Thread-Level Speculation for Web Applications

Jan Kasper Martinsen and Håkan Grahn
Blekinge Institute of Technology

jkm@bth.se, hgr@bth.se

Two important trends,

Web Applications and multicore computing,

Web Applications,

JavaScript, add some interactivity to web pages …

…however since then, a large number of applications have been developed (gmail, facebook,youtube…)

It has it’s charm...

Multicore computing,

We get more computing power, however there is a catch…

Multicore programming, often regarded as a black art

…We can’t escape it!

Multicore and Web Applications

Multicore → system programmers
Web Application → web designers

A pedagogical challenge?

“Concurrent programming for web designers” anyone?

High level question:

Can we hide the details, and still take advantage of multicore computing?

Thread Level Speculation seems to have some promise…

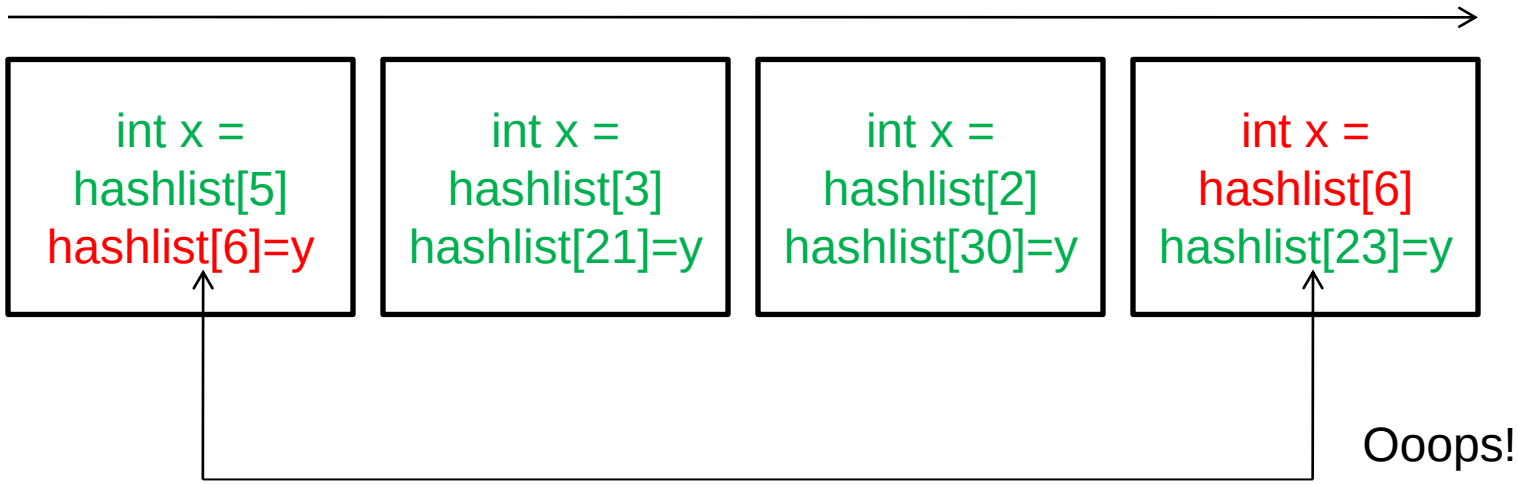
Idea, detect sections(at runtime), that can be parallelized

We know the runtime idea from Just in Time (JIT) compliation…

While the penalty of “wrongly” JIT is execution speed…

…The penalty of TLS is program correctness!!

```
while(1){
        int x = hashlist[key1]
        hashlist[key2] = y
}
```



We have played around with these ideas in a Python based JavaScript interpreter

It seems that TLS have not been tried out in interpreters

Our idea was to use information found in the source code to speculate (i.e., execute by traversion)

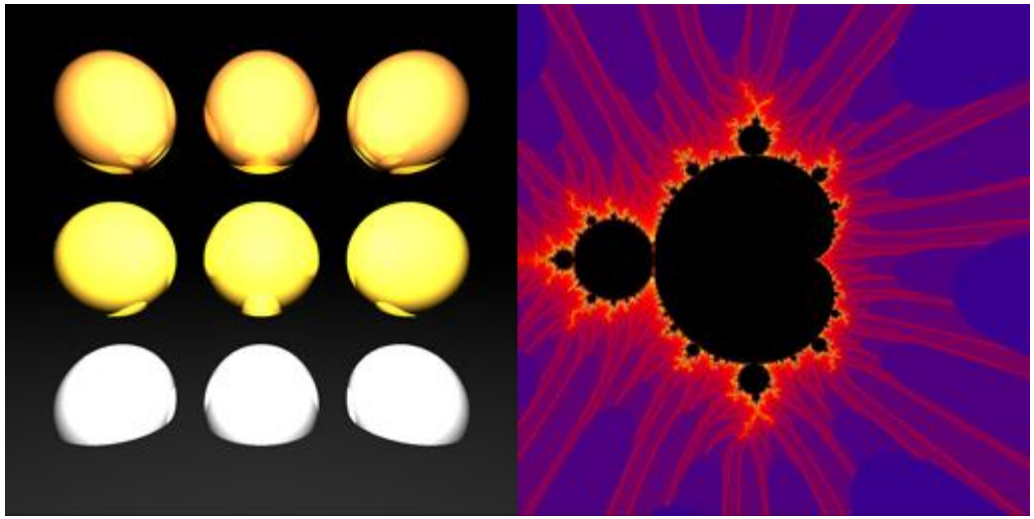
For instance:

```
for(var i=0;i<64;i++){ …
```

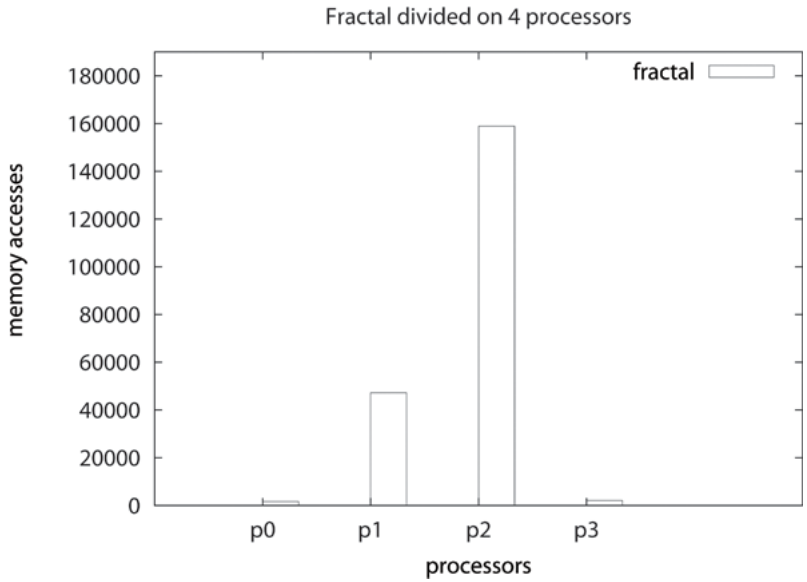
↓

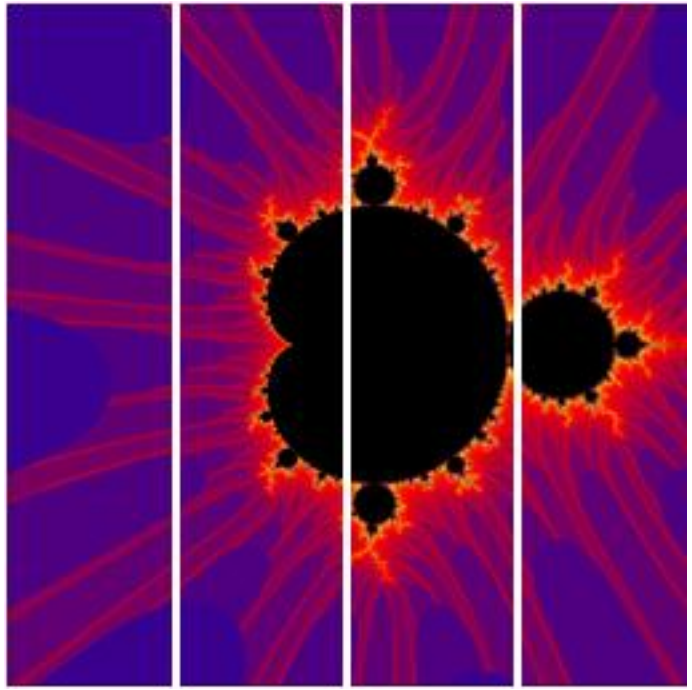
```
for(var i=0;i<16;i++){ …
for(var i0=16;i0<32;i0++){ …
for(var i1=32;i1<48;i1++){ …
for(var i2=48;i2<64;i2++){ …
```

For a selected set of selected benchmarks it has advantages and the results from the simulations are quite good…

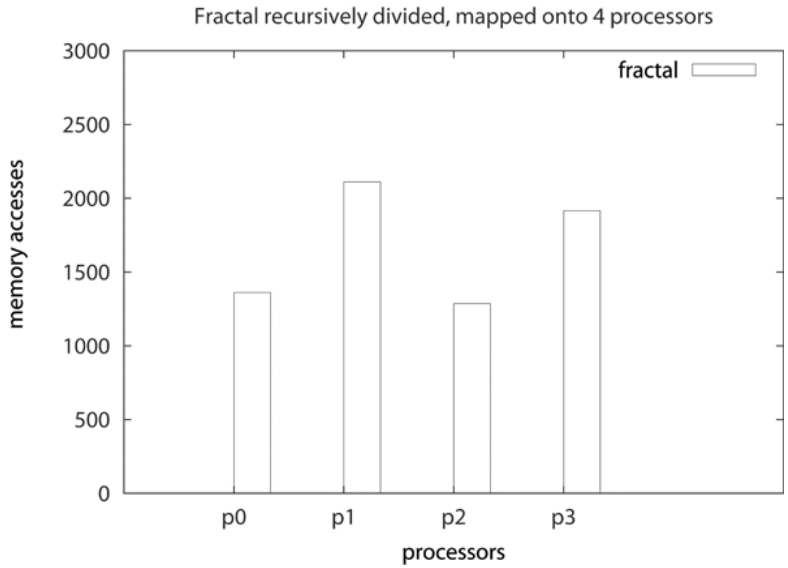


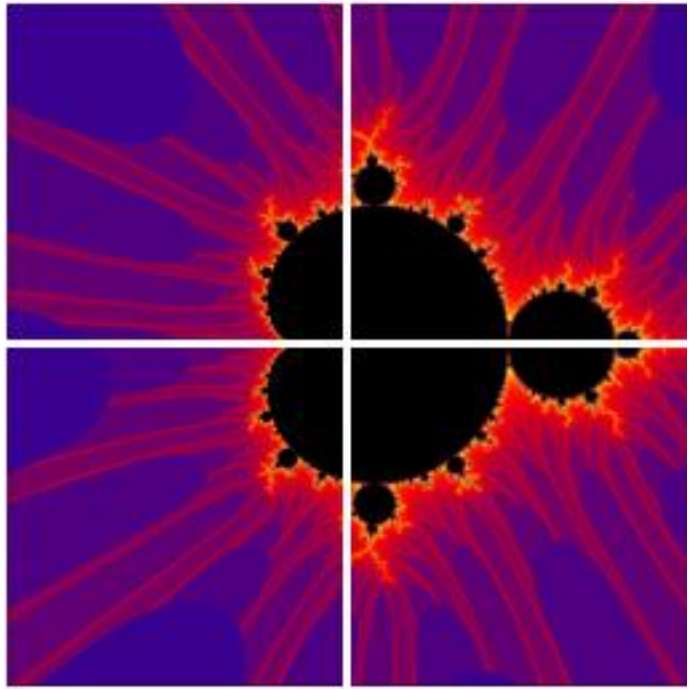
If we divide the first for-loop, we get the following work distribution





Since we have information from the source code, we can go handle nested for-loops …





Our own benchmarks, with simulated concurrency disregards a series (important) factors…

We have started to look into integrating these ideas into TraceMonkey

Perhaps some mid-level between source code and byte code is needed?

What about the benchmarks?