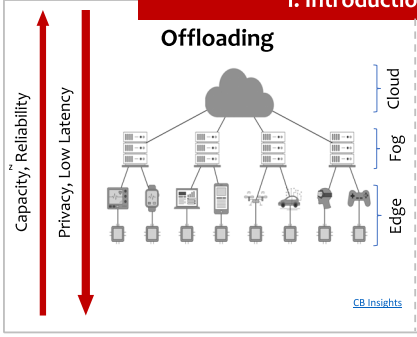
WebAssembly for agile offloading in edge computing

Mohammed Nurul Hoque and Khaled A. Harras

I. Introduction



Offloading

challenges

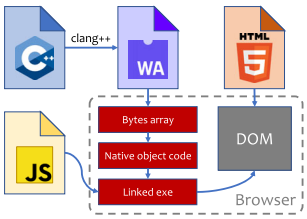
- **Portability**: architecture and OS heterogeneity is characteristic of the edge.
- **Migratability**: because of churn, execution might be interrupted. Migration saves partial work.

II. Portability Approaches

	Speed	Size	Cross OS	Cross Arch	Lang. Ind.	Live Migration
System-oriented						
Native	1.0x	Code	✗	✗	✓	✓
VMs	1.16x[1] [1]	+ ~2GB	✗	✓	✓	✓
Containers	1.02x [1]	+ ~200MB	✗	✗	✓	✓
Language-oriented						
JavaScript	1.88x [2]	Code	✓	✓	✗	✓
Java	1.1-1.9x [4]	Code	✓	✓	✗	✗
WebAssembly	1.1-1.45x [2]	Code	✓	✓	✓	✗

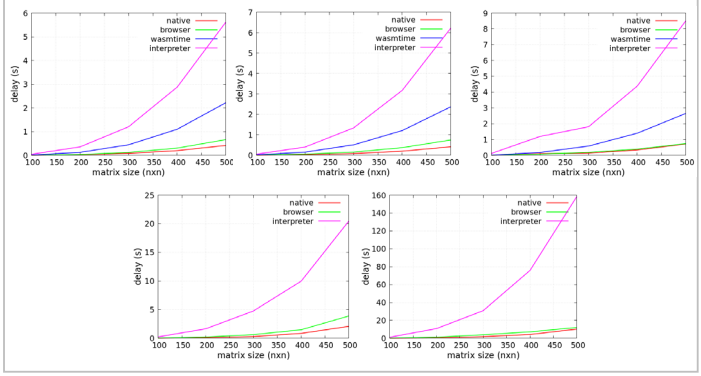
III. What is WebAssembly

- A web standard by major browser vendors, standardized in 2017.
- A low-level stack-based binary format for high-performance code on the web.
- A compilation target for high-level languages like C/C++ Rust and Go.
- Runs inside js runtime in the browser.
- Also possible to run standalone.

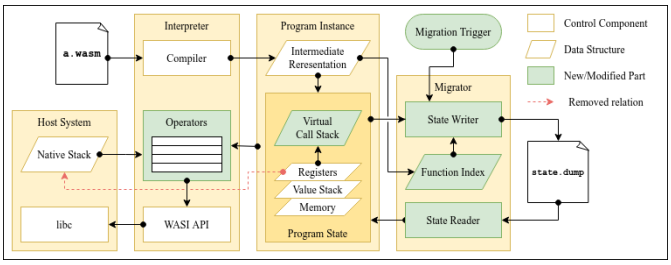


IV. Portability Testing

- We ran some tests to demonstrate the portability of WebAssembly.
- Measured delay for a series of matrix multiplications with increasing size.
- Running native, on browser, using a standalone runtime and using an interpreter.
- Browser is close to native speed.
- Interpreter is slow but very portable.



I. wasm3 Interpreter



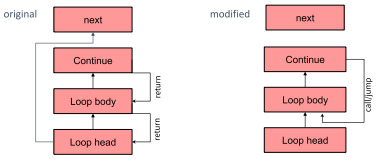
I. Enabling Migration

Steps to enabling migration

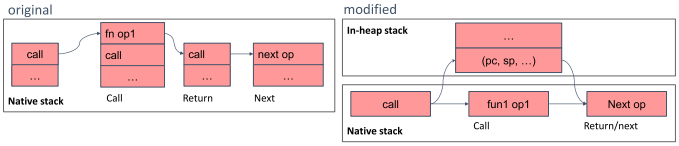
First, we setup the migration infrastructure necessary to migrate sequential infrastructure.
A migration flag is triggered by a timer event.
Each operation checks flag before next operation.

Encapsulation of program state

The program state has to be present in the current native frame.
We moved loop logic from head to continue operation.



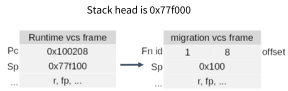
Virtualization of the call stack



Building a function index

Maps addresses to the functions containing them.

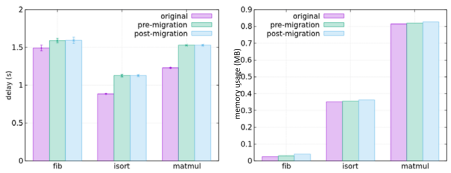
Fn ptr	id
0x100180	3
0x100200	1
0x100274	2



I. Evaluation

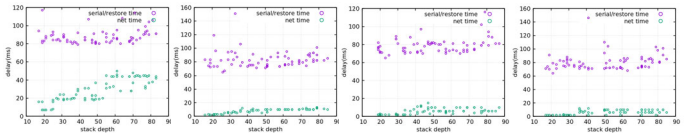
Migratability Overhead

fibonacci, insertion sort, matrix multiplication.
Running on:
1- unmodified interpreter
2- modified without migration
3- modified post-migration

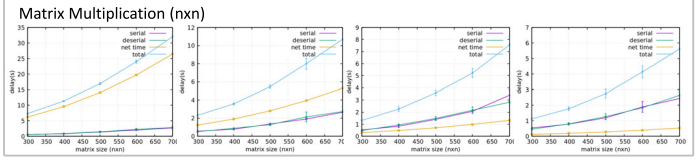


Migration Performance

Fibonacci (n)



Matrix Multiplication (nxn)



References

[1] Jangda, A., Powers, B., Berger, E. D. and Guha, A. (2019) Not So Fast: Analyzing the Performance of WebAssembly vs. Native Code in 2019 USENIX annual technical conference pp. 107-120.
[2] W. Felter, A. Ferreira, R. Rajamony and J. Rubio (2014) An Updated Performance Comparison of Virtual Machinesand Linux Containers. IBM Research Report.
[3] Gascon-Samson, Julien & Jung, Kumseok & Pattabiraman, Karthik. (2018). ThingsMigrate: Platform-Independent Migration of Stateful JavaScript IoT Applications. SEC 2018/
[4L. Gherardi, D. Brugali, and D. Comotti, "A java vs. c++performance evaluation: A 3d modeling benchmark," vol. 7628,Nov. 2012.

MEETING OF THE Minds Undergraduate Research Symposium

Carnegie Mellon University Qatar