Implementing an HTML5 mobile simulation utilizing web workers

Submitted on 24/11/2020

authored by
Lari Alakukku (528362), Miika Rouvinen (356770), Ilkka Malassu (430463)

***Index Terms*—HTML5, web, workers**

***Abstract*—Abstract of the paper here**

I. Introduction

The internet has progressed from a distributed document sharing system to a platform that can host computationally demanding applications. This evolution makes performance a crucial web-client requirement. The modern browser provides a possibility to develop intricate web applications utilizing only HTML, CSS and JavaScript. The developers of a web-based game, for example, need to optimize their code taking the processing capability of different browsers into account. Current hardware mostly focuses on concurrent execution while many web applications still only use the main browser thread. HTML5 introduced web workers as a first step towards concurrent execution in web applications. [1]

Web workers offload processing tasks from the main browser thread to background worker threads, requiring the platform to have basic support for concurrency. The communication between the threads is facilitated with message sending. Web workers can not use DOM operations or access window objects and parent objects. Also, direct data sharing between the main thread and the worker threads is not possible. [1], [2]

II. Related Work

For web-based games, Erbad et al. [1] propose a concurrent processing solution called DOHA, which consists of game event-loops running in worker threads and MultiProc, which is the module for scheduling, state management and other concurrent execution related tasks. Their game implementation had three main components: simulation, graphics rendering and AI. The main thread handles rendering and offloads game event processing to web workers. This communication also requires the sending of state information. In general, DOHA offered better scalability and responsiveness across different platforms. However, thread communication required replicating state across workers, which increased jitter. Zhang et al. [3] introduce WWOF, which is a framework for seamlessly offloading web workers to the cloud. On average, the framework achieved energy savings of 85% on devices such as mobile phones, desktop computers and pads. The performance of the devices improved by a factor of 2-4. Verdú et al. [4] examine how utilizing web workers scales the performance of a JavaScript application. They found that the optimal number of workers depends on various factors such as the CPU architecture and the browser. Using a large number of web workers did not prove to be beneficial compared to using only a few.

III. Implementation

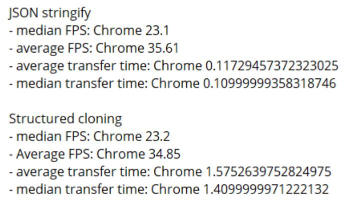
Here we present our implementation.

A. Context

Maybe a subsection if needed.

IV. Evaluation

Here we present our evaluation of the implementation. Example results in Figure 1.



```
JSON stringify
- median FPS: Chrome 23.1
- average FPS: Chrome 35.61
- average transfer time: Chrome 0.11729457372323025
- median transfer time: Chrome 0.10999999358318746

Structured cloning
- median FPS: Chrome 23.2
- Average FPS: Chrome 34.85
- average transfer time: Chrome 1.5752639752824975
- median transfer time: Chrome 1.4099999971222132
```

Fig. 1. This is a figure

V. Conclusions

We conclude our research paper here.

References

[1] A. Erbad, N. Hutchinson, and C. Krasic, “Doha: Scalable real-time web applications through adaptive concurrent execution,” Apr. 2012. DOI: 10.1145/2187836.2187859.

[2] Y. Watanabe, S. Okamoto, M. Kohana, M. Kamada, and T. Yonekura, “A parallelization of interactive animation software with web workers,” in *2013 16th International Conference on Network-Based Information Systems*, 2013, pp. 448–452. DOI: 10.1109/NBiS.2013.74.

[3] J. Zhang, W. Liu, W. Zhao, X. Ma, H. Xu, X. Gong, C. Liu, and H. Yu, “A webpage offloading framework for smart devices,” *Mobile Networks and Applications*, vol. 23, Jan. 2018. DOI: 10.1007/s11036-018-1009-z.

[4] J. Verdú and A. Pajuelo, “Performance scalability analysis of javascript applications with web workers,” *IEEE Computer Architecture Letters*, vol. 15, no. 2, pp. 105–108, 2016. DOI: 10.1109/LCA.2015.2494585.