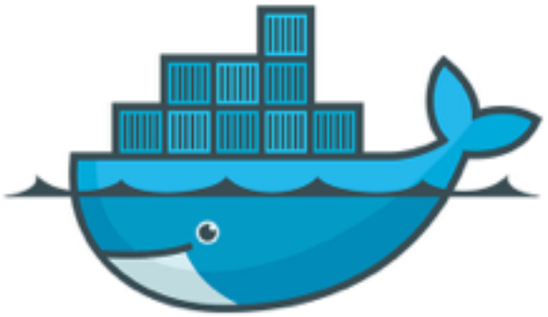
Run-Fail-Grow: Creating Tailored Object-Oriented Runtimes

Guille Polito
@guillep
https://guillep.github.io/

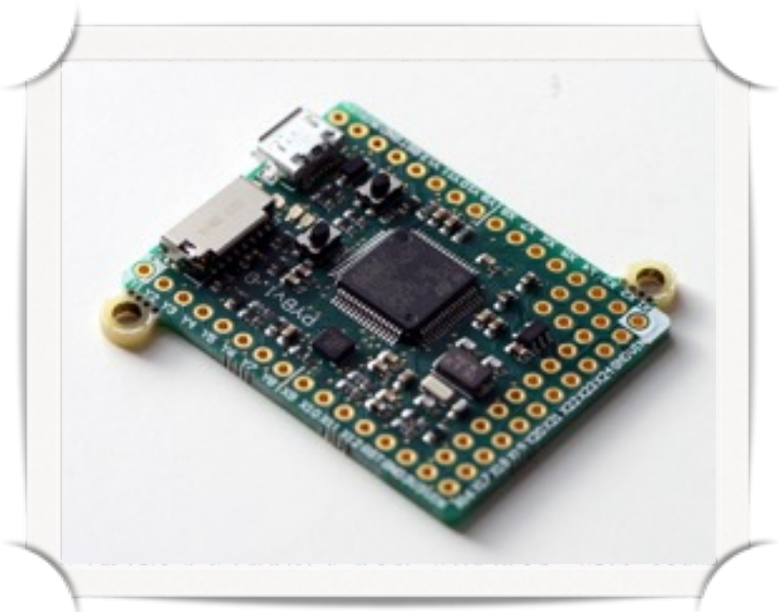
G. Polito, L. Fabresse, N. Bouraqadi, S. Ducasse,
in Journal of Object Technology

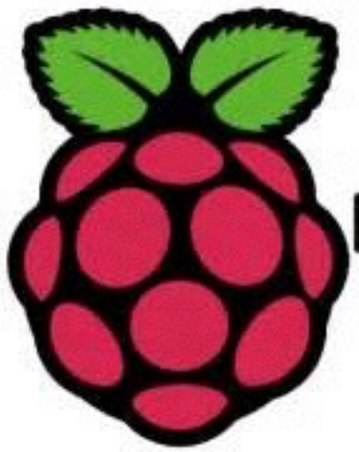
A(gain) need for
small and **fast startup**

- Micro Services
- Container architectures
- Small devices



docker





RaspberryPi

Lets look at an application...

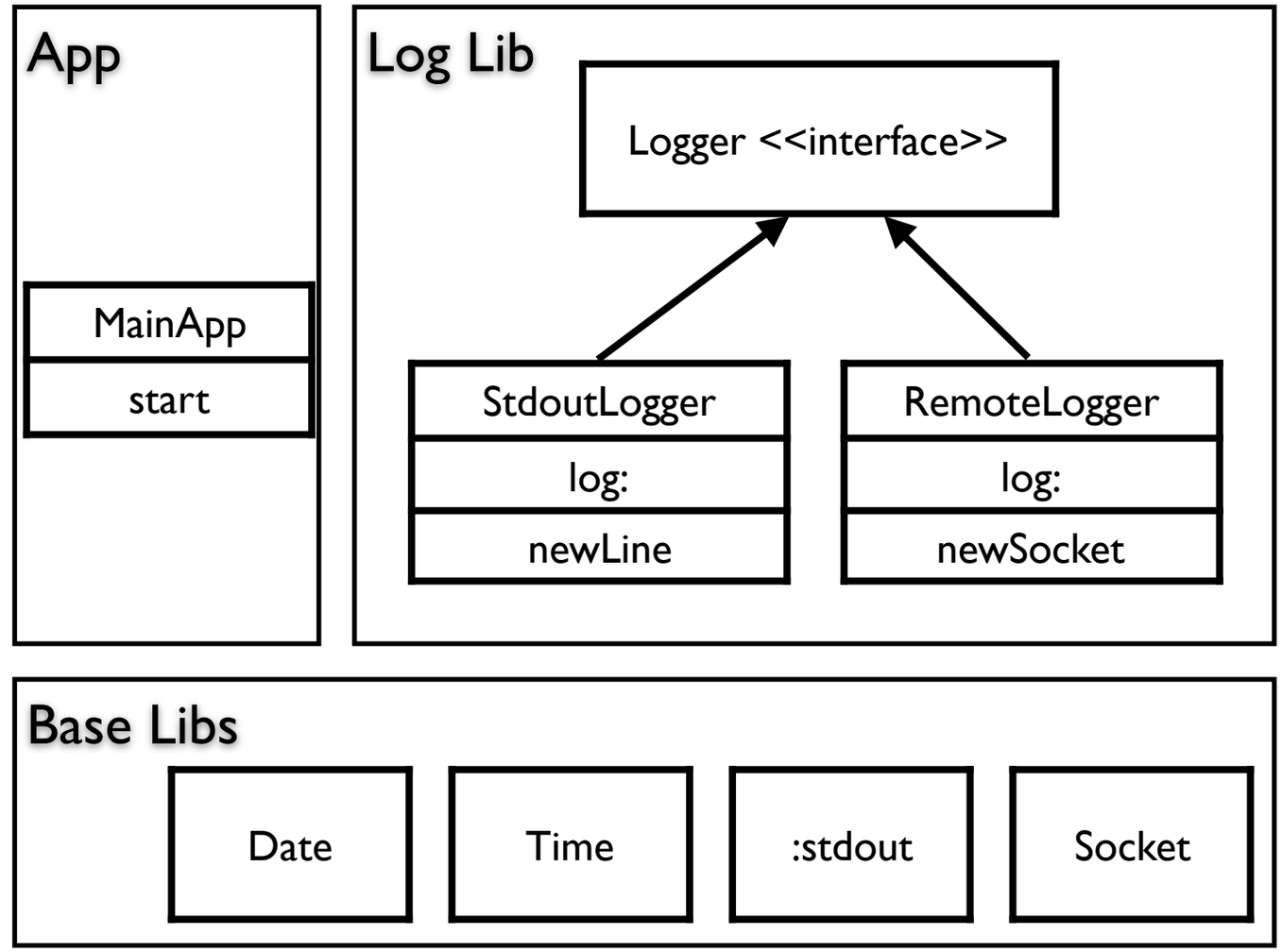
```
MainApp>>start
  logger := StdoutLogger new.
  logger log: 'Application has started'.
  "do something"
  logger log: 'Application has finished'.

StdoutLogger»newLine
  stdout newLine.

StdoutLogger»log: aMessage
  stdout nextPutAll: Time now printString.
  stdout nextPutAll: aMessage.
  stdout newLine.

RemoteLogger»log: aMessage
  | socket |
  socket := self newSocket.
  socket nextPutAll: Time now printString.
  socket nextPutAll: aMessage.
  socket newLine.

RemoteLogger»newSocket
  "...."
  "creates an instance of socket given some configuration"
```



There is *extra* stuff!

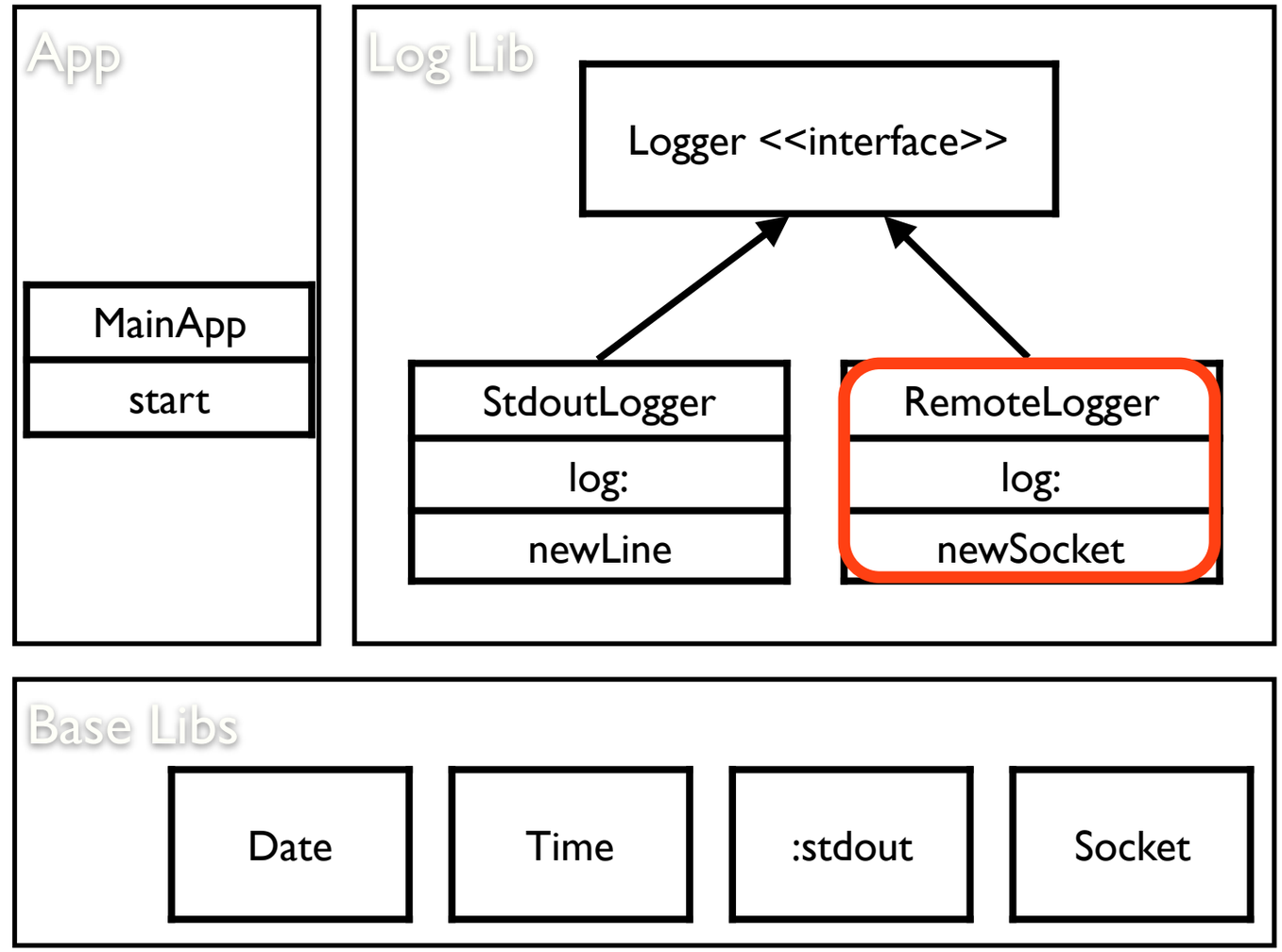
```
MainApp>>start
  logger := StdoutLogger new.
  logger log: 'Application has started'.
  "do something"
  logger log: 'Application has finished'.

StdoutLogger»newLine
  stdout newLine.

StdoutLogger»log: aMessage
  stdout nextPutAll: Time now printString.
  stdout nextPutAll: aMessage.
  stdout newLine.

RemoteLogger»log: aMessage
  | socket |
  socket := self newSocket.
  socket nextPutAll: Time now printString.
  socket nextPutAll: aMessage.
  socket newLine.

RemoteLogger»newSocket
  "...."
  "creates an instance of socket given some configuration"
```



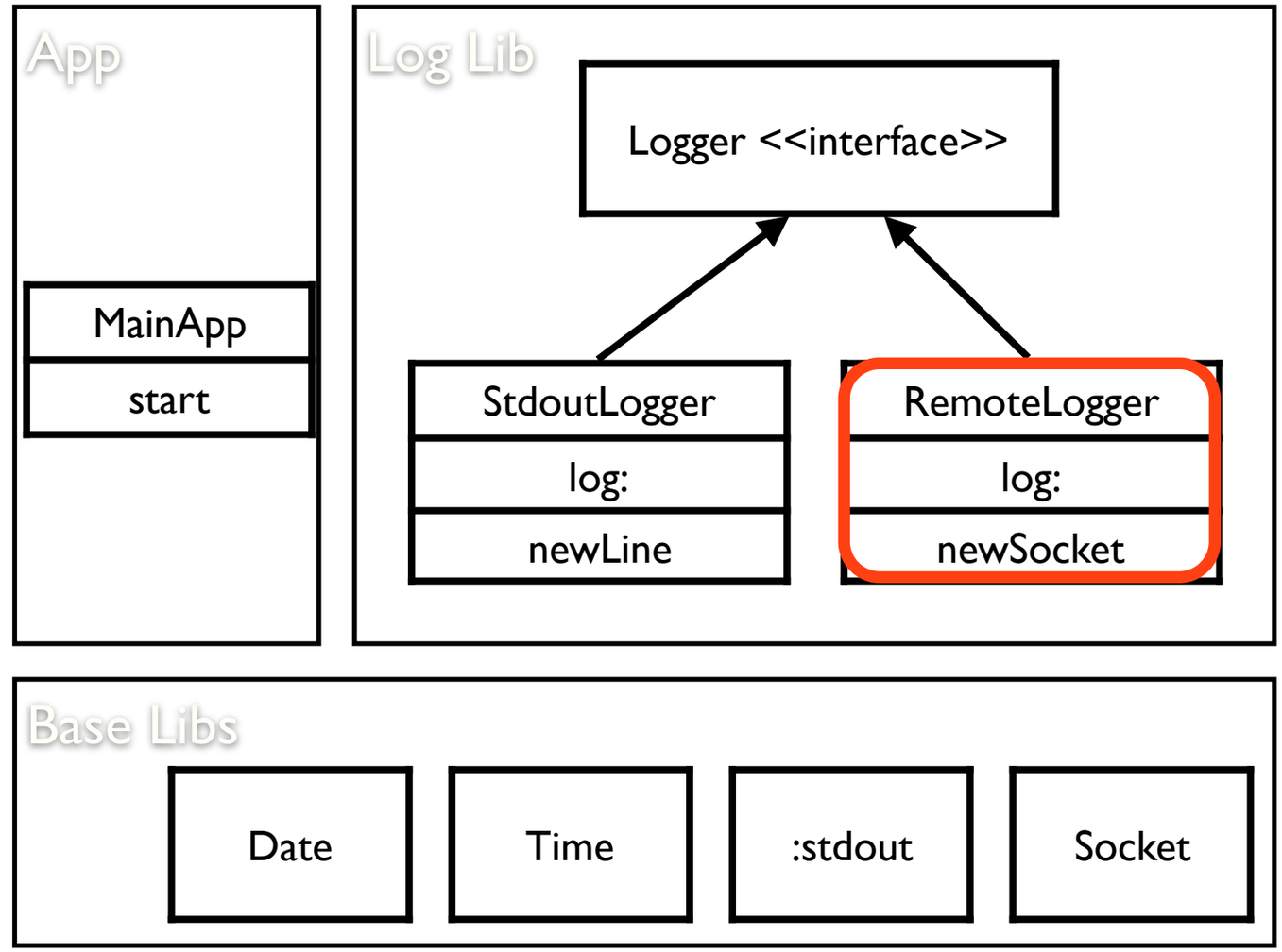
```
MainApp>>start
  logger := StdoutLogger new.
  logger log: 'Application has started'.
  "do something"
  logger log: 'Application has finished'.

StdoutLogger»newLine
  stdout newLine.

StdoutLogger»log: aMessage
  stdout nextPutAll: Time now printString.
  stdout nextPutAll: aMessage.
  stdout newLine.

RemoteLogger»log: aMessage
  | socket |
  socket := self newSocket.
  socket nextPutAll: Time now printString.
  socket nextPutAll: aMessage.
  socket newLine.

RemoteLogger»newSocket
  "...."
  "creates an instance of socket given some configuration"
```



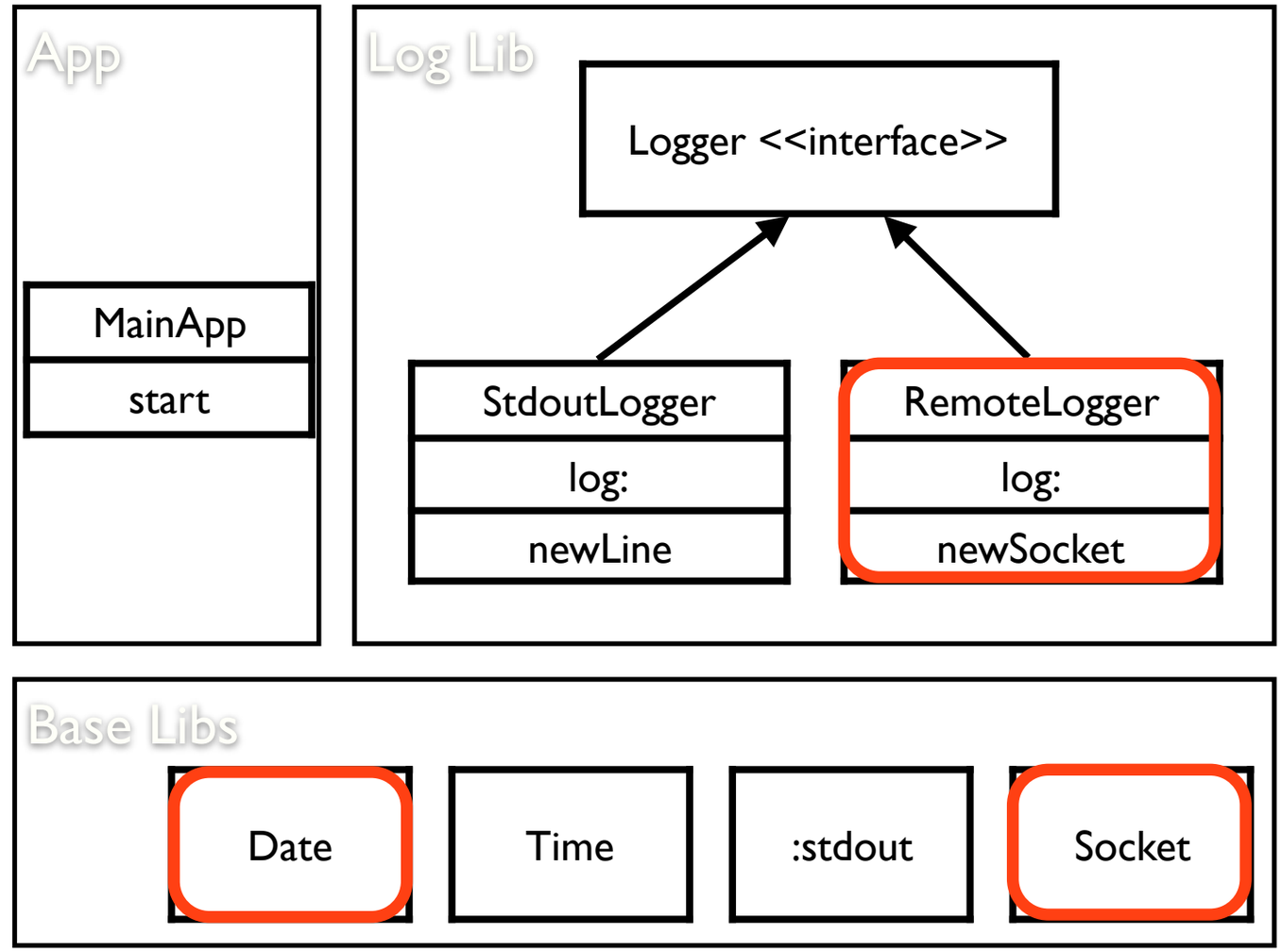
```
MainApp>>start
  logger := StdoutLogger new.
  logger log: 'Application has started'.
  "do something"
  logger log: 'Application has finished'.

StdoutLogger»newLine
  stdout newLine.

StdoutLogger»log: aMessage
  stdout nextPutAll: Time now printString.
  stdout nextPutAll: aMessage.
  stdout newLine.

RemoteLogger»log: aMessage
  | socket |
  socket := self newSocket.
  socket nextPutAll: Time now printString.
  socket nextPutAll: aMessage.
  socket newLine.

RemoteLogger»newSocket
  "...."
  "creates an instance of socket given some configuration"
```



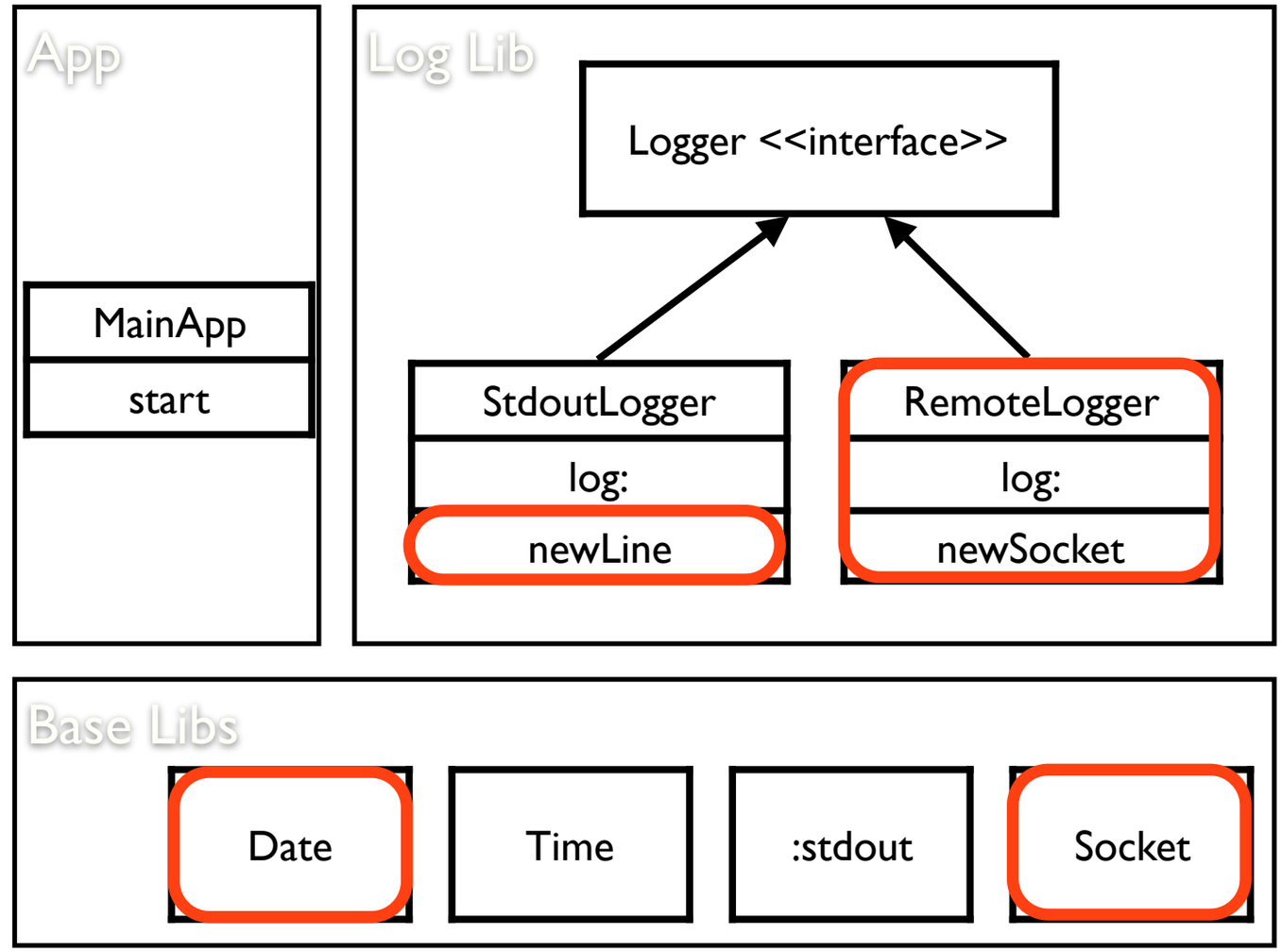
```
MainApp>>start
  logger := StdoutLogger new.
  logger log: 'Application has started'.
  "do something"
  logger log: 'Application has finished'.
```

```
StdoutLogger»newLine
  stdout newLine.
```

```
StdoutLogger»log: aMessage
  stdout nextPutAll: Time now printString.
  stdout nextPutAll: aMessage.
  stdout newLine.
```

```
RemoteLogger»log: aMessage
  | socket |
  socket := self newSocket.
  socket nextPutAll: Time now printString.
  socket nextPutAll: aMessage.
  socket newLine.
```

```
RemoteLogger»newSocket
  "...."
  "creates an instance of socket given some configuration"
```



But there is not only **BLOAT** *in*

application code

- Core Libraries take place and are distributed with the language
- Runtime objects take place also

Mariano Martinez Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse.
Visualizing objects and memory usage. In Smalltalks 2010

- And some of those are runtime meta-data, used for reflection

Research questions

Q: Can we tailor the runtime?

Q: How small can they be?

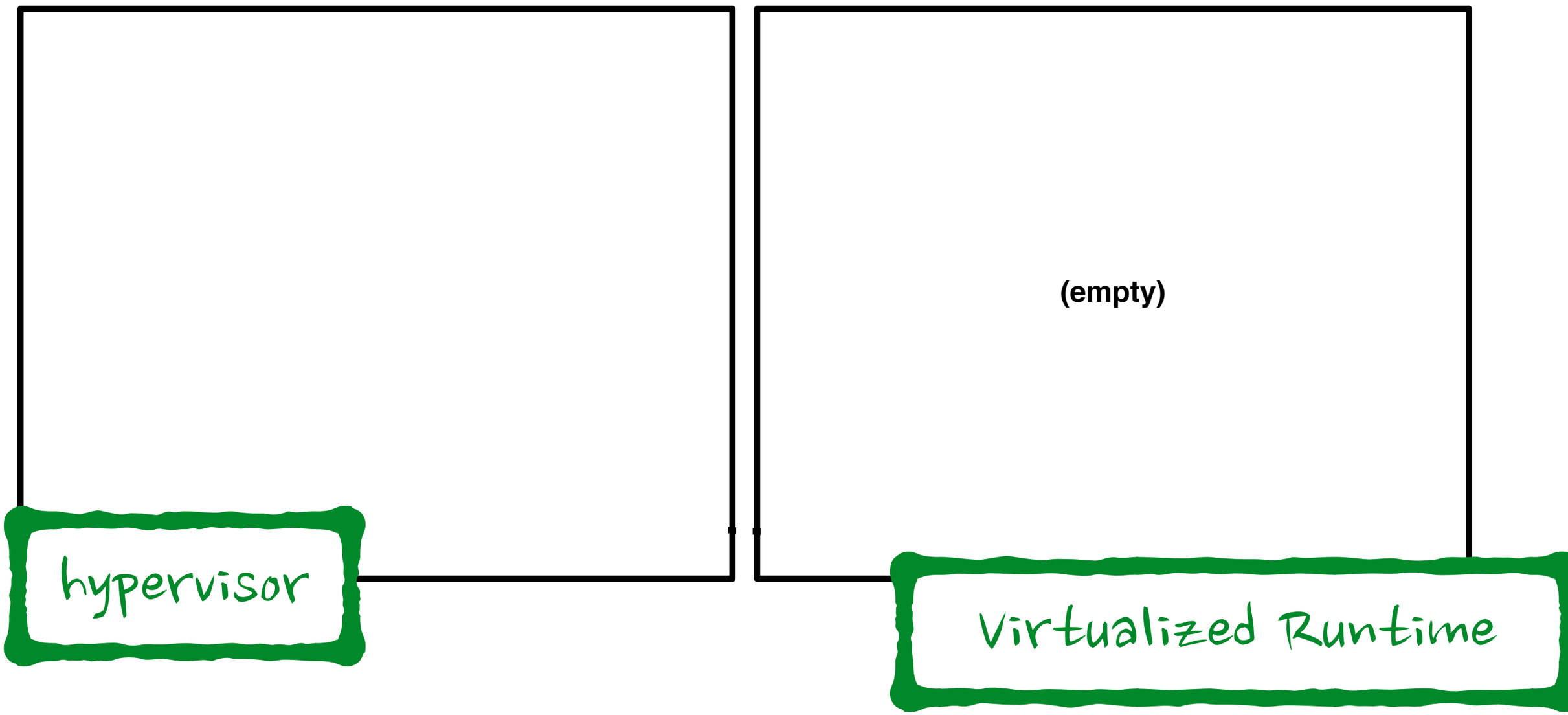
Our solution in a nutshell

a *runtime virtualisation* infrastructure

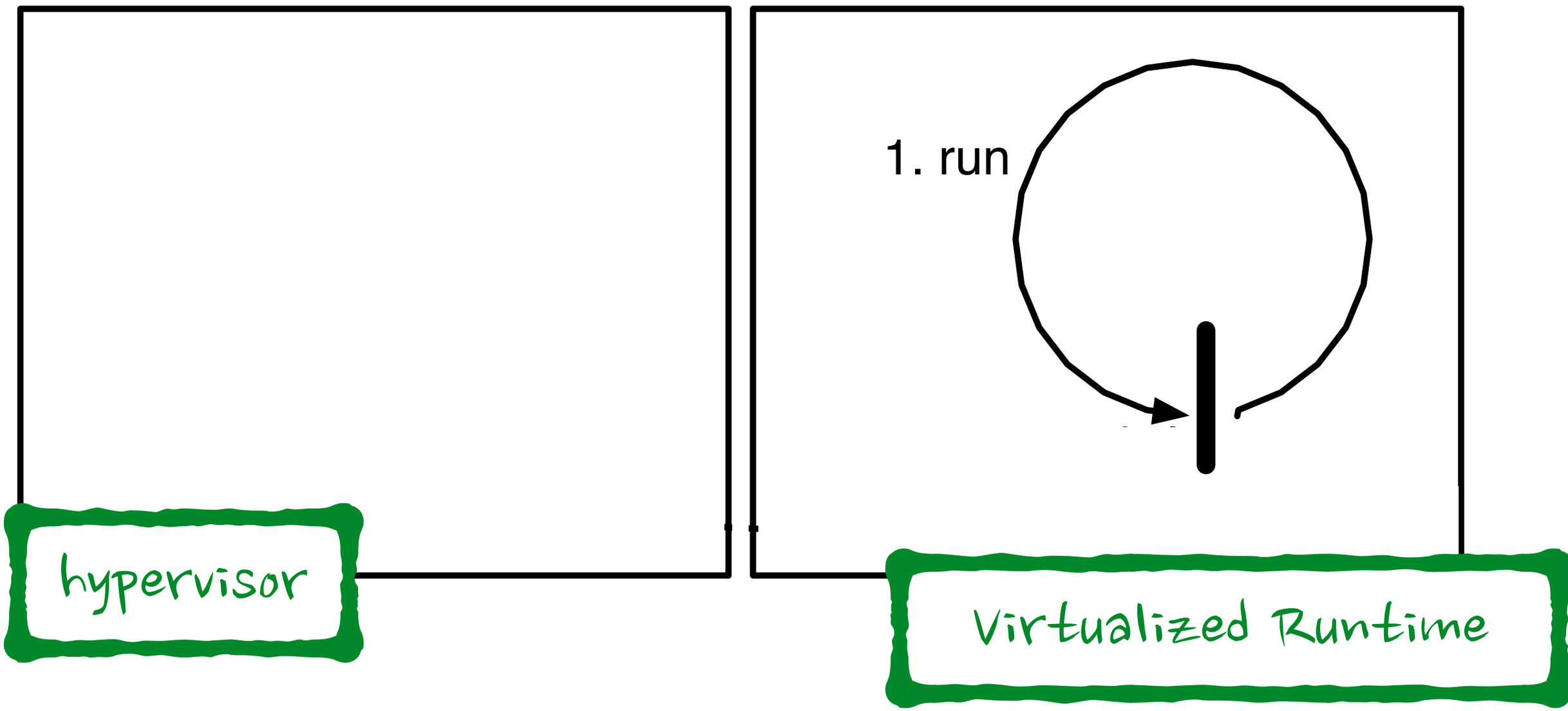
+

a **run-fail-grow** algorithm

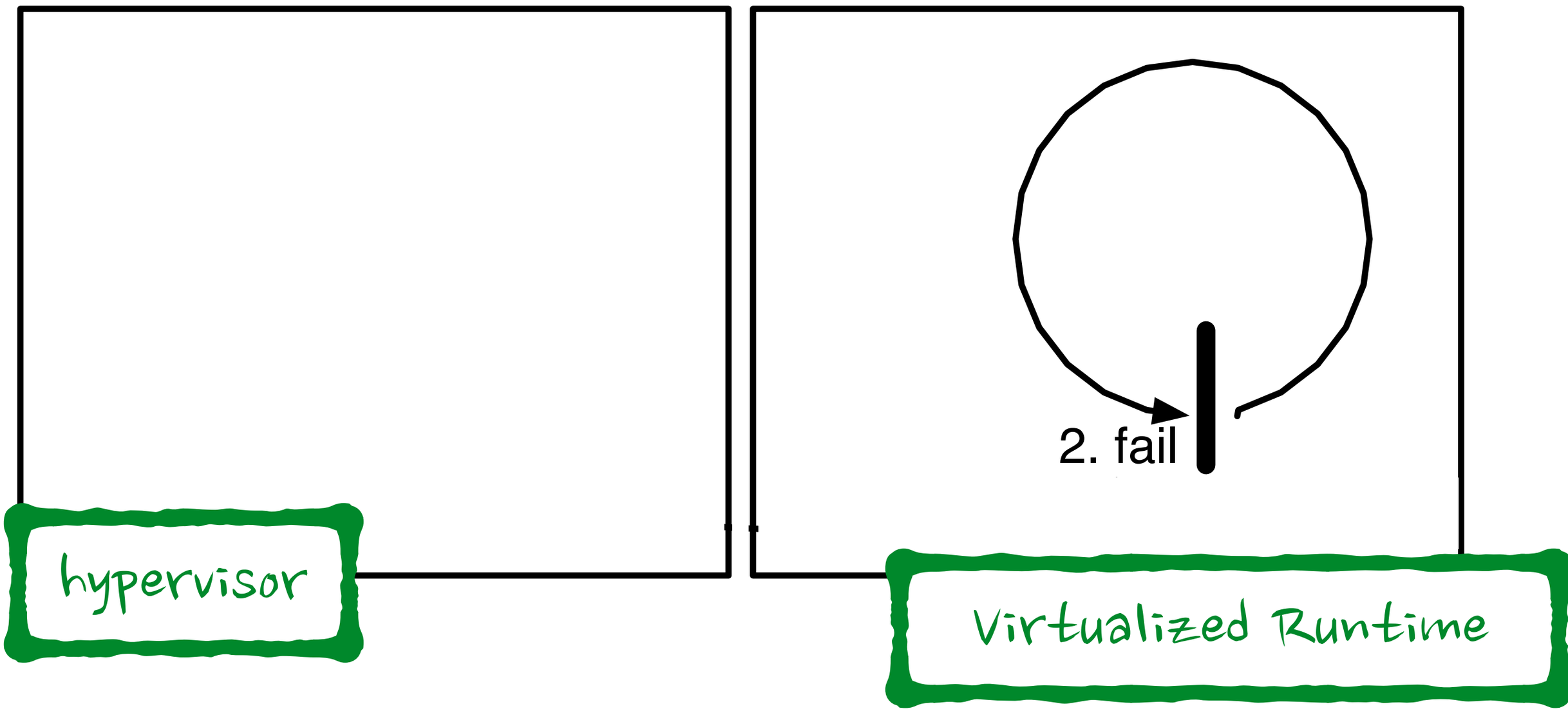
Tornado Run-Fail-Grow



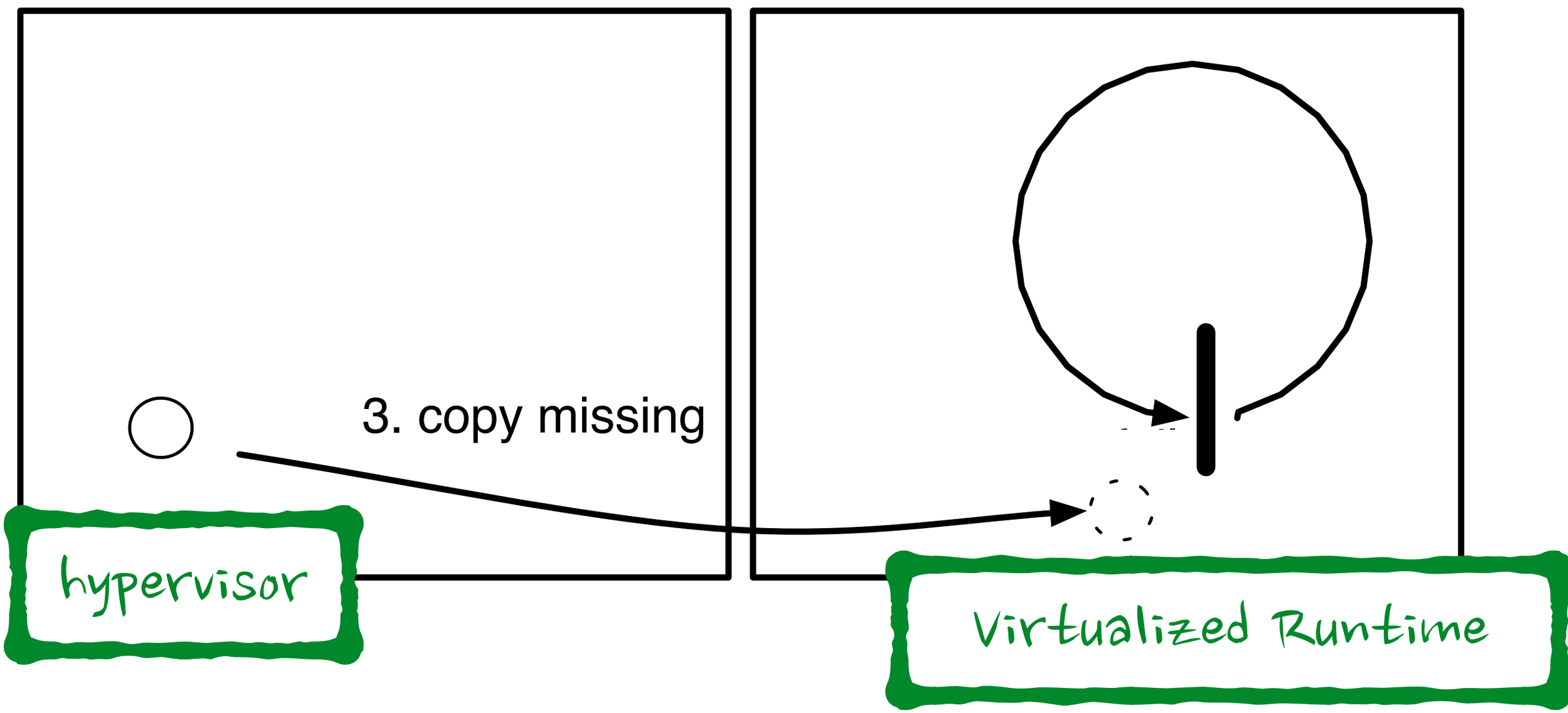
Tornado Run-Fail-Grow



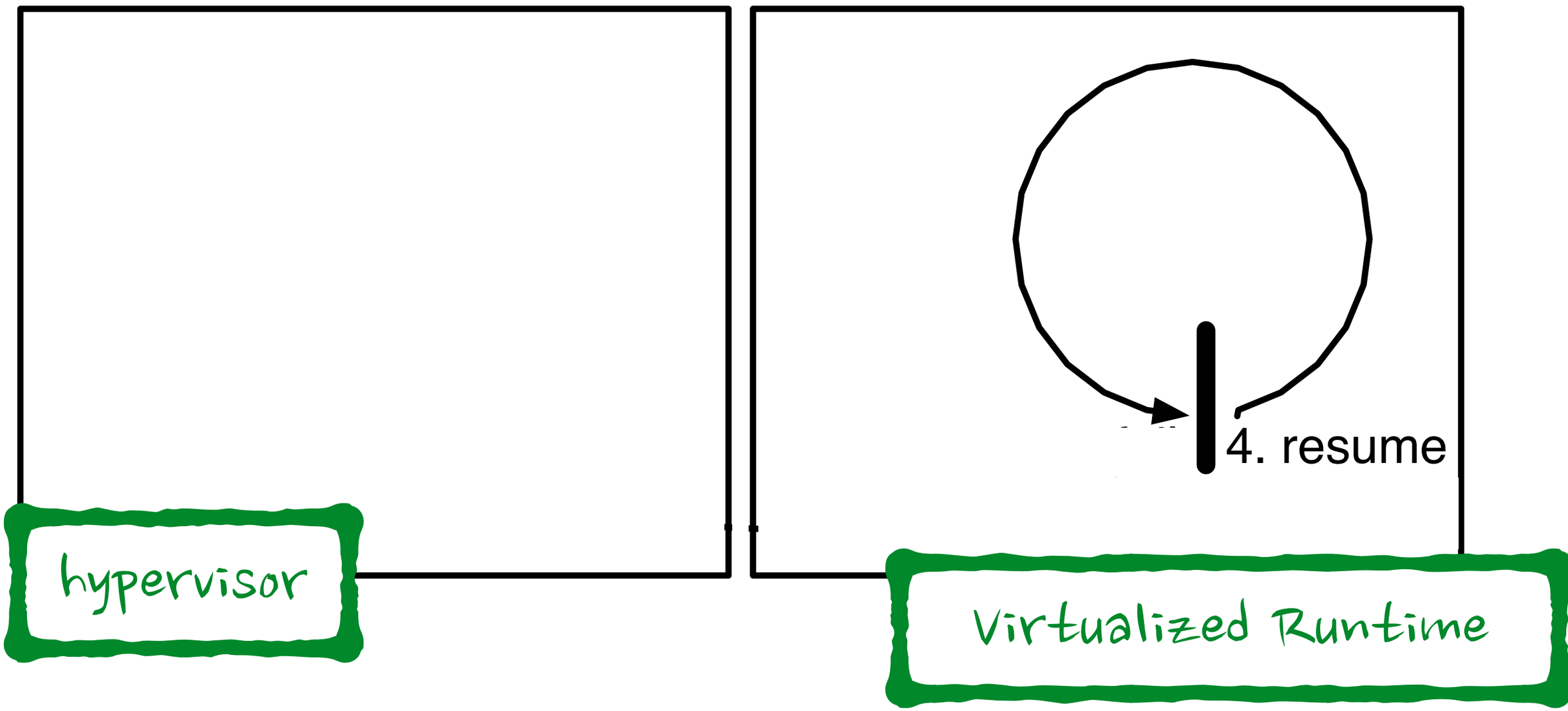
Tornado Run-Fail-Grow



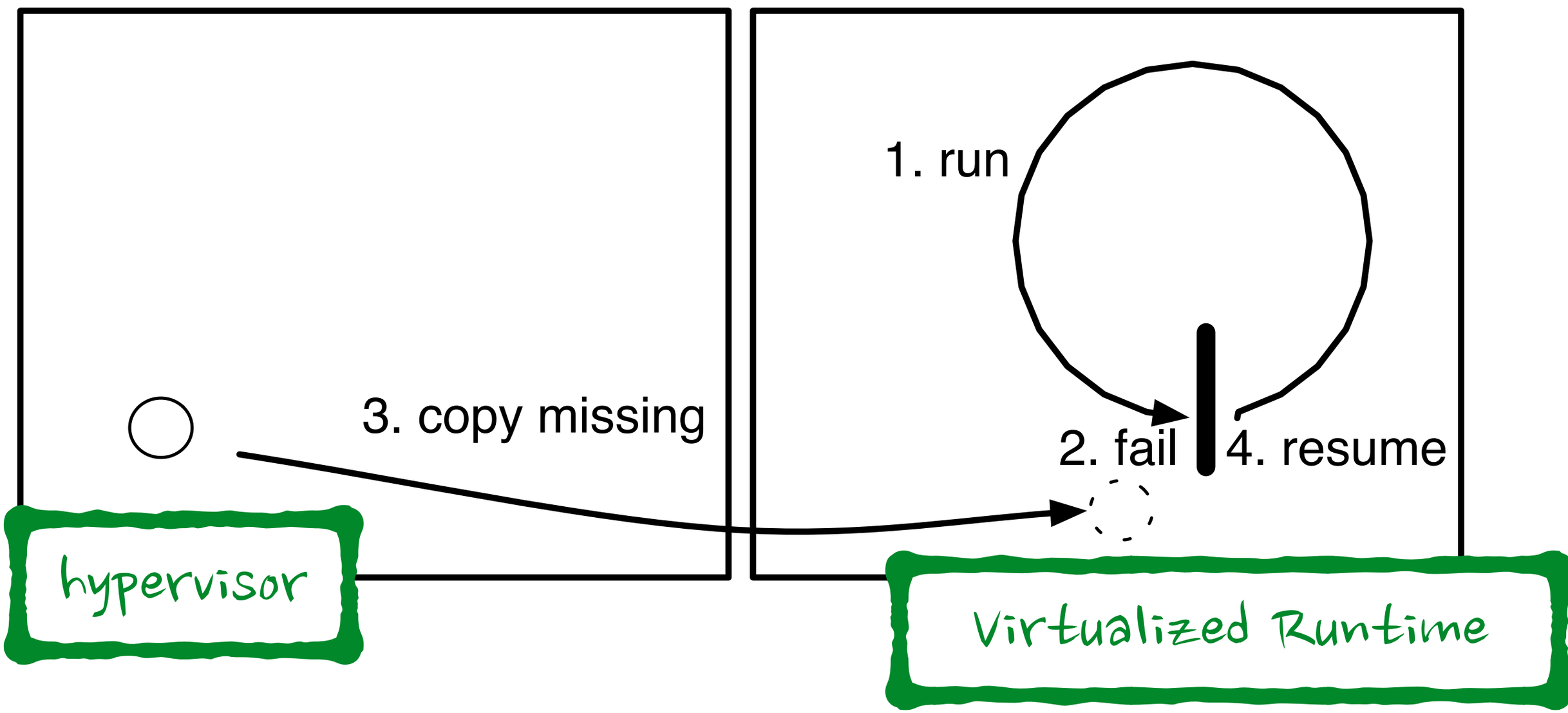
Tornado Run-Fail-Grow



Tornado Run-Fail-Grow



Tornado Run-Fail-Grow



How small can we get?

Experiment	Size (KB)	%Saved
Addition	11	99.99%
Reflective App	32	99.83%
Factorial 100 + I/O	89	99.39%
Seaside Counter	573	96.73%

Comparison with Traditional Tailoring Solutions

criteria	Dedicated Platforms	Static/Hybrid Analysis	Dynamic Analysis	Tornado

Comparison with Traditional Tailoring Solutions

criteria	Dedicated Platforms	Static/Hybrid Analysis	Dynamic Analysis	Tornado
Base libraries				
Third-party libraries				
Legacy Code				
Reflection Support				
General Purpose Infrastructure				
Configurability				
Dynamic Typing				
Minimality				
Completeness				

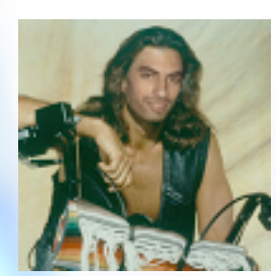
Comparison with Traditional Tailoring Solutions

criteria	Dedicated Platforms	Static/Hybrid Analysis	Dynamic Analysis	Tornado
Base libraries	★	★	★	★
Third-party libraries	☆	★	★	★
Legacy Code	☆	★	½	★
Reflection Support	★	☆	★	★
General Purpose Infrastructure	☆	★	½	★
Configurability	☆	☆	½	★
Dynamic Typing	★	☆	½	★
Minimality	☆	★	★	★
Completeness	★	☆	½	½

Run-Fail-Grow: Creating Tailored Object-Oriented Runtimes

- Run-fail-grow approach for tailoring
- Works on dynamic languages
- Small memory footprint results (10k!)

G. Polito, L. Fabresse, N. Bouraqadi, S. Ducasse,
in Journal of Object Technology



Guille Polito
@guillep
https://guillep.github.io/