Function multiversioning in system image

All about vector registers

Yichao Yu

Harvard

Oct. 16, 2017

Implement function multi versioning in sysimg #21849

Open yuyichao wants to merge 9 commits into `master` from `yyc/codegen/clone`

Conversation 99 | Commits 9 | Files changed 30

yuyichao commented on 13 May • edited — Member

Current TODO:

- [x] Implement more flexible cloning check
- [x] Handle "PLT" cloning
- [x] Generate dispatch data
- [x] Update sysimg loading
- [x] Fix `code_native`
- [x] ? Figure out what's wrong with AVX on x86.

Goal

Amount of implementation details

- Just enough to make Jeff happy
- Without losing the audience
- With a single version of the talk (**Important!**)

Goal

Amount of implementation details

- Just enough to make Jeff happy
- Without losing the audience
- With a single version of the talk (**Important!**)

Goal

Amount of implementation details

- Just enough to make Jeff happy
- Without losing the audience
- With a single version of the talk (**Important!**)

Goal

Amount of implementation details

- Just enough to make Jeff happy
- Without losing the audience
- With a single version of the talk **(Important!)**

How did it all begin

No performance scaling in threading #17395

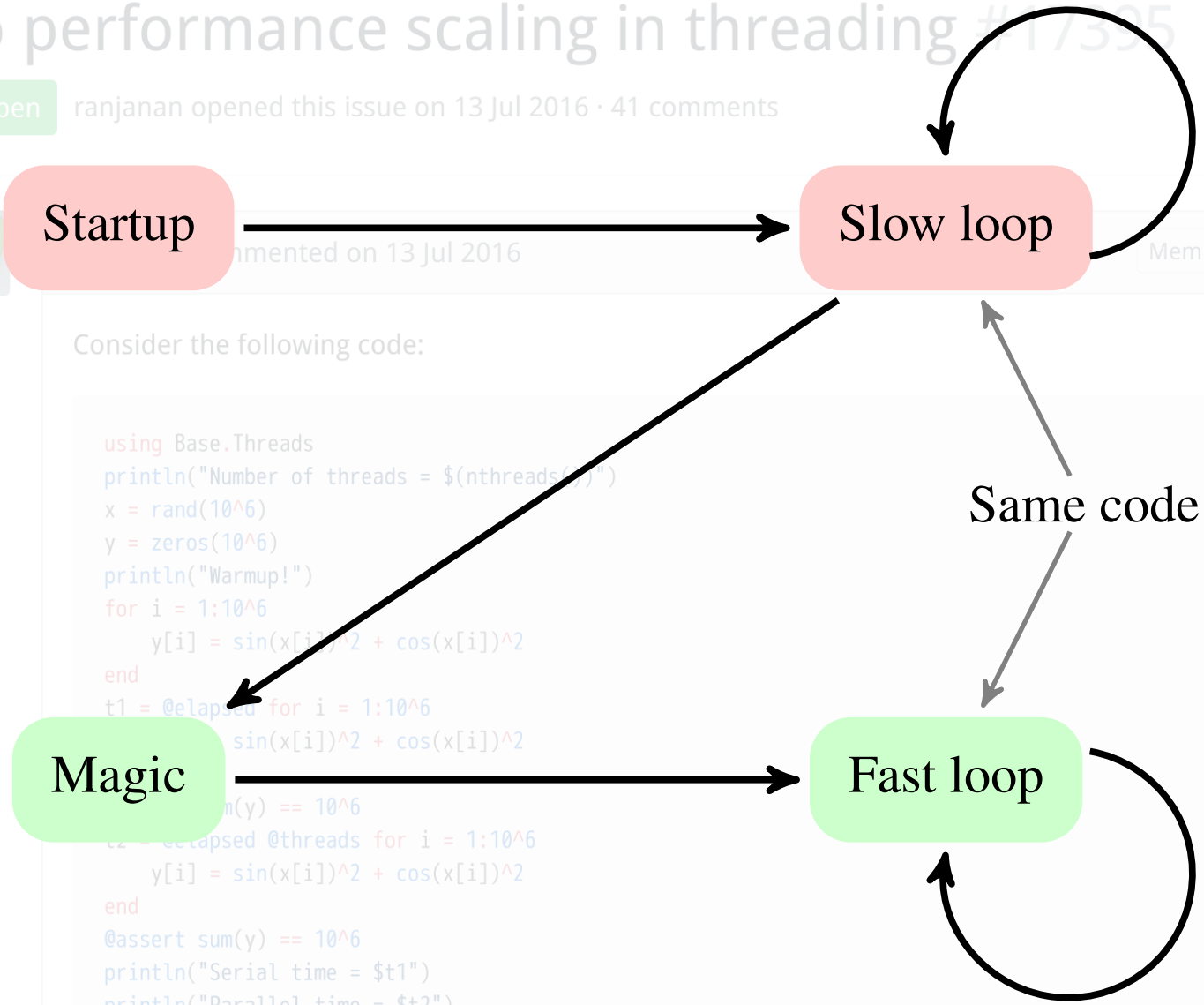
Open ranjanan opened this issue on 13 Jul 2016 · 41 comments

ranjanan commented on 13 Jul 2016 Member

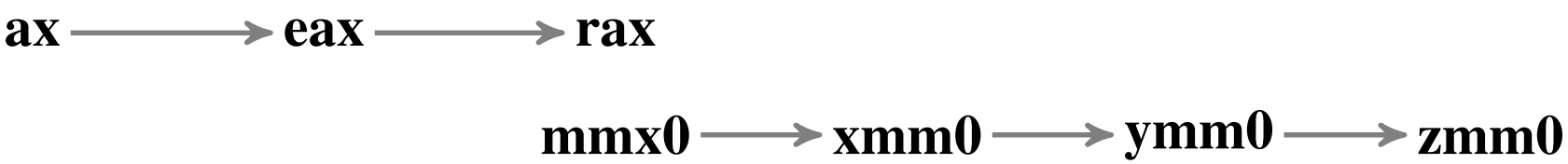
Consider the following code:

```
using Base.Threads
println("Number of threads = $(nthreads())")
x = rand(10^6)
y = zeros(10^6)
println("Warmup!")
for i = 1:10^6
    y[i] = sin(x[i])^2 + cos(x[i])^2
end
t1 = @elapsed for i = 1:10^6
    y[i] = sin(x[i])^2 + cos(x[i])^2
end
@assert sum(y) == 10^6
t2 = @elapsed @threads for i = 1:10^6
    y[i] = sin(x[i])^2 + cos(x[i])^2
end
@assert sum(y) == 10^6
println("Serial time = $t1")
println("Parallel time = $t2")
```

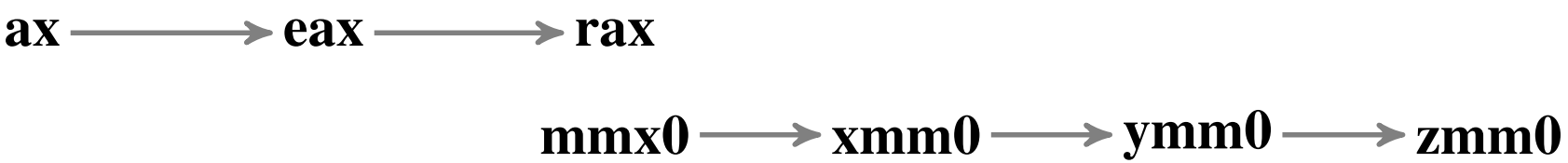
How did it all begin



Register size



Register size



More/larger registers ⟶ More states to manage

Register size

ax → **eax** → **rax**

mmx0 → **xmm0** → **ymm0** → **zmm0**

More/larger registers → More states to manage

(Intel got something wrong almost everytime)

Compile openlibm for AVX (ymm registers)

- GCC function attribute *target_clone*
  ...specify that a function be cloned into multiple versions compiled with different target options ...It also creates a resolver function (...) that dynamically selects a clone suitable for current architecture ...

- Has bug on current GCC versions so not usable in openlibm.
  But it's a good start.

Compile openlibm for AVX (ymm registers)

- GCC function attribute *target_clone*
  ...specify that a function be cloned into multiple versions compiled with different target options ...It also creates a resolver function (...) that dynamically selects a clone suitable for current architecture ...

- Has bug on current GCC versions so not usable in openlibm.
  But it's a good start.

Why function multiversioning

- For code in the system image
- For JIT code

Why function multiversioning

- For code in the system image
- For JIT code

Why function multiversioning

- For code in the system image
- For JIT code

  JIT features > system image features

Why function multiversioning

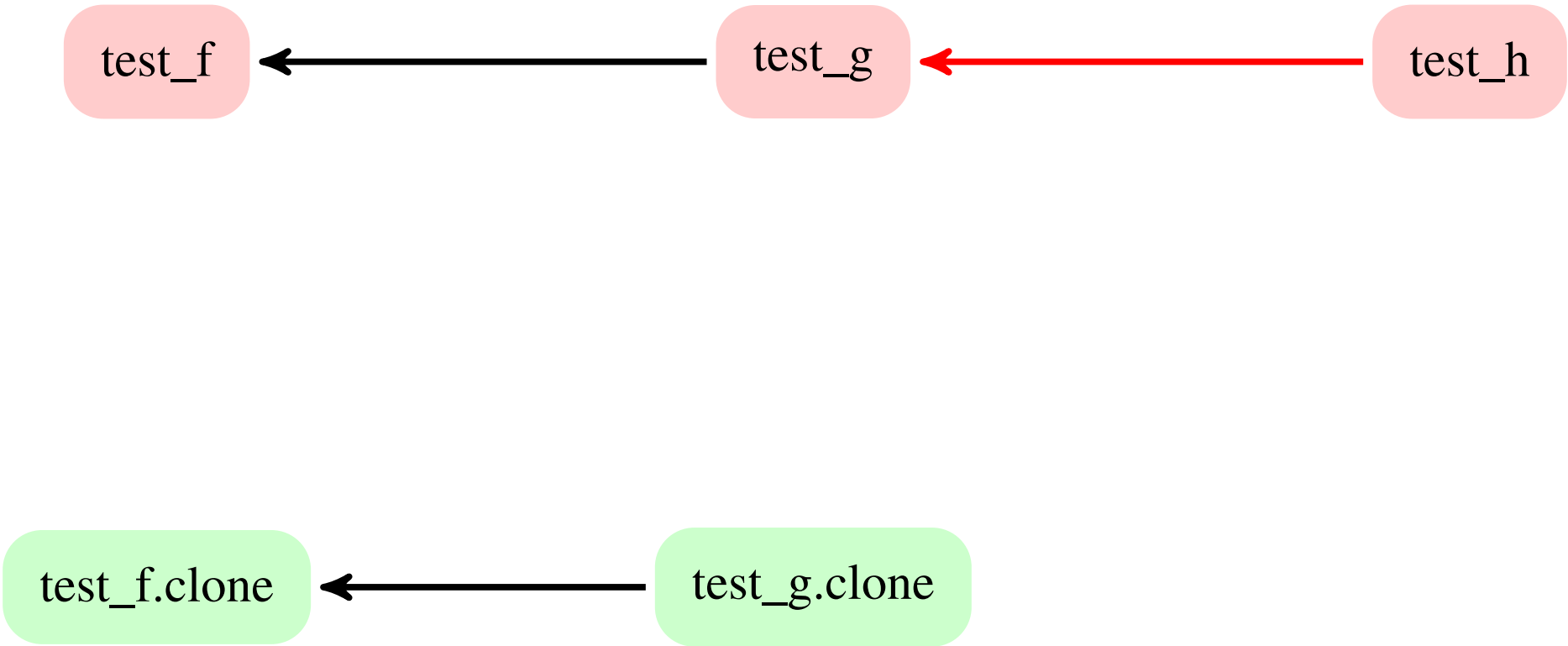
- For code in the system image
- For JIT code

  JIT features > system image features

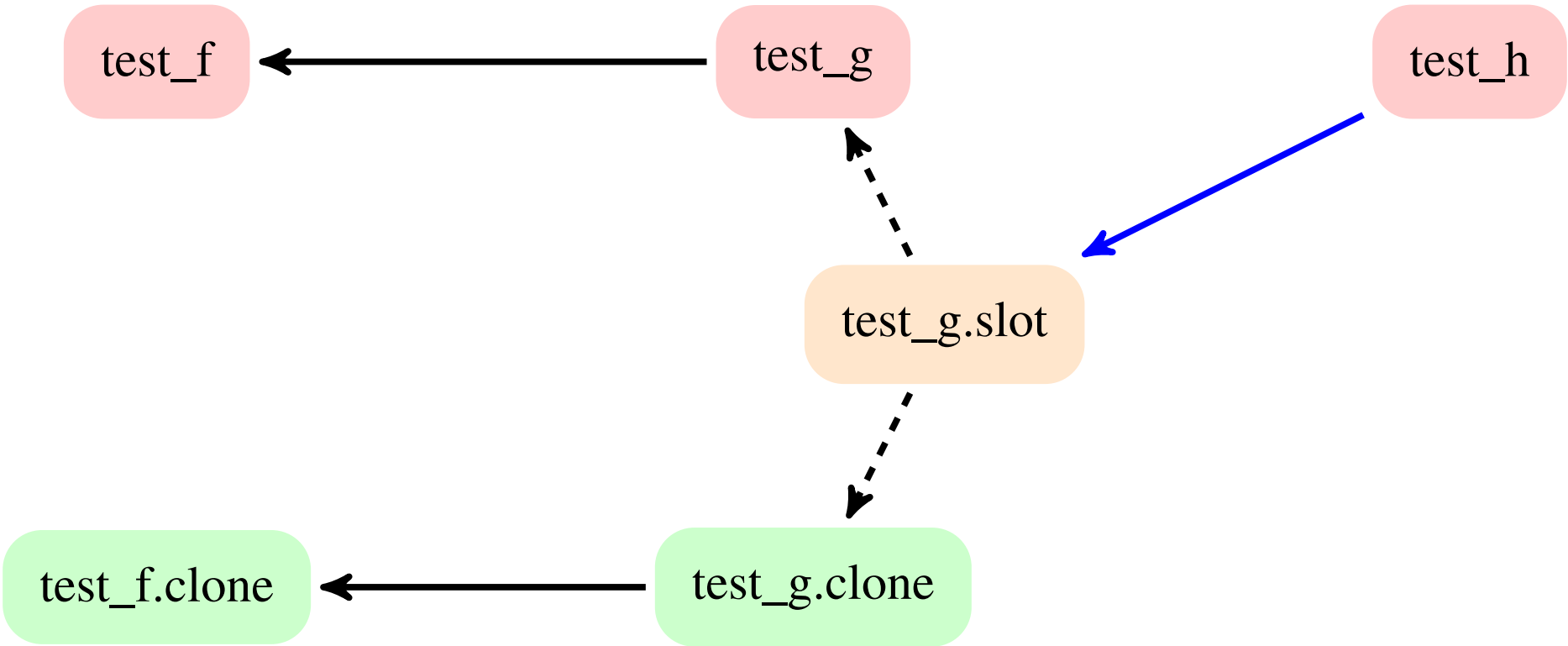
  JIT register size == system register size

Cloning

Cloning



Cloning



Next step

- Expose user API
  - Platform agnostic, e.g. “fma”
  - Platform specific, e.g. “x86::avx2”
- Make use of this in other part of runtime

Next step

- Expose user API
  - Platform agnostic, e.g. “fma”
  - Platform specific, e.g. “x86::avx2”
- Make use of this in other part of runtime

Next step

- Expose user API
  - Platform agnostic, e.g. “fma”
  - Platform specific, e.g. “x86::avx2”
- Make use of this in other part of runtime

Next step

- Expose user API
  - Platform agnostic, e.g. “fma”
  - Platform specific, e.g. “x86::avx2”
- Make use of this in other part of runtime

Next step

- Expose user API
  - Platform agnostic, e.g. “fma”
  - Platform specific, e.g. “x86::avx2”
- Make use of this in other part of runtime

  crc32c, PLT, . . .