Using Thread-Level Speculation to Improve the Performance of JavaScript Execution in Web Applications

Jan Kasper Martinsen, Håkan Grahn, Anders Isberg

jkm@bth.se, hgr@bth.se,

Anders.Isberg@sonyericsson.com

EASE
Embedded Applications Software Engineering

Motivation!

Web Applications are very popular!

- Facebook, Google maps, Twitter, Google mail
- Web Application multimedia:
  - HTML5, WebGL, WebCL
- JavaScript (client side logic) of Web Applications

Motivation! (..more)

- Multicore, everywhere
- Challenging to program!
  - Improve performance?
  - Program correctness?
- JavaScript a sequential language
  - Abstract away from the underlying hardware
  - How to utilize from JavaScript / Web Applications

Web workers

- Message passing
- The complexity remains!
- Increased performance or used to improve the user experience?

Our approach Thread-Level Speculation

- When we encounter a JavaScript function, we execute it as a thread
- If there will be a conflict between threads we need to rollback to an earlier point of time to ensure program correctness
- We can speculate on loops, but previous research has shown that loops are rare in Web Applications

Our implementation

- Modified the Squirrelfish interpreter (in Webkit)
- Nested speculation
- Each time we modify a global variable or an id, we check if writes or reads are in conflict with previous reads and writes
- Each time we speculate on a function, we save the state of execution before that point
- If there will be a conflict, we rollback to the points before the function was speculated (and do not speculate on previous misspeculations)

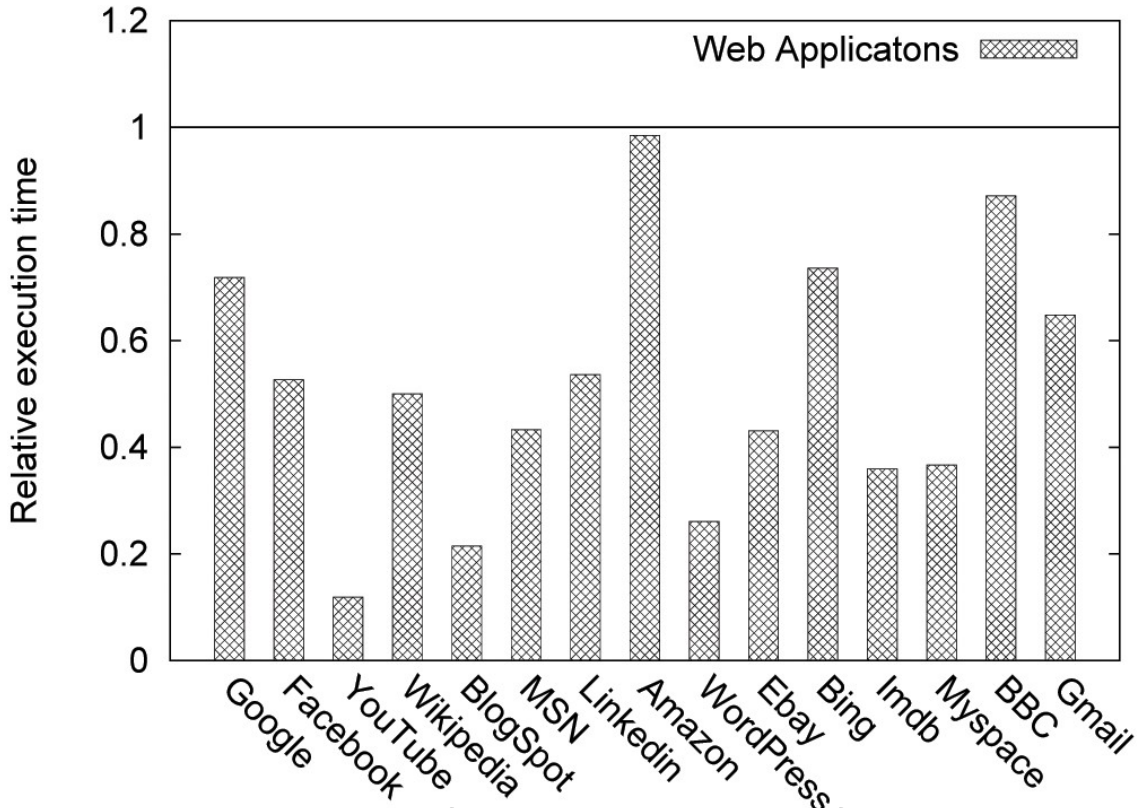
Experimental methodology

We selected 15 Web Application from the Alexa list, and selected them both based on their functionality and popularity

We compare the execution speed with the sequential and thread-level speculation enabled interpreter

Experiments were performed on a dual quad-core Linux computer

Thread-Level Speculation performance



Experiments shows significant speedups for many Web Applications

No. Speculations

- MSN : 12012
- Amazon : 10768
- YouTube : 7349
- Facebook : 968

Function depth (i.e, what is the largest number of nesten function speculations)

- MSN : 24
- Amazon : 23
- YouTube : 13
- Facebook : 22
- Wordpress : 99

Rollbacks

- MSN : 137
- Amazon : 267
- YouTube : 25
- Facebook : 51

Rollbacks / speculations

- MSN : 0.011
- Amazon : 0.025
- Youtube : 0.003
- Facebook : 0.052
- Wikipedia : 0 (no rollbacks!)

Memory usage at each rollback

MSN : 20.1 MB

Amazon : 14.1 MB

YouTube : 17.1 MB

Facebook : 7.1 MB

BBC : 33.0 MB

Average search depth from deletion upon rollbacks

- MSN : 5.85
- Amazon : 8.0
- YouTube : 5.44
- Facebook : 9.16

Max number of active threads:

- MSN : 191
- Amazon : 83
- YouTube : 407
- Facebook : 27

Conclusion

- A large number of threads is required for a high performance
- Rollbacks are potentially less problematic than memory requirements
- Large number of speculation does not mean improved performance
- Some of the concept of JavaScript interesting in Thread-Level Speculation context

Thanks for listening!

My Licentiate seminar 14.12.2011

You are hereby invited!!!

http://www.bth.se/com/jkm