Going for Speed: Full-Stack Performance Engineering in Modern Web-Based Applications

Organizers:
- Wolfram Wingerath† (wolfram.wingerath@baqend.com)
- Benjamin Wollmer†§ (benjamin.wollmer@baqend.com)
- Felix Gessert† (felix.gessert@baqend.com)
- Stephan Succo† (stephan.succo@baqend.com)
- Norbert Ritter§ (norbert.ritter@informatik.uni-hamburg.de)

† Baqend, Hamburg, Germany (https://baqend.com)
§ University of Hamburg (Databases and Information Systems, DBIS), Hamburg, Germany (https://vsis-www.informatik.uni-hamburg.de)

Introduction
Our tutorial addresses the 4 primary challenges of developing fast Web applications (cf. Figure 1).

First, **backend processing** is mandatory for many client requests as content today is usually tailored to individual users. Rendering a website therefore does not only require fetching generic resources like images and stylesheets, but also accessing databases for personalized information such as recently viewed items or product recommendations. As the second challenge, **network delays** contribute to the overall time it takes to deliver content from the Web server to the end user's device. Content delivery networks (CDNs) are often used to retrieve data from closeby edge nodes instead of the faraway origin server. However, the dynamic nature of personalized websites prohibits traditional Web caching and thus impedes fast content delivery when server and client are physically separated. The third challenge is **client rendering**, which includes executing scripts on the user device and creating visuals in the viewport. Optimizing rendering efficiency is difficult, because it requires minimizing the number and size of critical resources and requesting them in the order in which the browser processes them. As a fourth challenge, **performance tracking & analysis** on different levels of the application stack is required to monitor the user experience and correlate it with business key performance indicators (KPIs). But since tracking introduces an overhead by itself, measuring performance without introducing additional delays can be just as challenging as improving it.
**Structure & Overview**

The tutorial is designed as a half-day lecturing style presentation and is structured into 4 parts:

1. **Efficient Frontend Design:** The key principle of frontend optimization is to make the browser’s work as easy as possible. In this part, we will therefore explain what exactly happens when the browser loads a website and how this process can be accelerated. We will discuss several best practices for optimizing the critical rendering path (CRP), minimizing page weight, server-side image transcoding, and for getting the most out of browser or CDN caching. A particular focus will be on the relatively new Service Worker standard and the myriad of possibilities it adds to Web application development, such as add-to-homescreen functionality, the offline mode, push notifications, and different caching strategies (“Progressive Web Apps”). In doing so, we will provide background information on scopes, lifecycles, persistence, and other pivotal Service Worker concepts, as well as exciting features and APIs that are yet to come (e.g. for payment, sharing, and speech processing).

2. **High-Performance Networking:** This part starts with the basics of how the browser interacts with the network and how it handles requests. We will then deep-dive into the Internet protocol stack and discuss performance optimizations for TCP (e.g. fast-open, congestion and flow control), TLS (e.g. OCSP stapling, dynamic record sizing), CDNs (e.g. early TLS termination, warm backend connections), HTTP (REST, Web caching), and compression (e.g. Gzip, Brotli). Finally, we will dissect the HTTP/2 standard and its most significant improvements over its predecessor (e.g. multiplexing, server push, resource prioritization, and header compression). To provide a comprehensive wrap-up, we close with considerations on how to upgrade from HTTP/1.1 to HTTP/2, which performance recommendations are still valid, and which best practices have become anti-patterns.

3. **Scalable Backend Architectures:** For ideal performance, the backend has to produce a response for every incoming request at all times and despite a large number of concurrent users and possible network outages, machine failures, or other error scenarios. This part of the tutorial therefore provides a broad overview over data management and processing options in the backend. In more detail, we recall the basics of distributed data management (e.g. partitioning, replication, eventual consistency, NoSQL data models) and present a brief discussion of significant impossibility results (CAP and PACE/LC theorems). We then conduct an in-depth survey of the NoSQL landscape by discussing individual systems and their architectures, classifying them according to functional and non-functional properties in a framework we call the NoSQL Toolbox. Acknowledging Web applications with real-time requirements, we also cover systems specialized for push-based data access: After a short historical recap of push-based mechanisms in data management, we dissect popular real-time databases and stream processing frameworks w.r.t. their capabilities in storing, querying, and analyzing data with low latency.

4. **Performance Tracking & Analysis:** In this part of the tutorial, we explore the relationship between technical website performance and online business success. First, we focus on user expectations to examine when they are most likely to engage or buy. We then assume the business perspective and explore the correlation between technical performance measures such as the Google Web Vitals and pivotal business KPIs (e.g. conversion rate or revenue). Having thus established the business impact of Web performance, we turn to the challenge of measuring it correctly. To this end, we highlight the differences between technical and user-centric metrics and then compare traditional synthetic performance tests, tracking, and log analysis with more advanced approaches like CRUX data analysis and real-user monitoring. We conclude with findings from our ongoing study on how Web performance gains drive up user engagement and business success.
Literature