

Towards Hybrid Cooperative-Preemptive Scheduling

Yizheng Xie

yizheng_xie@brown.edu

Di Jin

di_jin@brown.edu

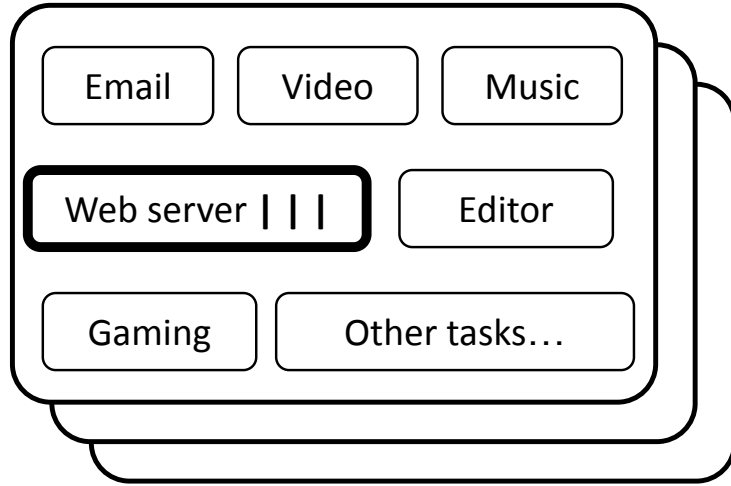
Nikos Vasilakis

nikos@vasilak.is



Brown University

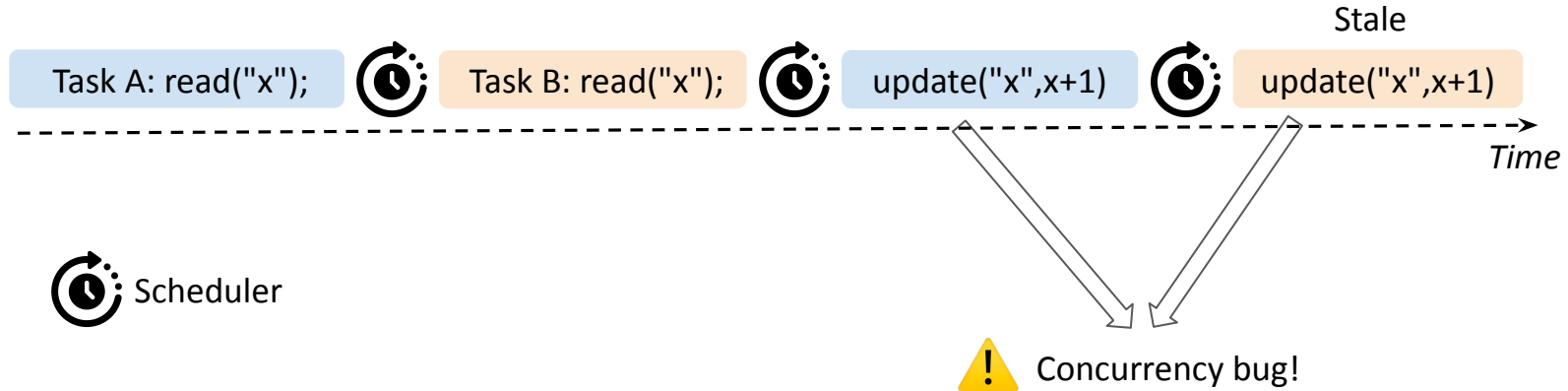
Multitasking



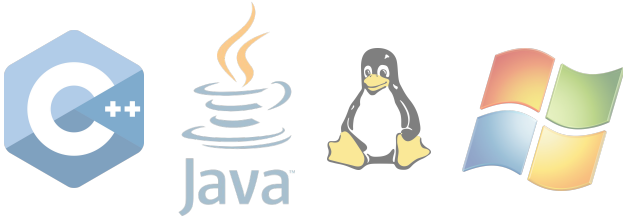
How to schedule tasks?

Preemptive scheduling v.s. **Cooperative** scheduling

Preemptive Scheduling



Linux Completely Fair Scheduler (CFS)



Mars Pathfinder: Priority Inversion Problem



Submitted By: Risat Mahmud Pathan

Cooperative Scheduling

Task A: `read("x"); update("x",x+1)`

Task B: `read(x); yield/await;`

Task X:

`while {}`

→ Time

How to prevent Coroutines from freezing Unity?

Unity Engine Scripting



S3M1CZ

Whenever I use Coroutines there is always a risk of it completely freezing my game upon pressing

Outage Postmortem - July 20, 2016

stackoverflow

Overview

On July 20, 2016 we experienced a 34 minute outage starting at 13:45. We identified the cause, 14 minutes to write the code to fix it, and the service was back up where Stack Overflow became available again.

The direct cause was a malformed post that caused one of our CPU on our web servers. The post was in the homepage and caused an `expression` to be called on each home page view. This caused the CPU to become unavailable since the load balancer took the server offline.

The events of July 20

CLOUDFLARE

On July 2, we deployed a new rule in our WAF Managed Rules that `caused CPUs to become exhausted` on every CPU core that handles HTTP/HTTPS traffic on the Cloudflare network worldwide. We are constantly improving WAF Managed Rules to respond to new vulnerabilities and threats. In May, for example, we used the speed with which we can update the WAF to `push a rule` to protect against a serious SharePoint vulnerability. Being able to deploy rules quickly and globally is a critical feature of our `WAF`.

Unfortunately, last Tuesday's update contained a `regular expression that backtracked enormously` and exhausted CPU used for HTTP/HTTPS serving. This brought down Cloudflare's core proxying, CDN and WAF functionality. The following graph shows CPUs dedicated to serving HTTP/HTTPS traffic spiking to nearly 100% usage across the servers in our network.



CPU utilization in one of our PoPs during the incident

runtime: non-cooperative goroutine preemption #24543

Closed



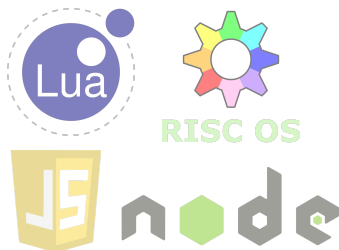
aclements opened on Mar 26, 2018 · edited by aclements

Edits Member ...

I propose that we solve `#10958` (preemption of tight loops) using non-cooperative preemption techniques. I have a detailed design proposal, which I will post shortly. This issue will track this specific implementation approach, as opposed to the general problem.

Edit: [Design doc](#)

Currently, Go currently uses compiler-inserted cooperative preemption points in function prologues. The majority of the time, this is good enough to allow Go developers to ignore preemption and focus on writing clear parallel code, but it has sharp edges that we've seen degrade the developer experience time and time again. When it goes wrong, it goes spectacularly wrong, leading to mysterious system-wide latency issues (`#17831`, `#19241`) and sometimes `complete freezes` (`#543`, `#12553`, `#13546`, `#14561`, `#15442`, `#17174`, `#20793`, `#21053`). And because this is a language implementation issue that exists outside of Go's language semantics, these failures are surprising and very difficult to debug.



Example: Regular Expression Denial-of-Service

Malicious request: {input: "aaaaaaaaaaaaaaaaaaaaa"}

```
function regex_handler(req, res) {  
  const match = "/(a+)+b/".match(req.body);  
}  
app.get("/regex", regex_handler);
```

Table 1: Results of our search for SL regexes in the npm and pypi module registries. Troublingly, 1% of unique regexes were SL regexes, affecting over 10,000 modules.

Registry	Total Modules	Scanned Modules	Unique Regexes	SL Regexes	Affected Modules
npm	565,219	375,652 (66%)	349,852	3,589 (1%)	13,018 (3%)
pypi	126,304	72,750 (58%)	63,352	704 (1%)	705 (1%)

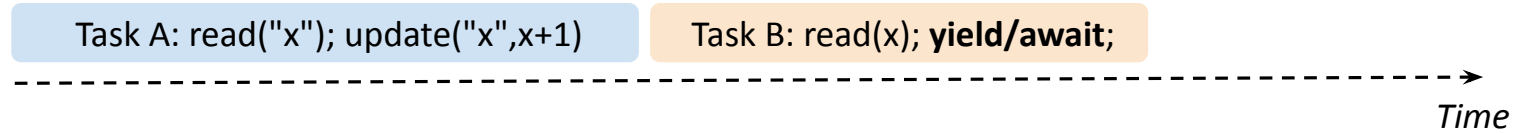
"safe-regex" or "worker_threads" -> Not generalizable to other problems

Stopify (PLDI'18), Compiler Interrupts (PLDI'21), Concord (SOSP'23) -> Not always practical

-> Not flexible

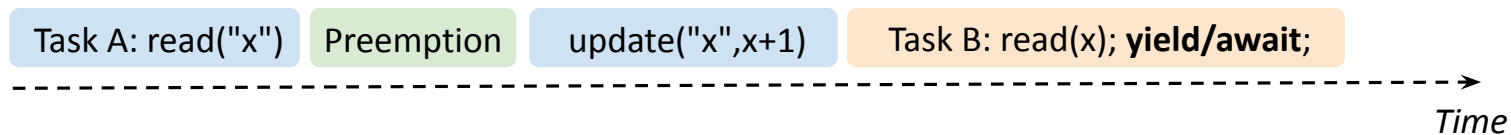
A **hybrid** scheduler that allows only **carefully controlled** and **developer-configurable** **preemption** in an otherwise cooperative environment.

A Hybrid Scheduling Runtime



Cooperative Abstraction

A Hybrid Scheduling Runtime



Safety Preemption  $x=x'$

Flexibility Preemption  Whether and when?

Timeout Health Check ... What task?

Cooperative Abstraction + **Controlled** and **Configurable** Preemption

Example: Regular Expression Denial-of-Service

Malicious request: {input: "aaaaaaaaaaaaaaaaaaaaa"}

```
import hybrid

function regex_handler(req, res) {
  const match = "/(a+)+b/".match(req.body);
}

app.get("/regex", regex_handler);
```

```
hybrid.registerPreemption ({
  "preemption_interval": 50, // ms
  "action": monitor });
```

Developer-configurable Policy and Action

```
function monitor() {
  if hybrid.getTaskExecTime(regex_handler) > 100 { // ms
    hybrid.abort();
  } else { hybrid.continue(); }
}
```

System Design

System Design: Abstraction—Introducing Configurable Preemption

```
func T;  
hybrid.registerPreemption ({  
  "preemption_interval":  $\varphi$ ,  
  "action":  $\alpha$  });
```



Cooperative Runtime

Main Execution
Context T

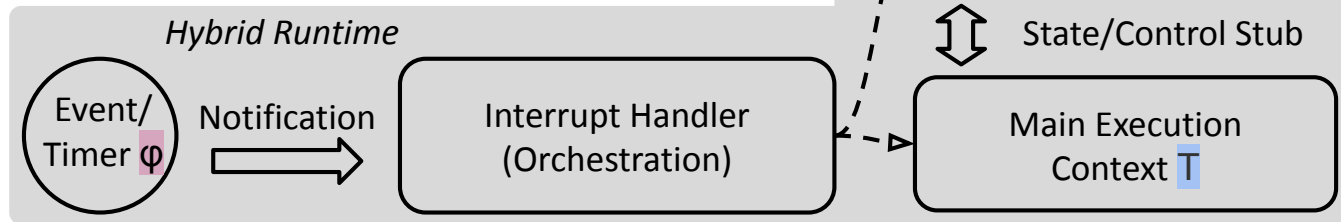


System Design: Underlying Mechanism and Implementation

```
func T;  
hybrid.registerPreemption ({  
  "preemption_interval":  $\varphi$ ,  
  "action":  $\alpha$  });
```



IPC type	Relative Latency (normalized)
User IPI	1.0
Signal	14.8

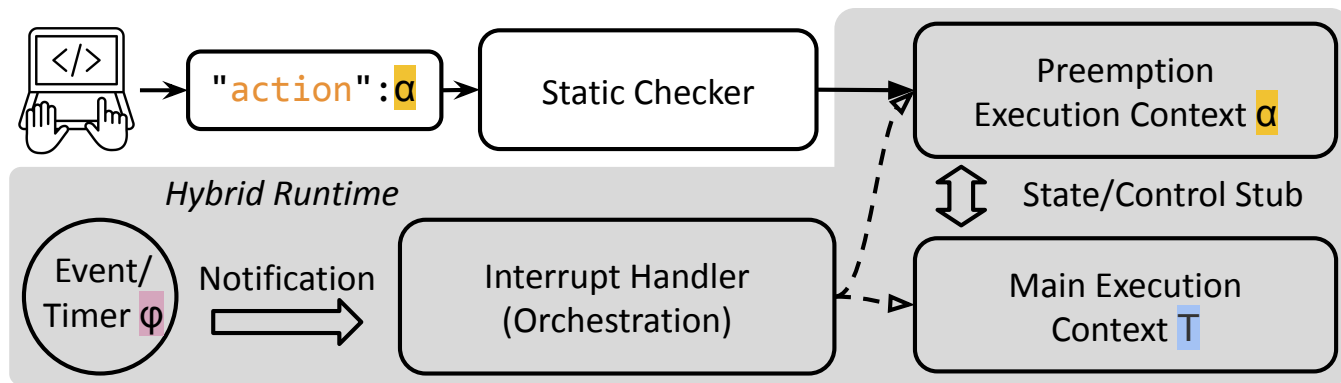


System Design: Safe Extensibility

```
func T;  
hybrid.registerPreemption ({  
  "preemption_interval":  $\varphi$ ,  
  "action":  $\alpha$  }));
```

```
< $\alpha$ > ::= "continue()" | "abort()" | <more>  
<more> ::= ...
```

(1) Termination (2) No Third-party Libs (3) No State Mutation



Broader Uses

- Performance: prevents head-of-line blocking, delayed garbage collections...

Long-running Task

Critical

Garbage Collection



RocksDB

- Security: prevents asymmetric denial-of-service and resource exhaustion...



ROBLOX



Unity



- Observability: supports instrumentations, profiling...

collector.execute				
collector.(*hwMonCol...	collector.(*...	collector.(*netStatColl...	collector.(*systemdCollector).Update	colle...
collector.(*hwMonCol...	collector.ge...	regexp.(*Regexp).doE...	collector.filterUnits	rege...
collector.collectS...	collector....	regexp.(*Regexp).b...	regexp.(*Regexp).doExecute	rege...
collector.explode...	regex...	regexp.(*Regexp)....	regexp.(*Rege... regexp.(*Regexp)...	
regexp.(*Regexp...	reg...		regexp.(*R...	
regexp.(*Rege...	reg...			
regexp.(*Reg...	reg...			
regexp.(...				

A hybrid scheduler that allows only **carefully controlled and developer-configurable preemption** in an otherwise cooperative environment.

Yizheng Xie

yizheng_xie@brown.edu

Di Jin

di_jin@brown.edu

Nikos Vasilakis

nikos@vasilak.is

