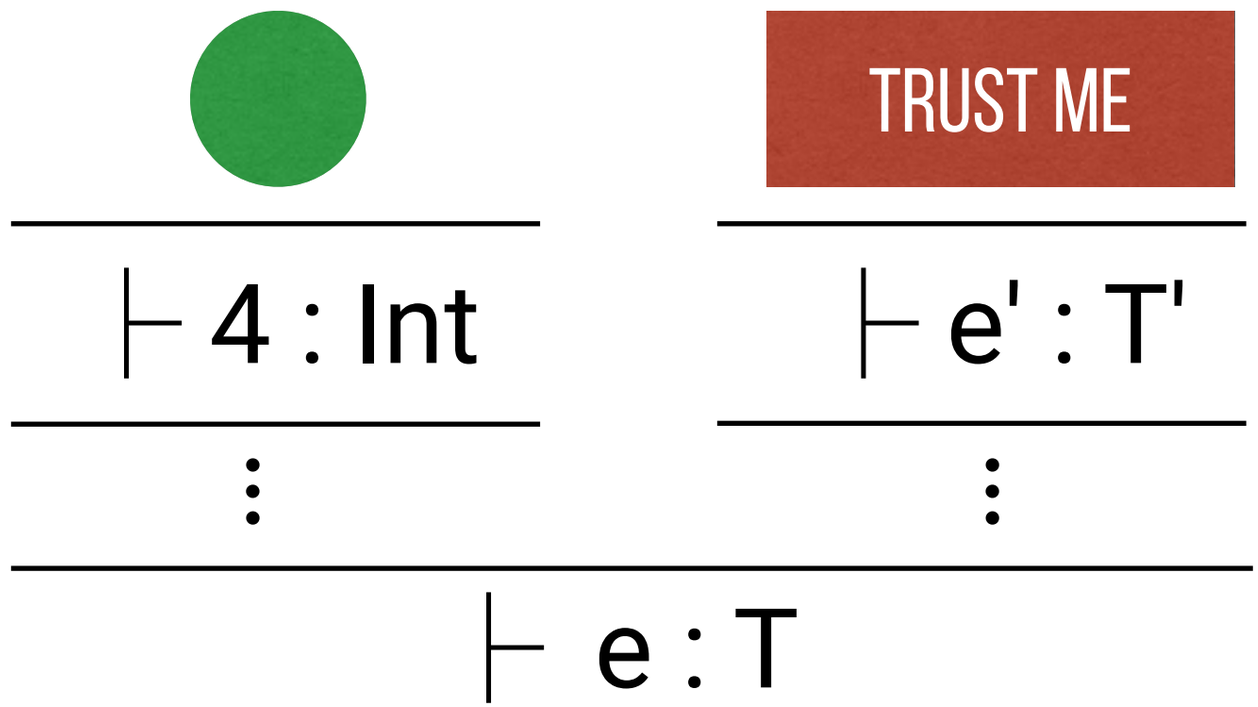
Transient Racket

Ben Greenman
Northeastern University

$$e : T$$

What is the meaning of types?

What is the meaning of types?



What is the meaning of types?

$$\frac{\dfrac{}{x{:}\mathsf{Int} \vdash x : \mathsf{Int}} \qquad \dfrac{y{:}\mathsf{Dyn} \qquad \mathsf{Dyn} \sim \mathsf{Int}}{\vdash y : \mathsf{Int}}}{x{:}\mathsf{Int}, y{:}\mathsf{Dyn} \vdash x + y : \mathsf{Int}}$$

$$\vdash (\lambda\, x\, y\, .\, x + y) : \mathsf{Int}\ \mathsf{Dyn} \to \mathsf{Int}$$

What is the meaning of types?

Easy!
Generalize classic soundness

Classic Type Soundness

If $\vdash e : T$ then either:

- e -->* v and $\vdash v : T$
- e diverges
- e -->* RuntimeError

Example: "RuntimeError"

$$\frac{\dots. \vdash x : \mathsf{Int} \qquad \dots. \vdash y : \mathsf{Int} \quad \blacklozenge \text{ Assuming y non-zero}}{\dots. \vdash x \,//\, y : \mathsf{Int}}$$

Classic Type Soundness

If $\vdash e : T$ then either:

- e -->* v and $\vdash v : T$
- e diverges
- e -->* RuntimeError

Generalized Type Soundness

If $\vdash e : T$ then either:

- $e \rightarrow^* v$ and $\vdash v : T$
- e diverges
- $e \rightarrow^*$ RuntimeError
- $e \rightarrow^*$ CheckError

Example: CheckError #1

Int Dyn
((λ x y . x + y) 2 "NaN") -->T

Dyn
((λ y . 2 + y) "NaN") -->T

Dyn ~ Int
2 + "NaN" -->T

CheckError

Example: CheckError #2

$$((\lambda\ x\ y\ .\ x + y)\ 2\ \text{"NaN"}) \quad \overset{\text{Int Int}}{\text{-->T}}$$

$$((\lambda\ y\ .\ 2 + y)\ \text{"NaN"}) \quad \overset{\text{Int}}{\text{-->T}}$$

CheckError

Example: CheckError #3

Int->Int

((λ f . ....) (λ x . "hello")) -->T

CheckError

(SNAPL 2017)

Generalized Soundness

If $\vdash e : T$ then either:

- e -->* v and $\vdash v : T$
- e diverges
- e -->* RuntimeError
- e -->* CheckError

Classic vs. Generalized

RuntimeError ~ assumption about ∂

CheckError ~ assumption about $\vdash$

Practical Issues

(Generalized) Soundness

If $\vdash e : T$ then either:

- e -->* v and $\vdash v : T$
- e diverges
- e -->* RuntimeError
- e -->* CheckError

How to implement checks?

$$\vdash v : T$$

A. Run the type checker

B. Check finite values,
   monitor behaviors (for infinite values)

Monitor Behaviors

((λ f . ....) (λ x . "hello")) Int->Int -->T

.... [f ↦ v+]

where v+ = mon(Int->Int, (λ x . "hello"))

Generalized Soundness v2

If $\vdash e : T$ then either:

- e -->* v and [(e -->* v) ~~$\vdash$~~ v ~~: T~~]
- e diverges
- e -->* RuntimeError
- e -->* CheckError

Case closed?

No!

Monitor Behaviors

((λ f . ....) (λ x . "hello")) -->T

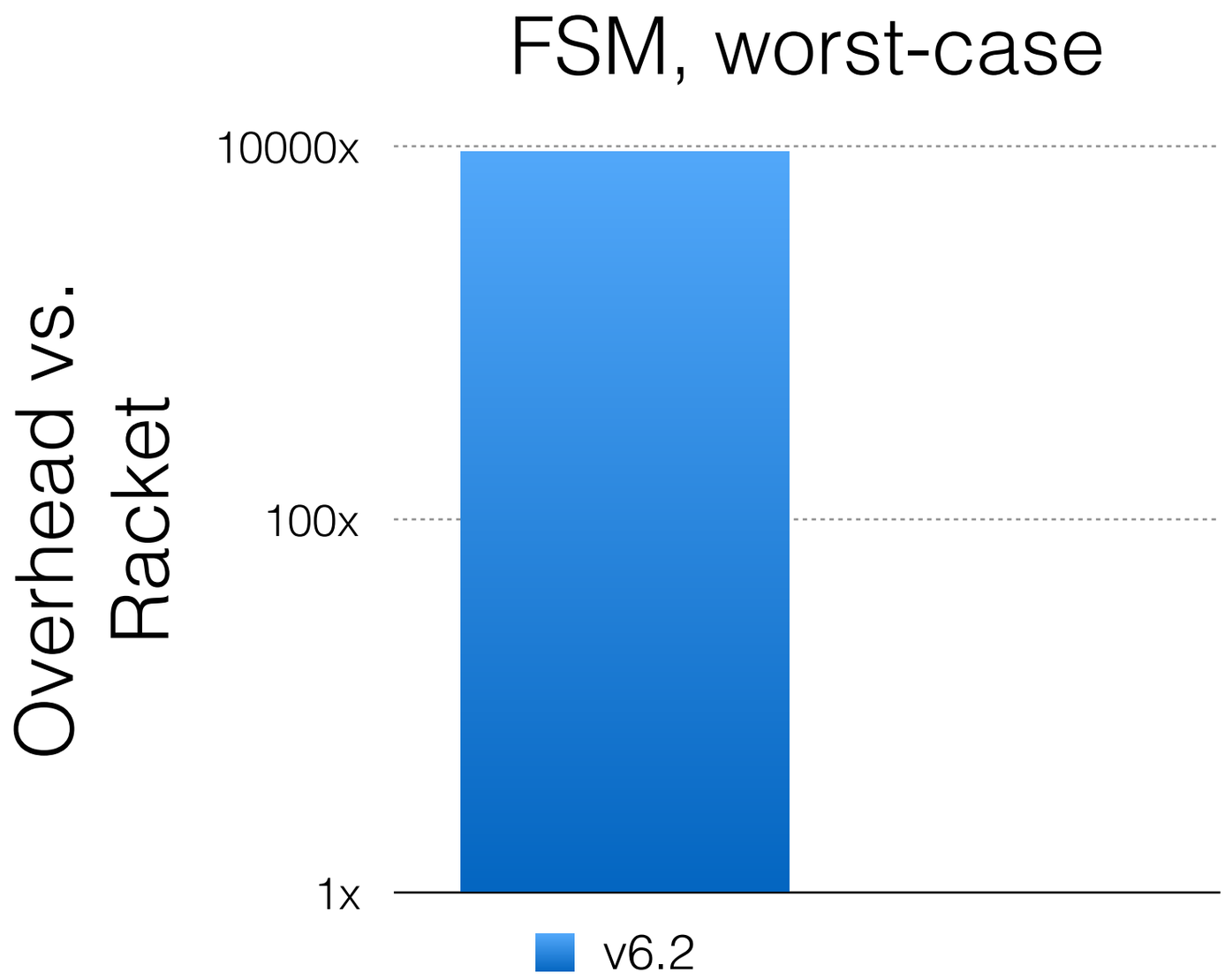
.... [f -> v+]

where v+ = mon(Int->Int, (λ x . "hello"))

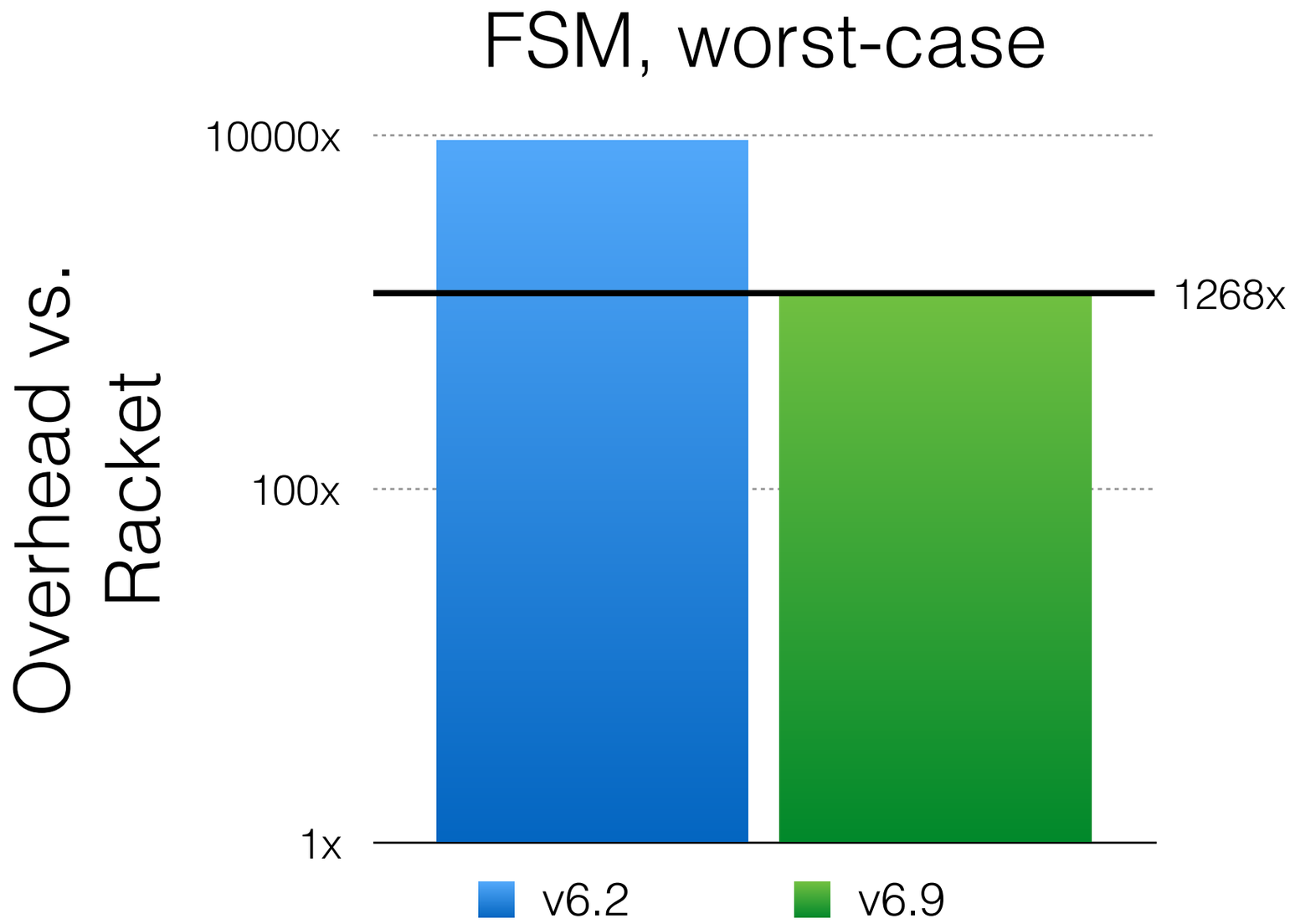
Costs of Monitoring

1. Checking
2. Allocation
3. Interposition

Costs of Monitoring



Costs of Monitoring



Any Program, Any Types

"[P]rogrammers should be able to add or remove type annotations without any unexpected impacts on their program" -- SNAPL 2015

$$e : T$$

Big Types in Little Runtime

Open-World Soundness and Collaborative Blame for Gradual Type Systems

Michael M. Vitousek Cameron Swords Jeremy G. Siek
Indiana University, USA
{mvitouse,cswords,jsiek}@indiana.edu

Tag Soundness

If $\vdash e : T$ then $\vdash e : \lfloor T \rfloor$ and either:

- $e \dashrightarrow^* v$ and $\vdash v : \lfloor T \rfloor$
- e diverges
- $e \dashrightarrow^*$ RuntimeError
- $e \dashrightarrow^*$ CheckError

$$\lfloor T \rfloor = K$$

- $\lfloor Int \rfloor = Int$
- $\lfloor List(T) \rfloor = List$
- $\lfloor T \to T \rfloor = \to$

Tag Soundness

If $\vdash e : T$ then $\vdash e : \lfloor T \rfloor$ and either:

- $e \dashrightarrow^* v$ and $\vdash v : \lfloor T \rfloor$
- e diverges
- $e \dashrightarrow^*$ RuntimeError
- $e \dashrightarrow^*$ CheckError

Tradeoffs

- ⌊T⌋ weaker than T
- Weaker compositional reasoning

+ O(1) to check ⌊T⌋
+ No allocation, "less" interposition

Tag Example #1

Int Dyn
((λ x y . x + y) 2 "NaN") -->K

Dyn
((λ y . 2 + y) "NaN") -->K

Dyn ~ Int
2 + "NaN" -->K

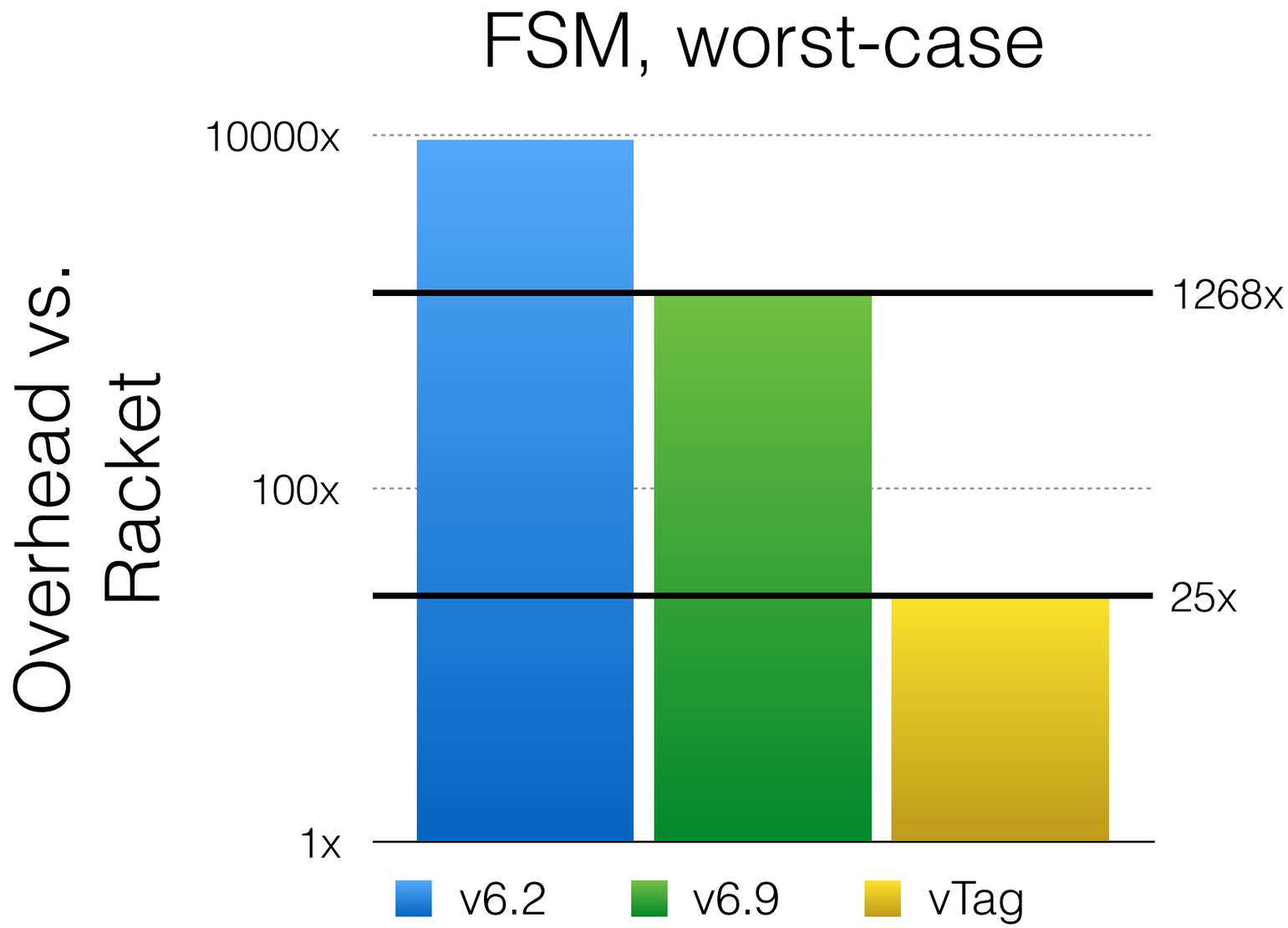
CheckError

Tag Example #2

((λ f$^{->}$. ....) (λ x . "hello")) -->K

.... [f -> (λ x . "hello")] -->K

Costs of Monitoring



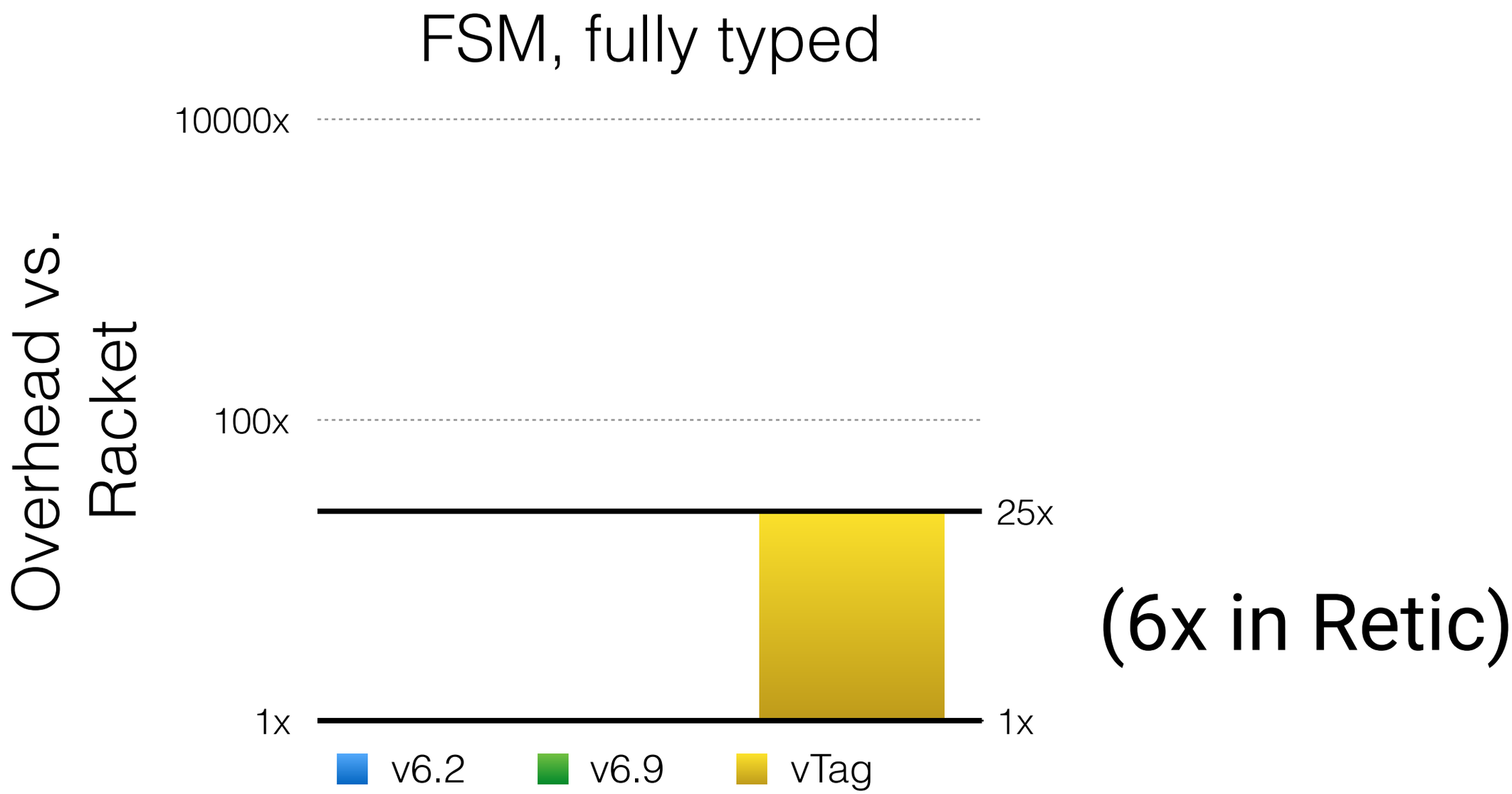
Coming Soon

```
#lang transient/racket

;; tag soundness
;; O(1) checks
;; no monitors
```

2 Closing Thoughts

Monitors are pay-as-you-go



(6x in Retic)

JIT compilation

C[((λ a b . e) v0 v1)] -->K ????

Need to check inputs?

Depends on the context!

2^2 versions of e

"Preservation" types

- Generalized Soundness: -->T
- Tag Soundness: -->K
- TypeScript: -->Dyn

"Preservation" types

- Generalized Soundness: -->T
- Tag Soundness: -->K
- TypeScript: -->Dyn