus, ELK (Elastic Search, Logstash, Kibana), Kubernetes pod monitoring
Mobile app testing	Appium, UI Automator
Application profiling	Open Source: JProbe, Eclipse Profiler Commercial: Jprofiler, OptimizeIt,
Load testing	Open Source: Apache JMeter, Grinder, Apache Bench, HTTPPerf Commercial: HP LoadRunner, NeoLoad, BlazeMeter
Service testing	LoadUI, SOAPUI

Performance Assessment Report

Based on the detailed performance assessment, we can provide the assessment report. In this section, we have detailed the common recommendations for each of the layers.

Recommended Next Steps for Web Performance Optimization

In table 2, we have provided the main problem patterns and performance optimizations that can be done on the presentation layer.

Table 2: Web Performance Optimizations

Problem Pattern	Brief Description	Suggested Remediation Step
Numerous JS and CSS files	Each of the pages has on an average 20+ JS files, 18+ CSS files which are not minified	Merge and minify CSS and JS files to create minimal number of files
Numerous images	Many pages have 50+ images	• Reduce image number through CSS Sprite • Compress images • Lazily load the images
Positioning of JS and CSS files	All JavaScript files are placed on top of the page which blocks the page load	Place CSS at top and JS at bottom to optimize perceived page load time
Non caching of static assets	In almost all pages, the caching headers is not set for static assets (images, JS, CSS files). On an average we found 10+ images have cache headers missing	Based on the update frequency, we need to set the cache headers for static assets for optimal performance.
Very high Time to First Byte (TTF)	Very high TTF indicates higher server response time	Need deep-dive analysis of server side factors
Large page size	Static assets contribute to high page size	Merging and minification along with CSS Sprite should address this issue. Enable HTML compression at web server level
Unused CSS files	On an average 3+ CSS pages per page are not used	Remove unused CSS files

Recommended Next Steps for Server-Side Performance Optimization (Services And Database)

In table 3 we have provided the main problem patterns and performance optimizations that can be done on server side (for database and services).

Table 3: Server side Performance Optimizations

Problem Pattern	Brief Description	Suggested Remediation Step
Numerous database queries for each page	We noticed that sometimes one page results in execution of numerous queries	Minimize number of db calls. One DB call per page is the best case scenario Move all DB logic into database stored procedure so that database engine can do the heavy lifting Fine tune all slow performing queries. Use the batch/bulk query call supported by hibernate
Database query in loop	In some scenarios query is invoked within a loop	We should completely avoid database calls in a loop. To avoid this, we can either create an oracle stored procedure to do all the database heavy lifting operations and invoke the stored procedure only once from the application passing all needed parameters.
Absence of caching	In few pages the common data is reused and we make duplicate database calls to get common data (such as user profile data)	Leverage coherence cache or object cache to cache commonly used data and lookup values.
Fine tune IIB calls	The IIB calls are taking lot of time	IIB calls need to be optimized.
Avoid WSRP calls	In couple of pages lfn document is slow due to WSRP calls	Avoid WSRP calls and fetch data asynchronously or using light weight REST calls/microservice calls

Conclusion

I have tried to cover all the major parameters to keep in mind for performance assessment exercise. Hope this helps to optimize your web development efforts. Feel free to reach out to me at Shaileshkumar.Shivakumarasetty@mindtree.com for any queries.

References

1. https://medium.com/globalluxsoft/5-popular-software-development-models-with-their-pros-and-cons-12a486b569dc

Dr. Shailesh Kumar Shivakumar

Solution Architect

Dr. Shailesh Kumar Shivakumar has 19+ years of experience in a wide spectrum of digital technologies including, enterprise portals, content management systems, lean portals, and microservices. He holds a Ph.D. degree in computer science and has authored eight technical books published by the world's top academic publishers such as Elsevier Science, Taylor and Francis, Wiley/IEEE Press, and Apress. Dr. Shailesh has authored more than 14 technical white papers, five blogs, twelve textbook chapters for various undergraduate and post-graduate programs and has contributed multiple articles. He has published 20+ research papers in reputed international journals. Dr. Shailesh holds two granted US patents, apart from ten patent applications. Dr. Shailesh has presented multiple research papers at international conferences. Dr. Shailesh's Google Knowledge Graph can be accessed at https://g.co/kgs/4YoaiN . He has successfully led several large scale digital engagements for Fortune 500 clients. Shailesh can be reached at Shaileshkumar.Shivakumarasetty@mindtree.com