Lecture 9

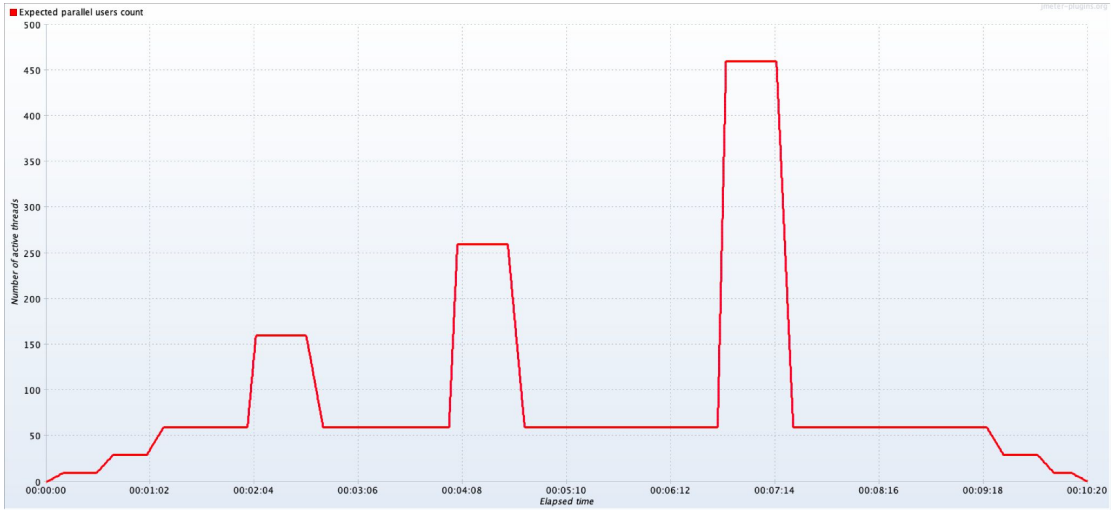
Websockets, Benchmarking / Stress testing,
State management on the Frontend

Websockets

- Similar to sockets, except on the Web
- Allows two way, real-time communication between the client and the server (no polling)
  - REST is one way: the server cannot send data to the client unprompted
- Ideal for things that need faster updates or bidirectional communications:
  - Chats
  - Financial apps, especially trading
- On the backend: most frameworks should support it
  - Django: https://channels.readthedocs.io/en/stable/introduction.html
  - Spring: https://www.baeldung.com/websockets-spring
- On the frontend:
  - React: https://www.npmjs.com/package/react-use-websocket
  - Angular: https://towardsdatascience.com/build-a-websocket-application-with-fastapi-and-angular-988157dce554

Benchmarking / stress testing

- It’s important to know how many users our app can serve before becoming slow
- We will use JMeter with The Ultimate Thread Group plugin for this
- We will set up spike tests like in the following figure:



Benchmarking / stress testing

- See: https://www.perfmatrix.com/jmeter-ultimate-thread-group/
- The example application gets to 100% CPU usage quite fast, with under 50 simultaneous users
- You will have to run similar stress tests to see how many users your application can handle
- Make sure you allow time for the AWS graphs to catch up. You can also use the **uptime** command on the machine itself, or other graphing tools on the machine
- The previous image probably has very high numbers: try to find the max that your setup can handle without reaching 100% CPU usage.
- Try to optimize things as much as you can.

State management on the frontend

- Example scenario:
  - After login, the top app bar should change to display at least your username
  - This should be done without refreshing the page
  - The app bar needs a way to respond to User Service events
- We can achieve this in multiple ways:
  - For Angular:
    - Using Subject / BehaviorSubject and subscribing to it from the App Bar component
  - For React:
    - Using the Context API: https://react.dev/learn/passing-data-deeply-with-context
  - Using Redux, but it's overkill unless the app is huge
- SPAs should never refresh the page to achieve something