Rubah
DSU for Java on a Stock JVM

Luís Pina[1]
luis.pina@tecnico.ulisboa.pt

Luís Veiga[1]
luis.veiga@inesc-id.pt

Michael Hicks[2]
mwh@cs.umd.edu

[1]INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal

[2]University of Maryland
College Park, MD, USA

October 22nd, 2014

OOPSLA '14

Software Updates are necessary

- New features
- Fix bugs

Software Updates are necessary

- New features
- Fix bugs

Software Updates are necessary

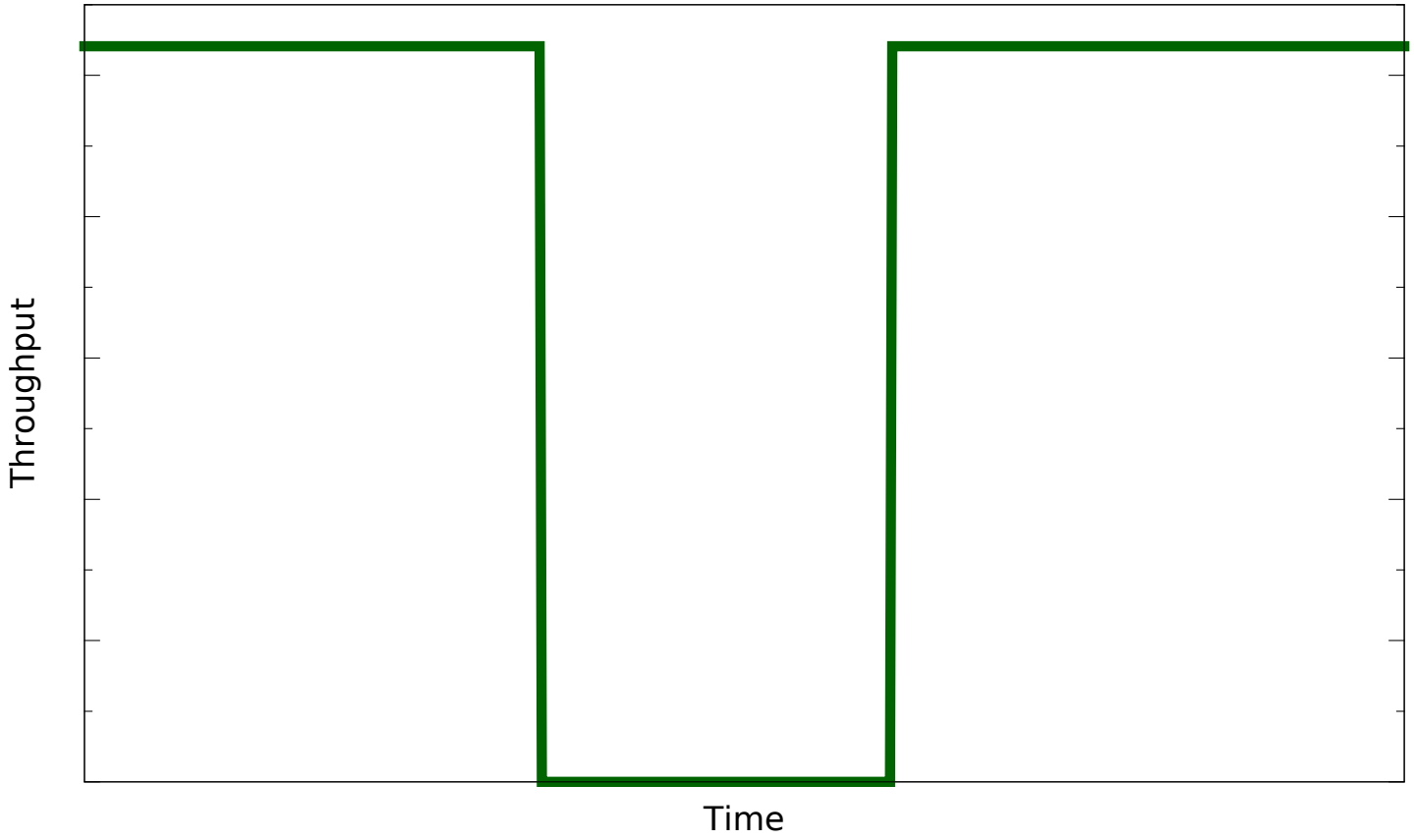
- New features
- Fix bugs

NASDAQ®

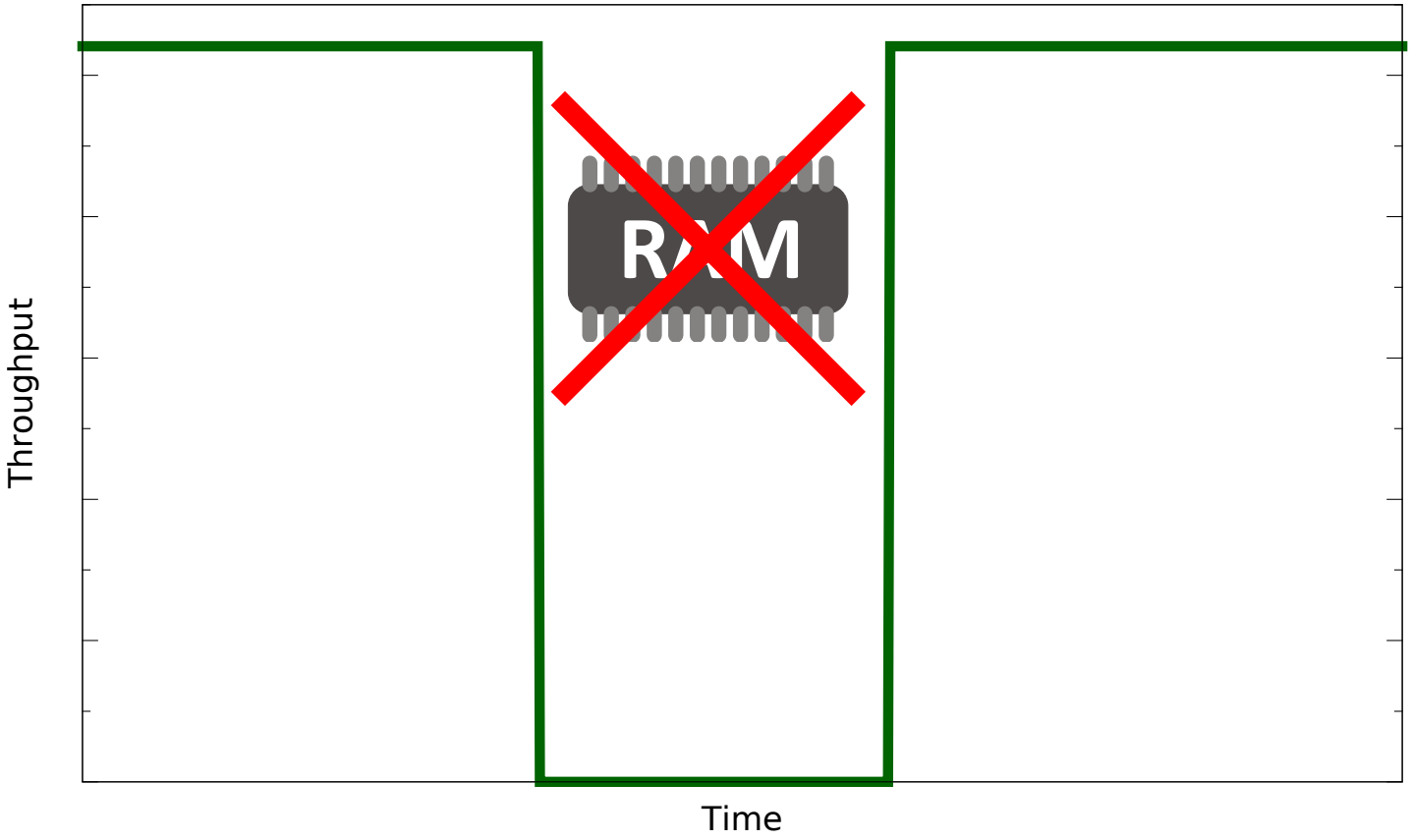
- 2010 cyber-attack
- Software left outdated

Software Updates are necessary and disruptive

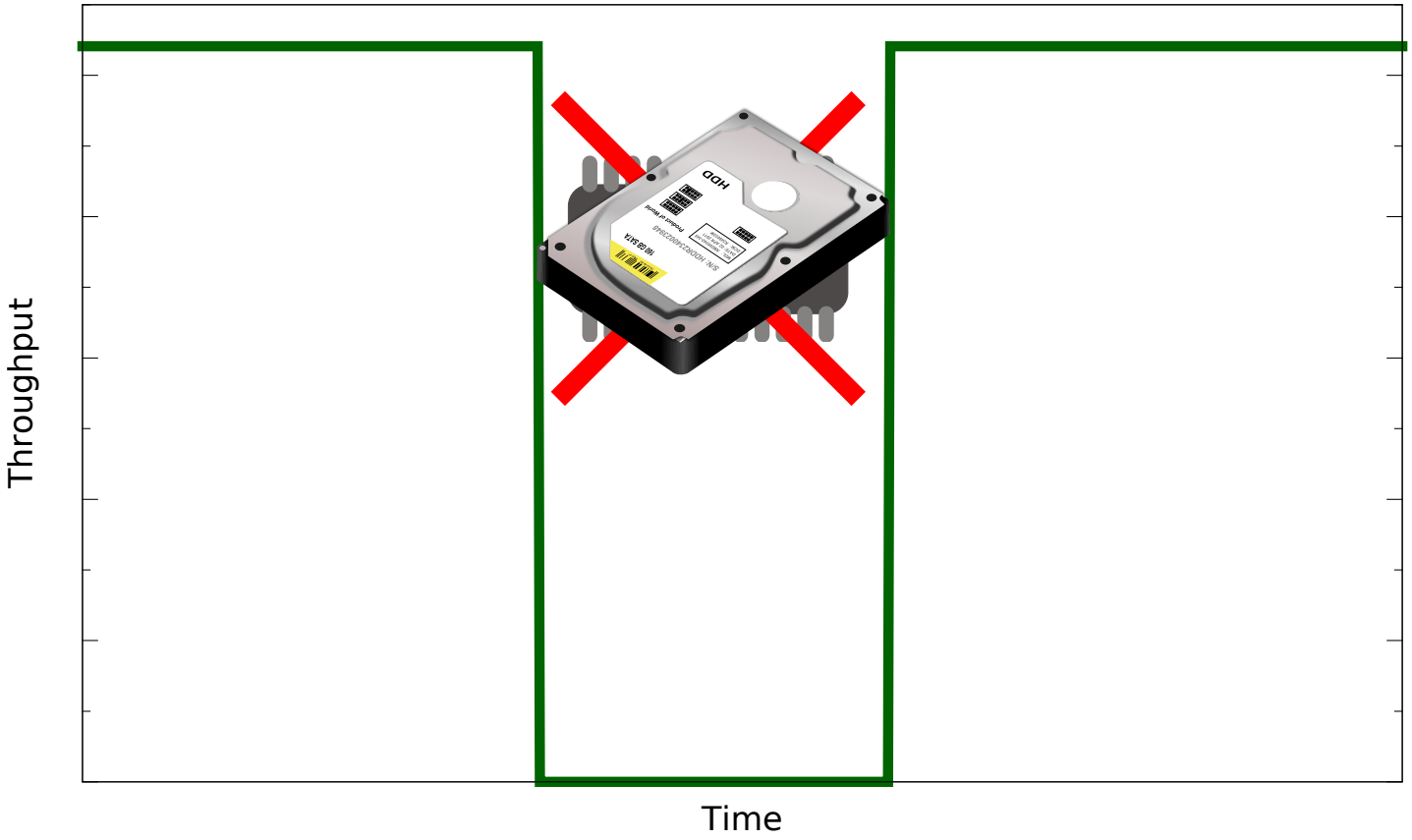
Software Updates are necessary and disruptive



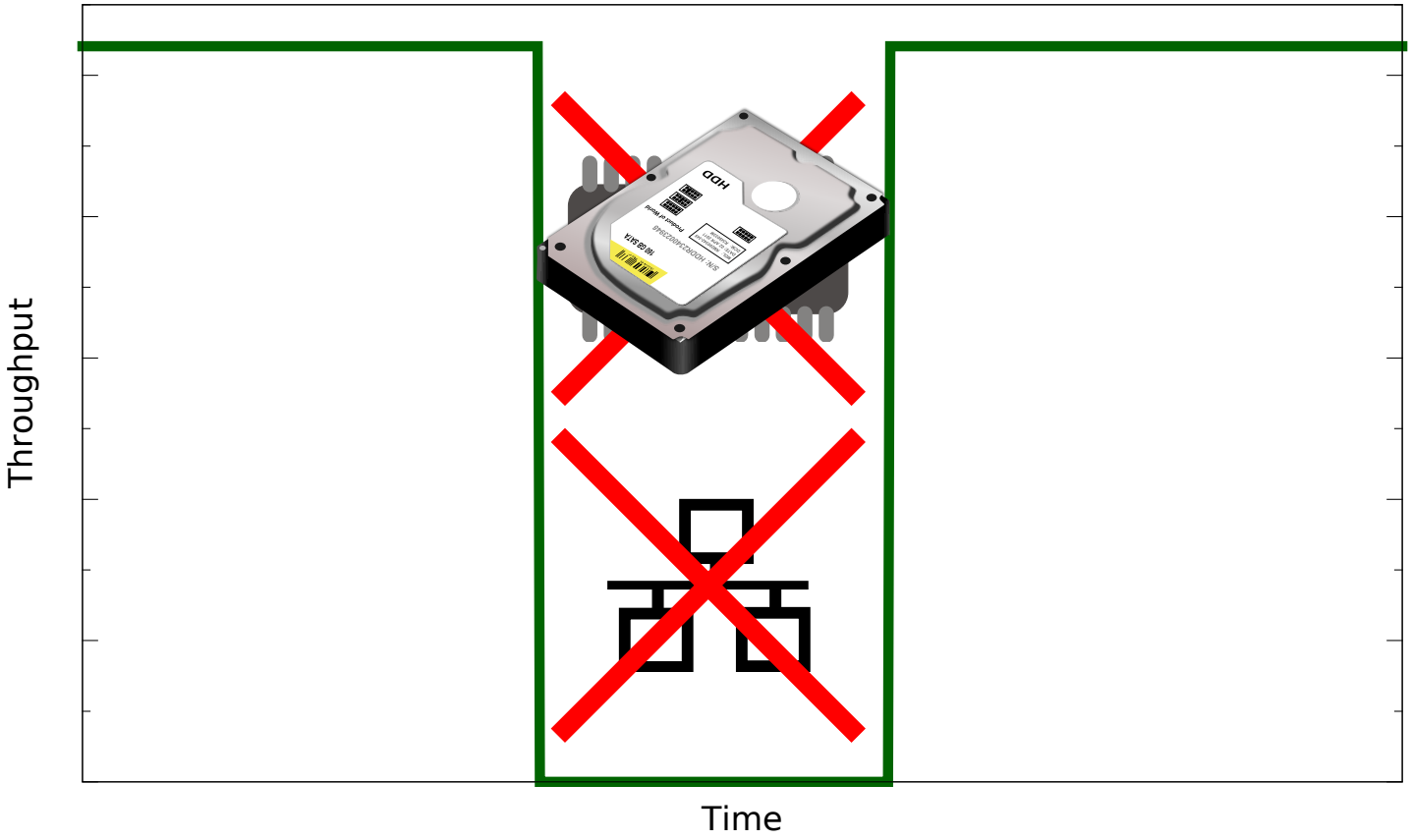
Software Updates are necessary and disruptive



Software Updates are necessary and disruptive

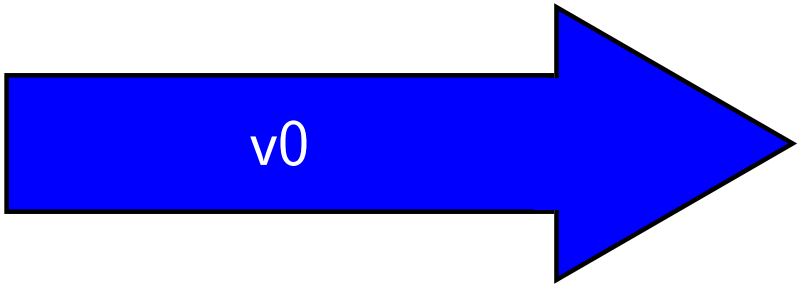


Software Updates are necessary and disruptive

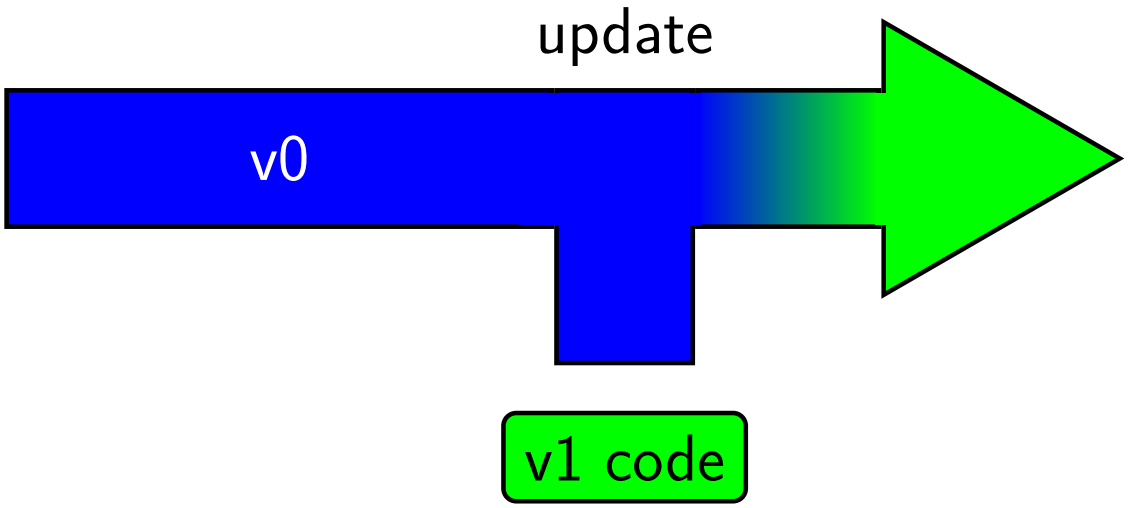


Dynamic Software Updating

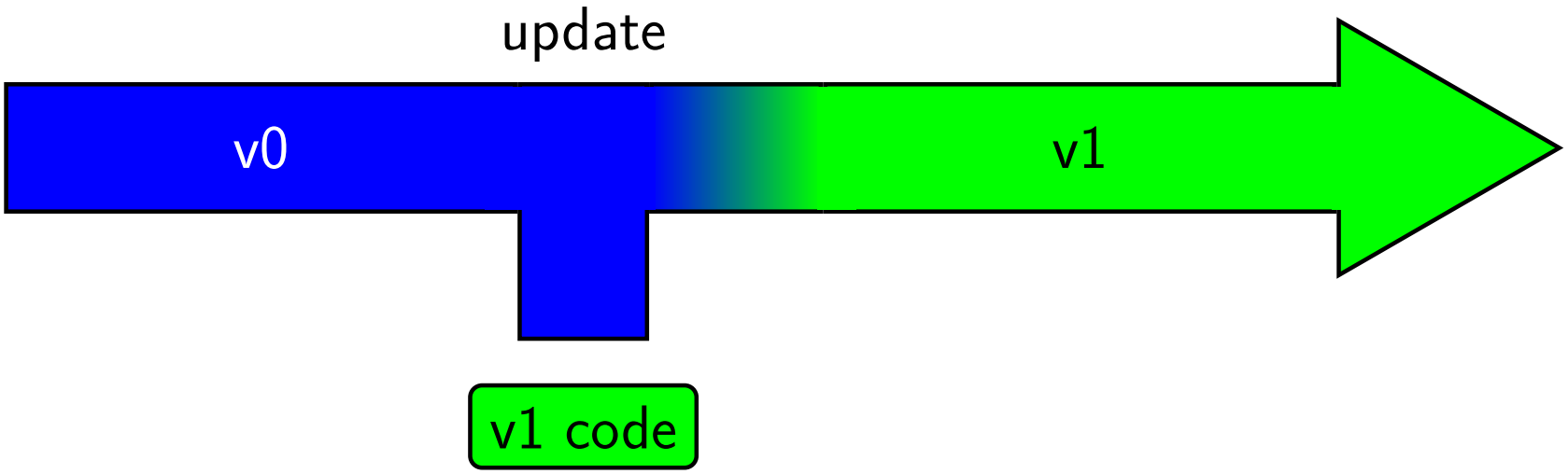
Dynamic Software Updating



Dynamic Software Updating



Dynamic Software Updating



Existing DSU for Java

Efficient?	Stock JVM? No	Stock JVM? yes
No	JDrums	DUSC, DuSTM
Yes	JVolve, DCE-VM	**Rubah**

Rubah

- Efficient / Non-disruptive
- Stock JVM
- Flexible

Rubah

- Efficient / Non-disruptive
  - No steady-state overhead
  - Parallel/lazy algorithms
- Stock JVM
- Flexible

Rubah

- Efficient / Non-disruptive
  - No steady-state overhead
  - Parallel/lazy algorithms
- Stock JVM
  - Optimizations rely on internals
    - Unsafe memory access, object memory layout
    - Tested on Oracle HotSpot
    - Adapatable to different JVMs (OpenJDK, IBM Jikes)
- Flexible

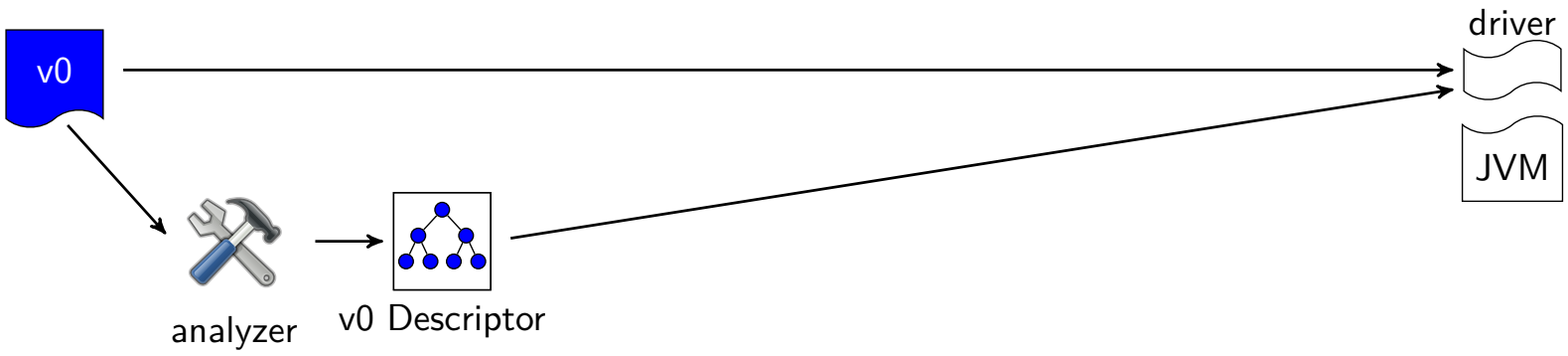
Rubah

- Efficient / Non-disruptive
  - No steady-state overhead
  - Parallel/lazy algorithms

- Stock JVM
  - Optimizations rely on internals
    - Unsafe memory access, object memory layout
    - Tested on Oracle HotSpot
    - Adapatable to different JVMs (OpenJDK, IBM Jikes)

- Flexible
  - Unrestricted class updates
  - 13 versions of 5 real-world applications
    - **H2**
    - **Voldemort**
    - **CrossFTP**
    - **Jake2**
    - **Java Email Server**

Using Rubah



Rubah Driver

- Loads the program

Rubah Driver

- Loads the program

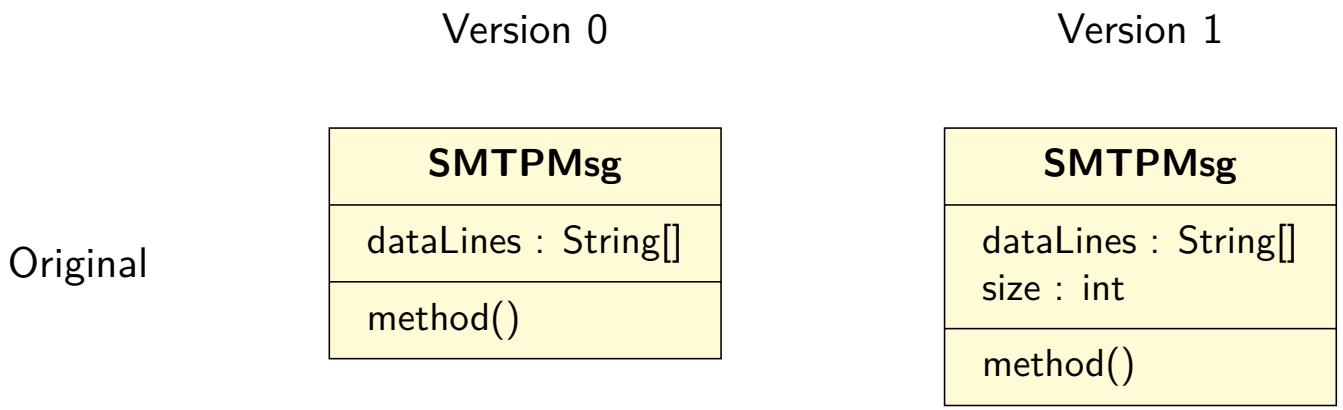
```
java <main-class> <args>
```

becomes

```
java rubah.Rubah <descriptor> <main-class> <args>
```

Rubah Driver

- Loads the program
- Rewrites bytecode



Rubah Driver

- Loads the program
- Rewrites bytecode

	Version 0	Version 1
Original	**SMTPMsg** dataLines : String[] method()	**SMTPMsg** dataLines : String[] size : int method()
Rewritten	**SMTPMsg___0** dataLines : String[] method() GETdataLines SETdataLines	**SMTPMsg___1** dataLines : String[] size : int method() GETdataLines SETdataLines GETsize SETsize

Rubah Driver

- Loads the program
- Rewrites bytecode

```
java <main-class> <args>
```

VS

```
java rubah.Rubah <descriptor> <main-class> <args>
```

Rubah Driver

- Loads the program
- Rewrites bytecode

```
java <main-class> <args>
```

VS

```
java rubah.Rubah <descriptor> <main-class> <args>
```

Overhead = [-2.5% ; 2.5%]

Rubah Driver

- Loads the program
- Rewrites bytecode
- **No extra overhead!**

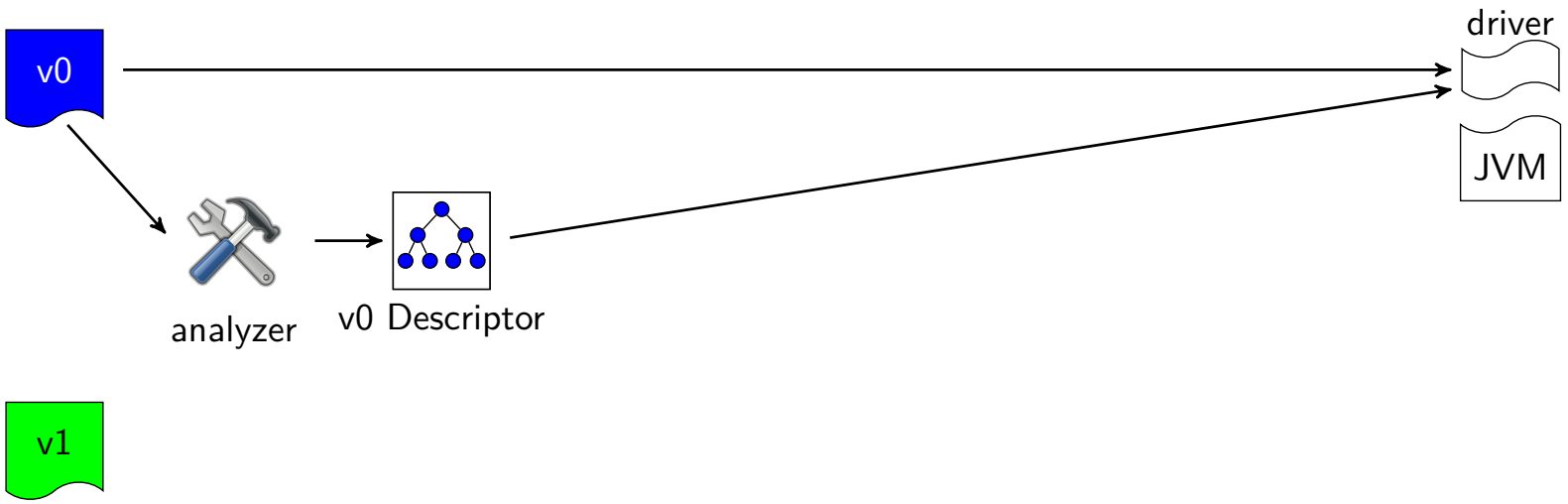
```
java <main-class> <args>
```

VS

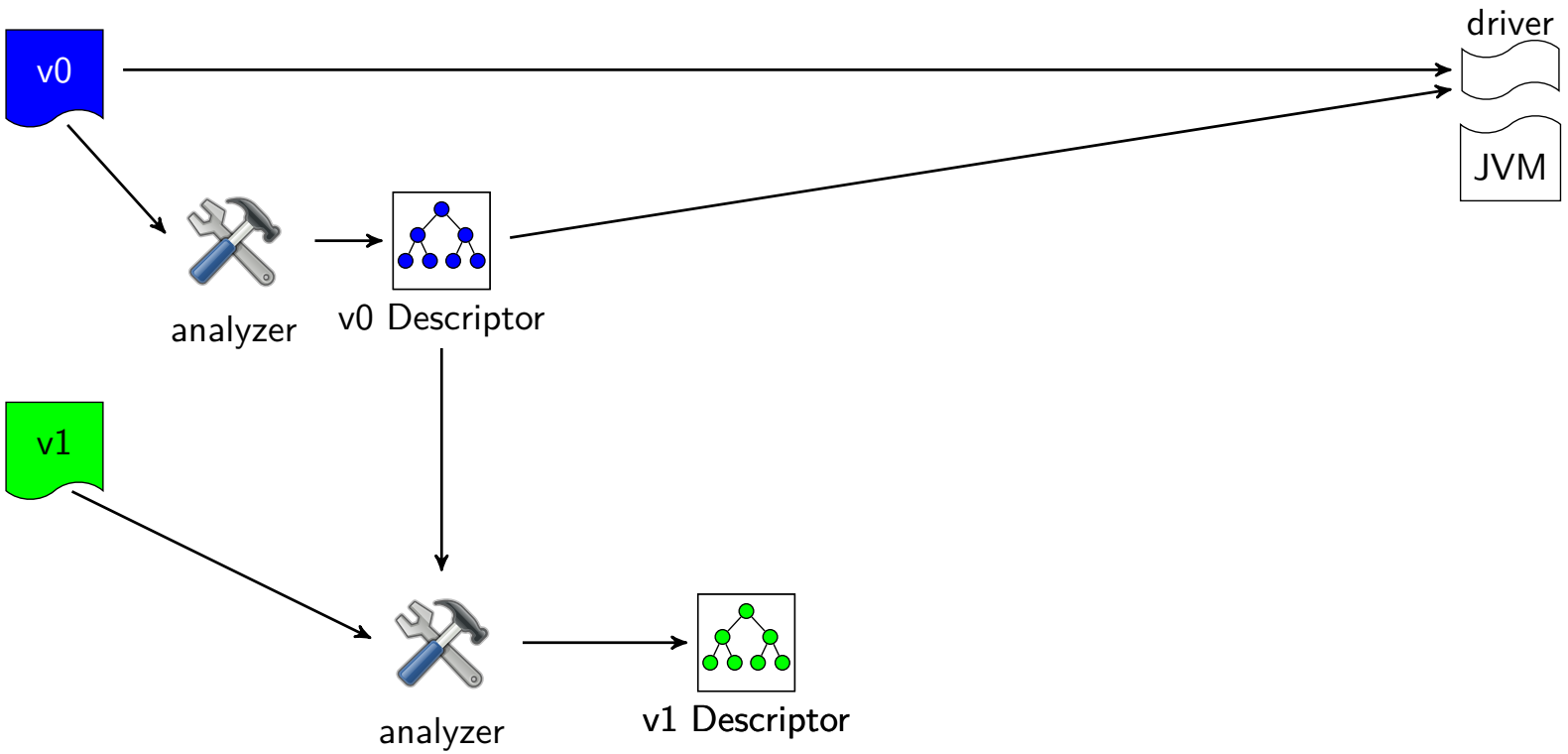
```
java rubah.Rubah <descriptor> <main-class> <args>
```

Overhead = [-2.5% ; 2.5%]

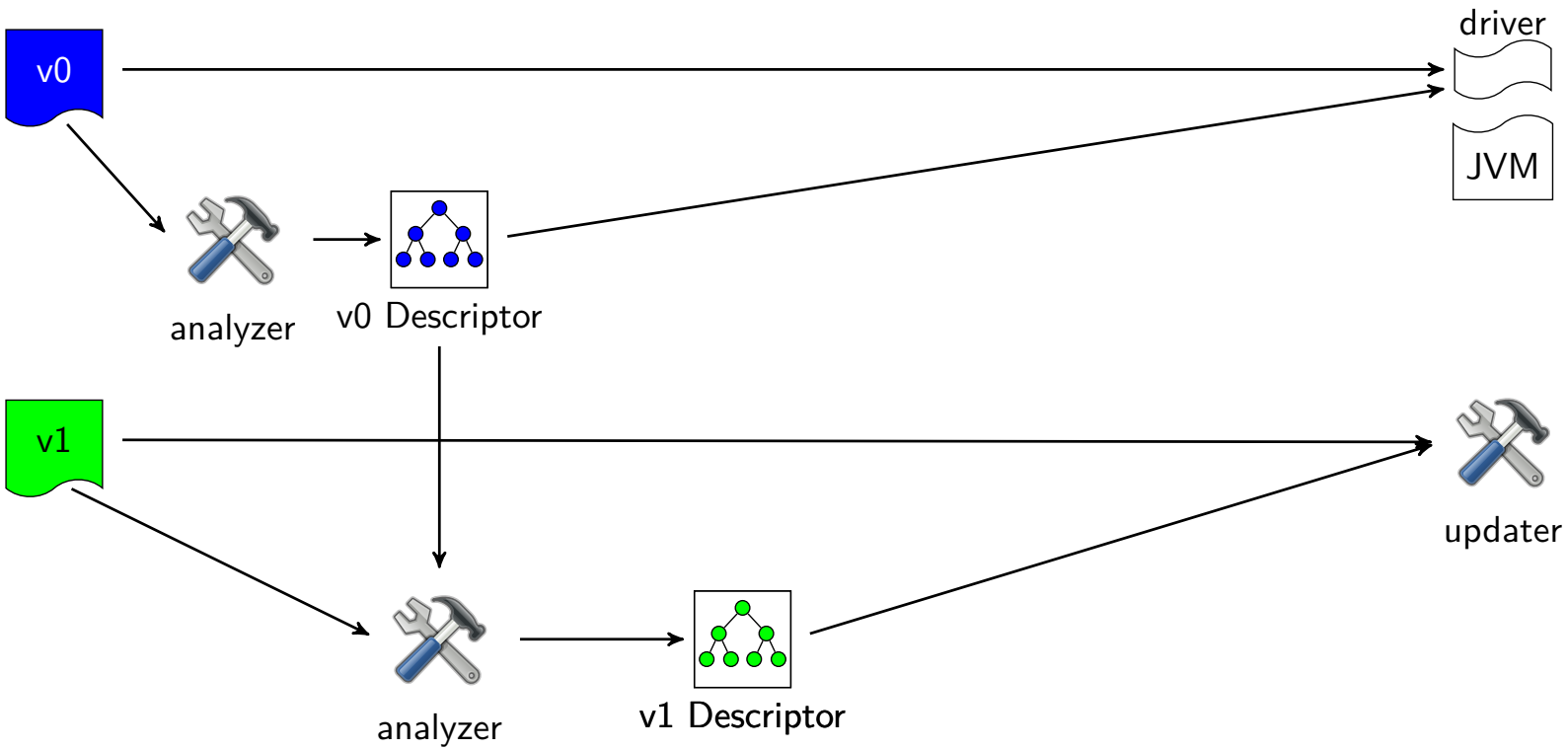
Using Rubah



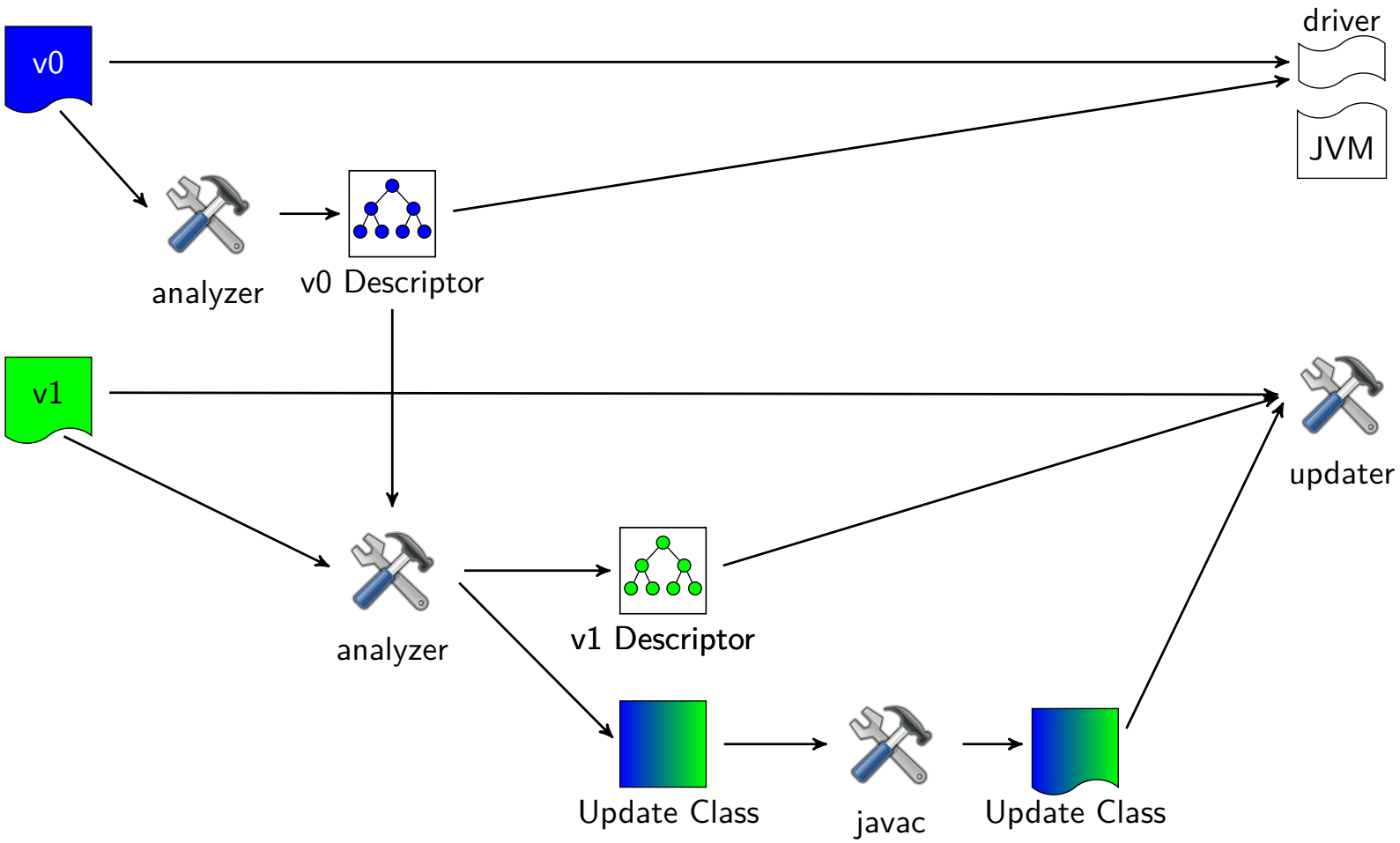
Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
analyzer
v1 Descriptor

Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
updater
analyzer
v1 Descriptor

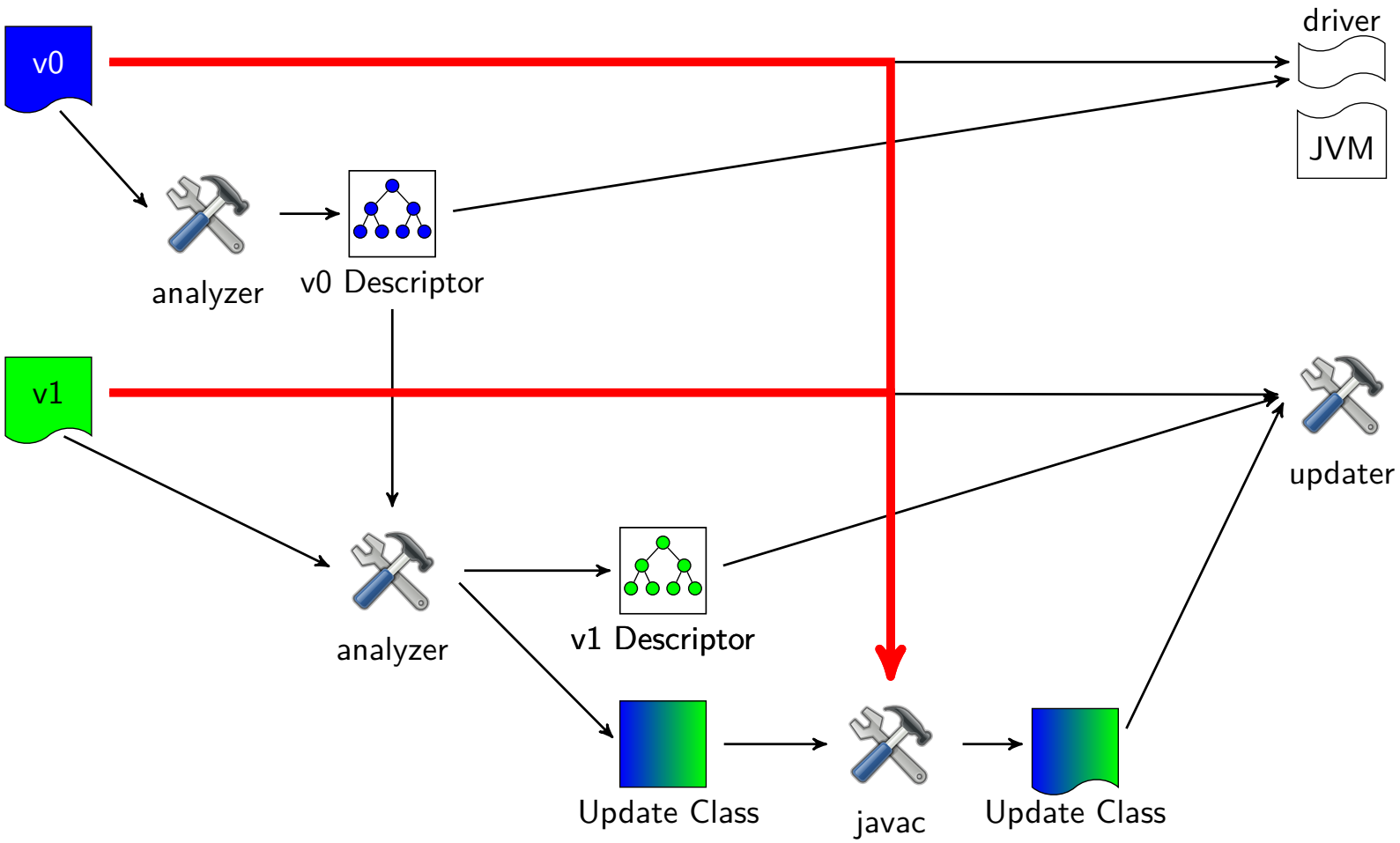
Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
updater
analyzer
v1 Descriptor
Update Class
javac
Update Class

Convert method

```
01
02
03      void convert( ???  o0, ???  o1) {
04
05
06
07
08      }
09
```

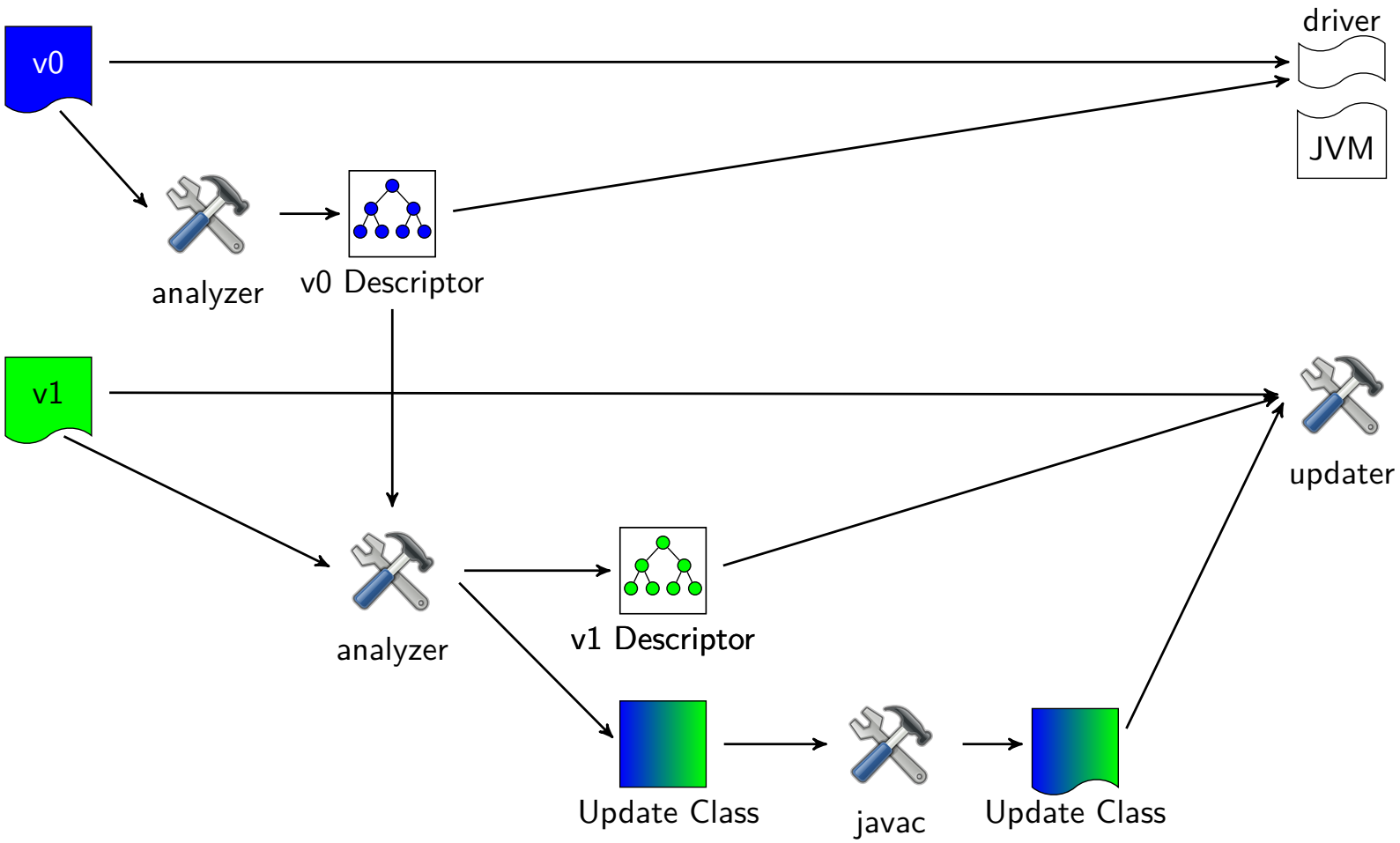
Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
updater
analyzer
v1 Descriptor
Update Class
javac
Update Class

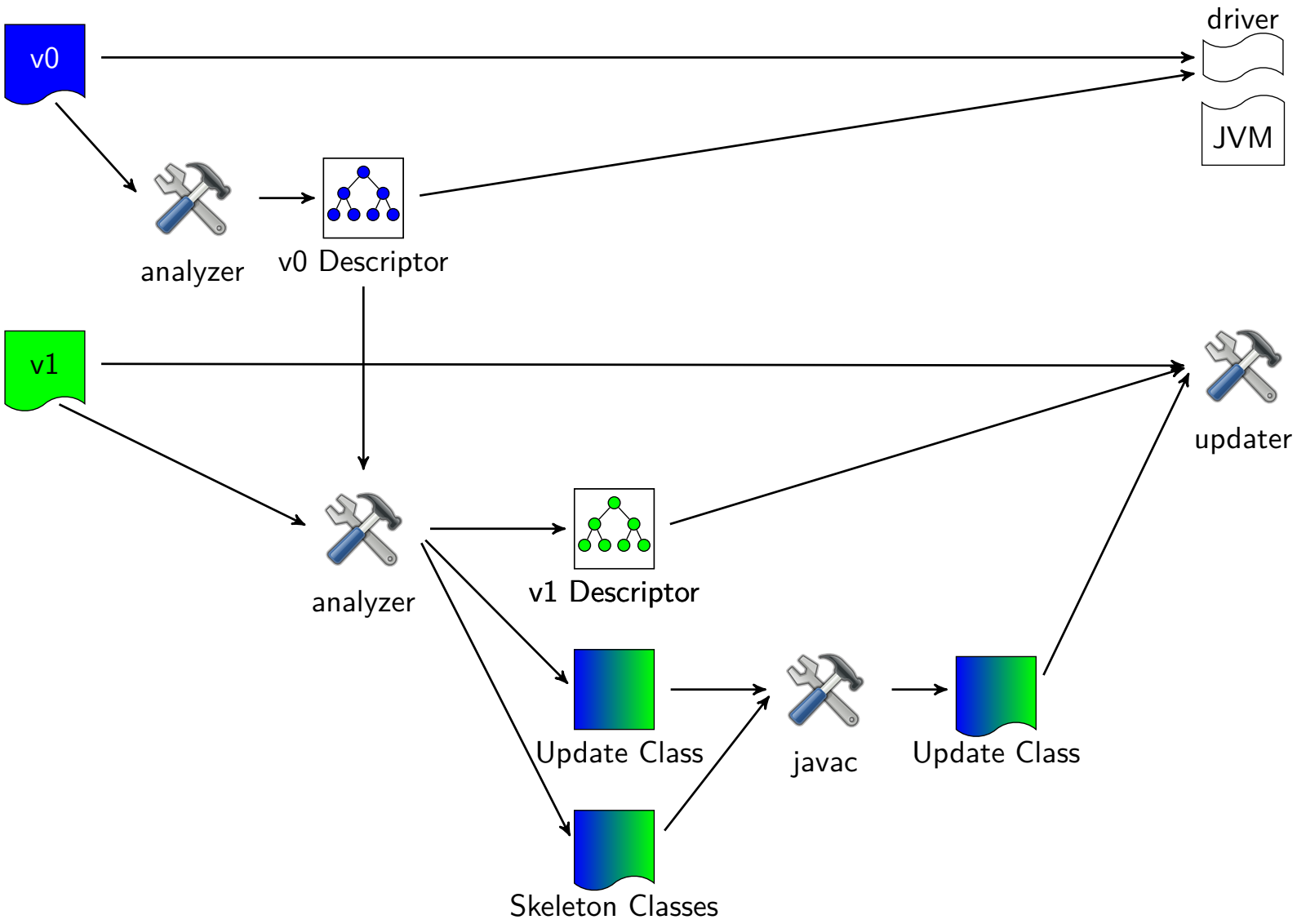
Convert method

```
01
02
03      void convert( SMTPMsg  o0, SMTPMsg  o1) {
04
05
06
07
08      }
09
```

Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
updater
analyzer
v1 Descriptor
Update Class
javac
Update Class

Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
updater
analyzer
v1 Descriptor
Update Class
javac
Update Class
Skeleton Classes

Convert method

```
01
02
03      void convert( v0. SMTPMsg o0, v1. SMTPMsg o1) {
04
05
06
07
08      }
09
```

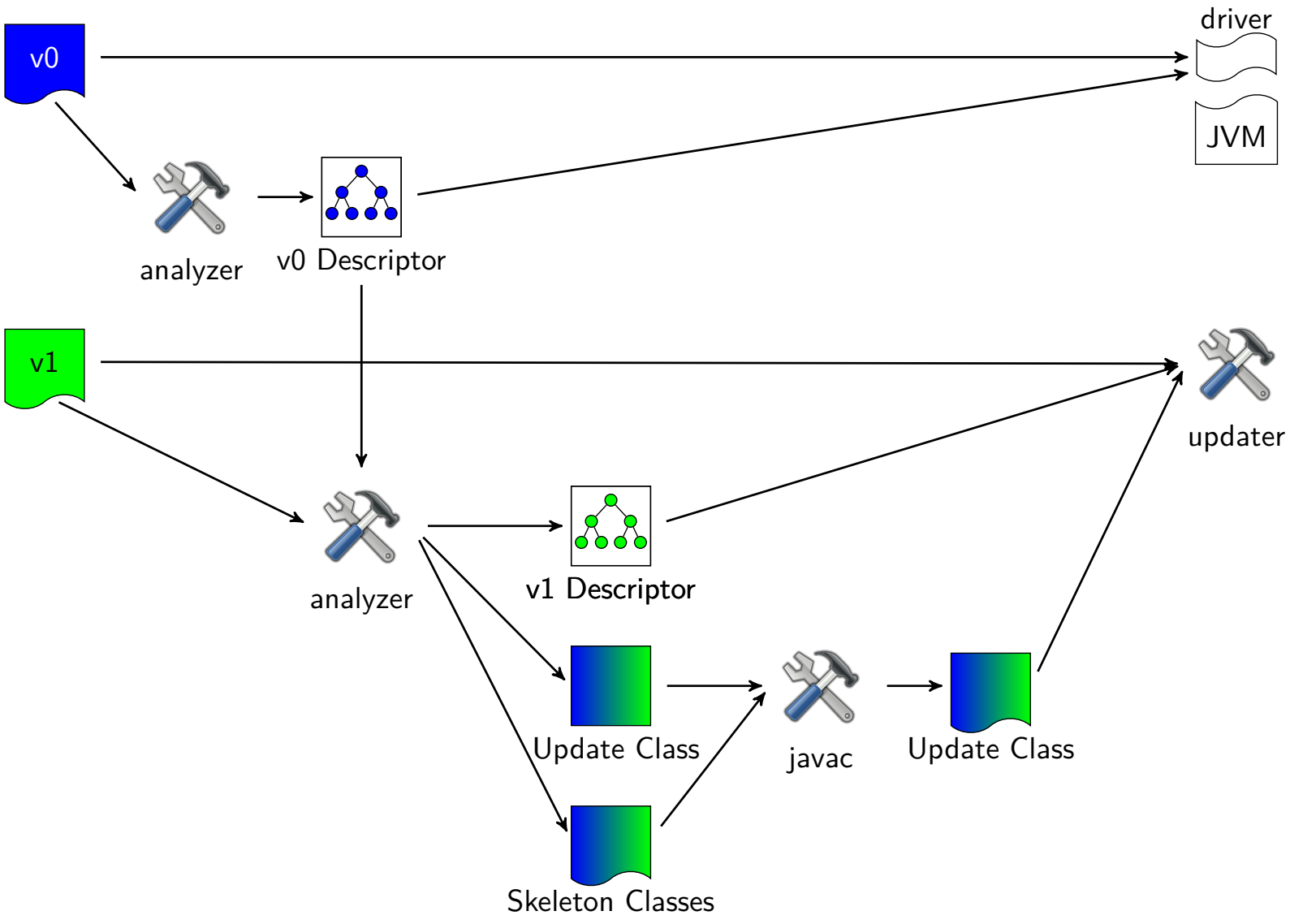
Convert method

```
01 class UpdateClass {
02
03     void convert(v0.SMTPMsg o0,v1.SMTPMsg o1) {
04         //o1.dataLines = o0.dataLines;
05         o1.size = 0;
06
07
08     }
09 }
```

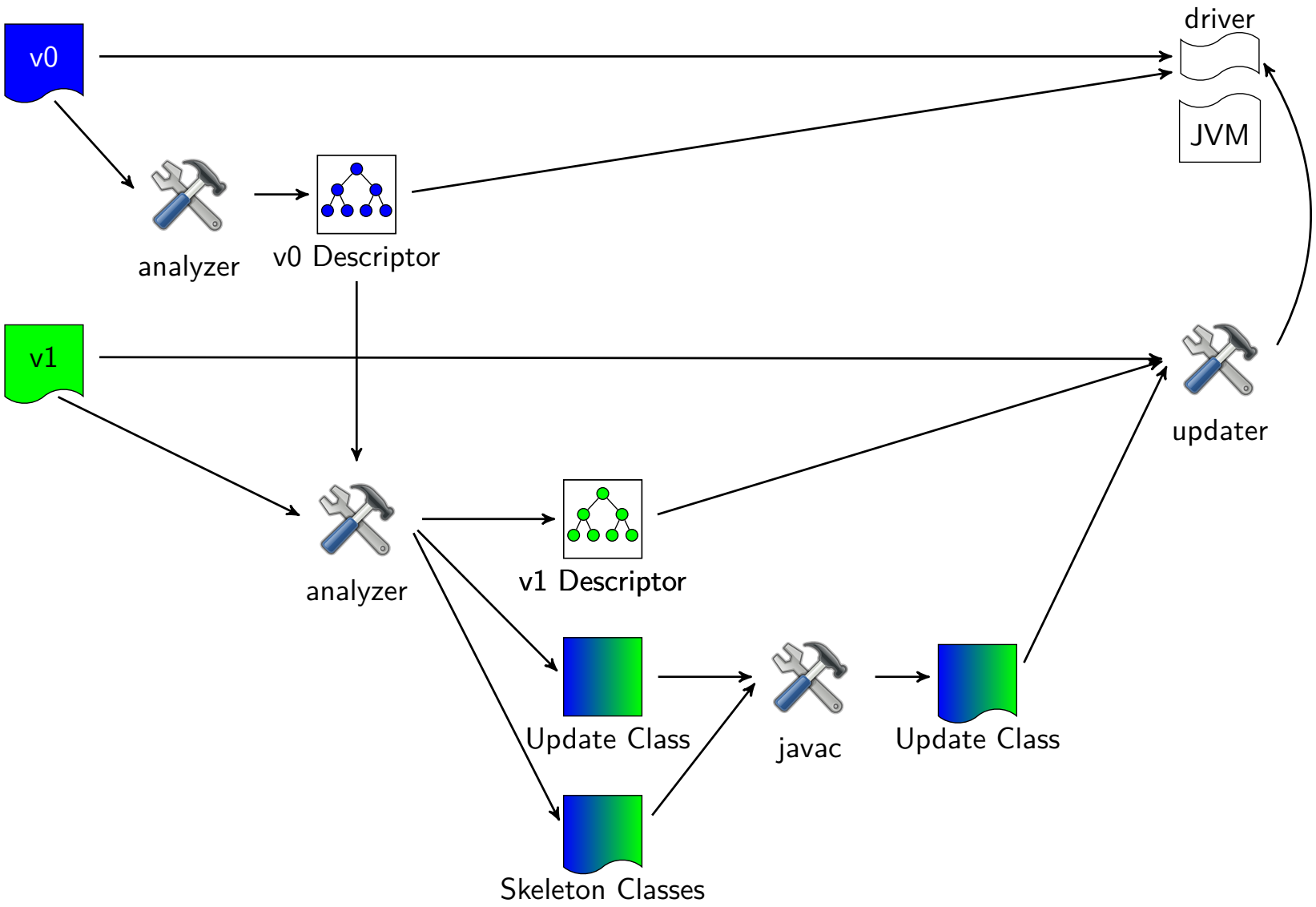
Convert method

```
01 class UpdateClass {
02
03     void convert(v0.SMTPMsg o0,v1.SMTPMsg o1) {
04         //o1.dataLines = o0.dataLines;
05         o1.size = 0;
06         for (String s : o0.dataLines)
07           o1.size += s.length();
08     }
09 }
```

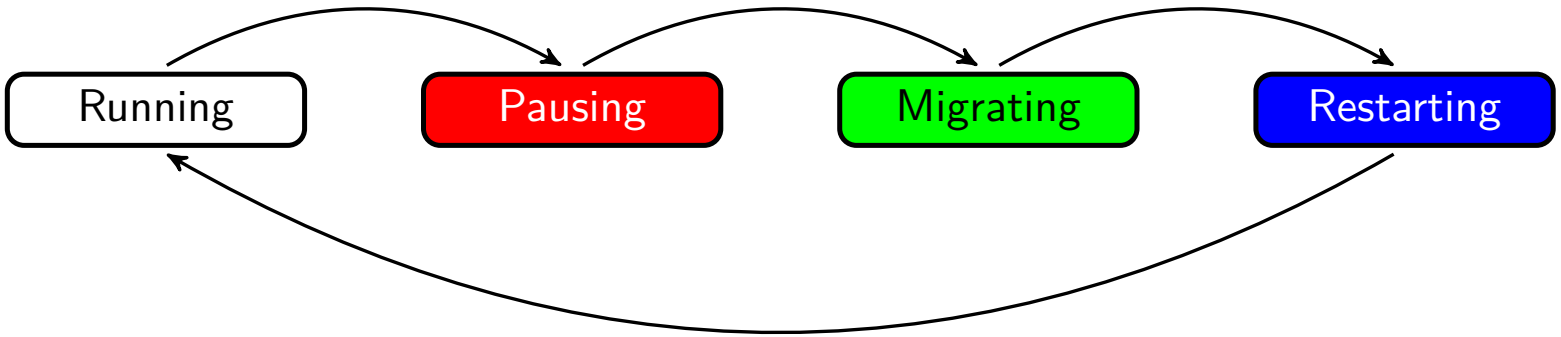
Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
updater
analyzer
v1 Descriptor
Update Class
javac
Update Class
Skeleton Classes

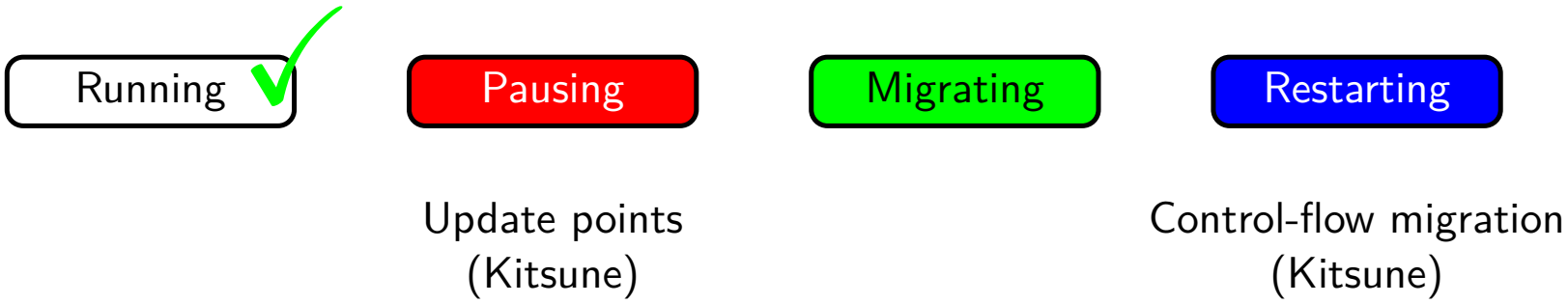
Using Rubah


v0
driver
JVM
analyzer
v0 Descriptor
v1
updater
analyzer
v1 Descriptor
Update Class
javac
Update Class
Skeleton Classes

Installing an update



Installing an update



Retrofitting

Changes

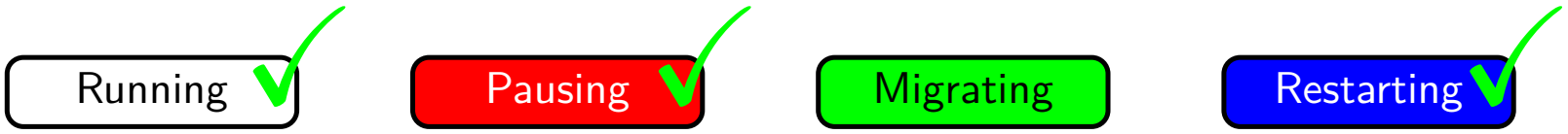
- Update points
  - Top of long running loops
  - Stops thread when update available
- Control-flow migration
  - Transfer control between update points in different versions
- Blocking I/O
  - Use non-blocking I/O or pooling
  - Rubah API

Retrofitting

Effort

Benchmark	Original LOC	Modified	
		LOC	*%*
H2	40119	267	0.67
Voldemort	87516	175	0.19
Jake2	85408	29	0.03
CrossFTP	18221	224	1.23
Java Email Server	2368	183	7.72

Installing an update



Update class
Parallel/lazy migration

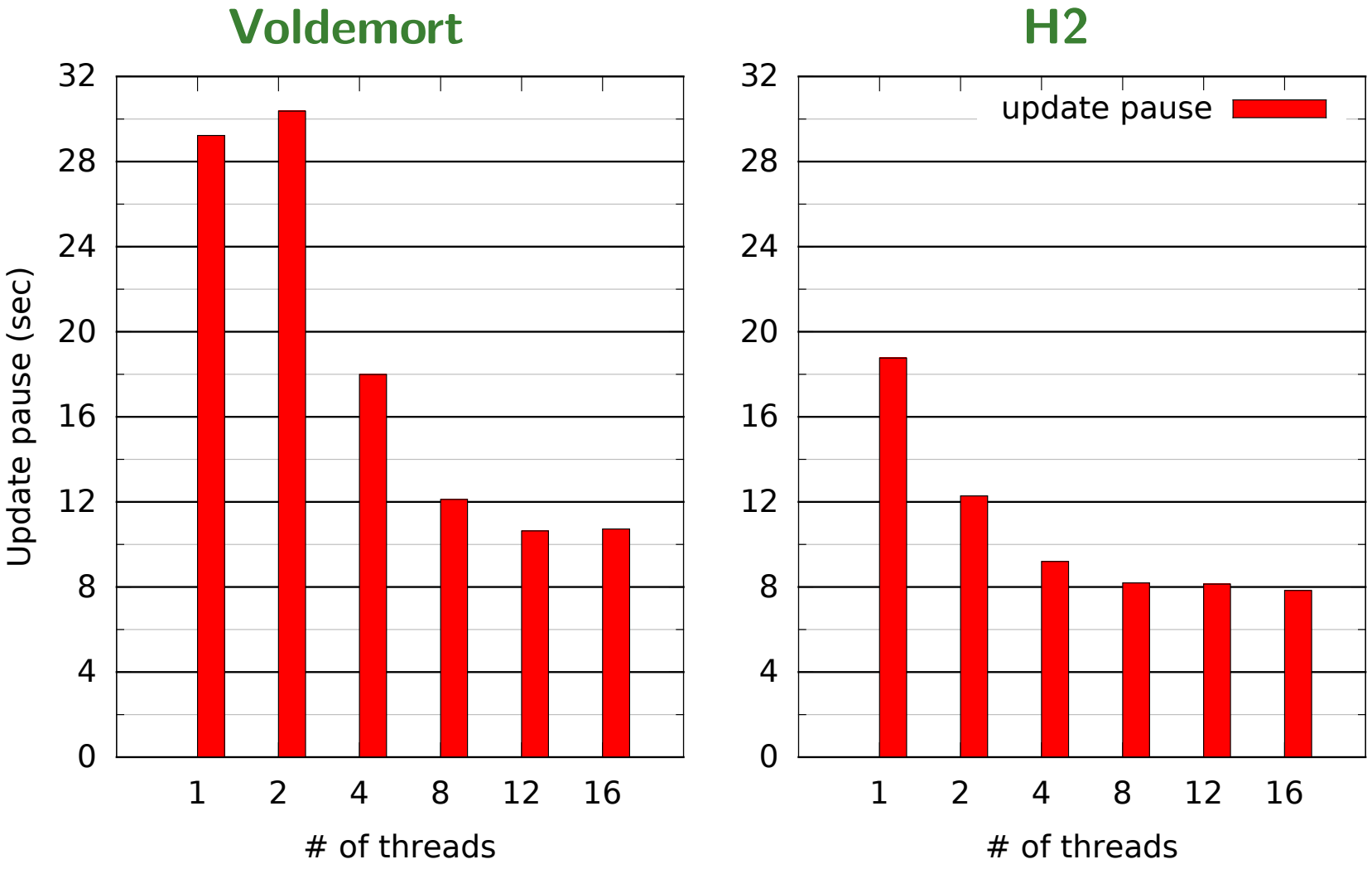
Migrating the state

Parallel migration

- Traverse heap with multiple threads
- Share work with task queue

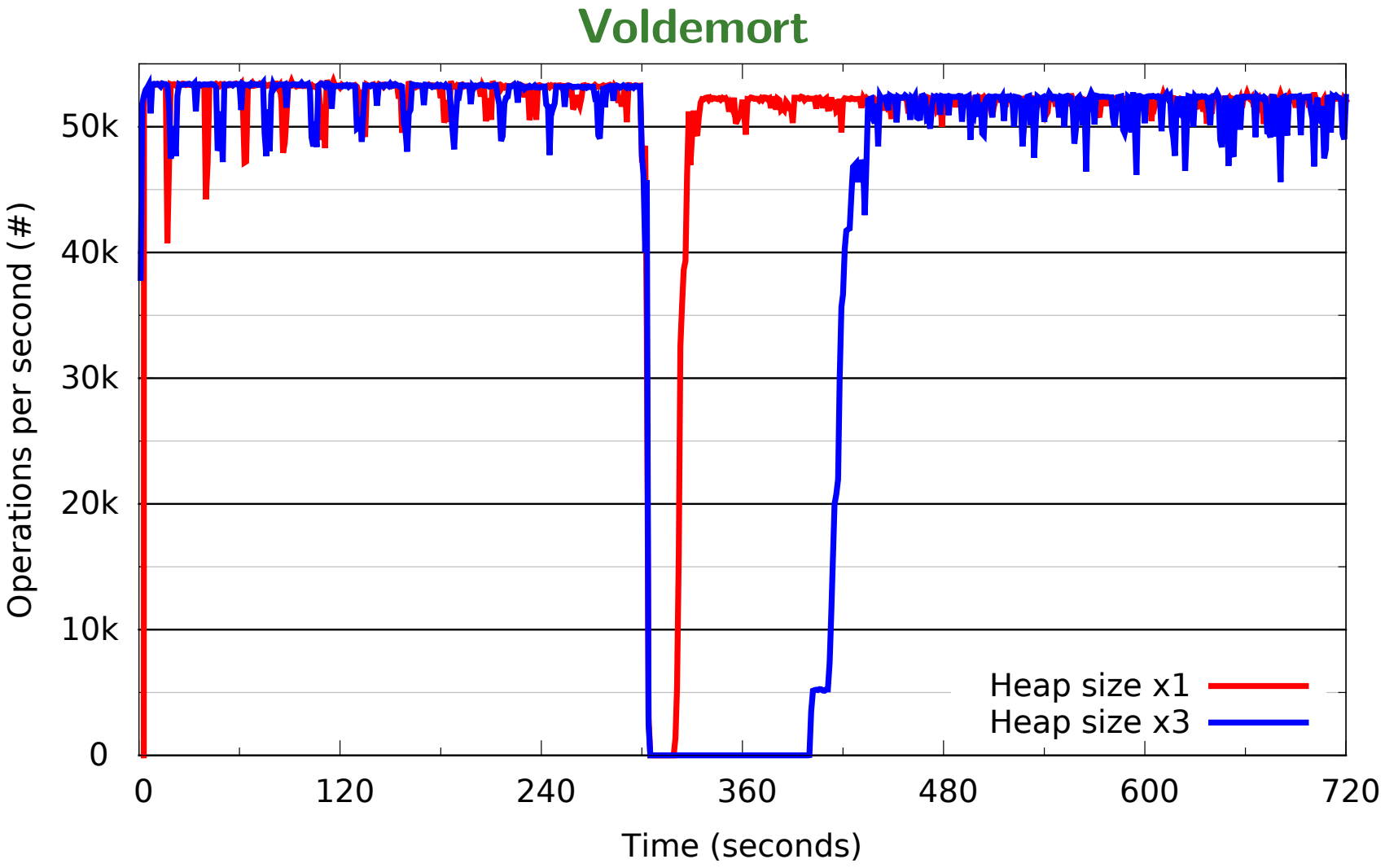
Migrating the state

Parallel migration



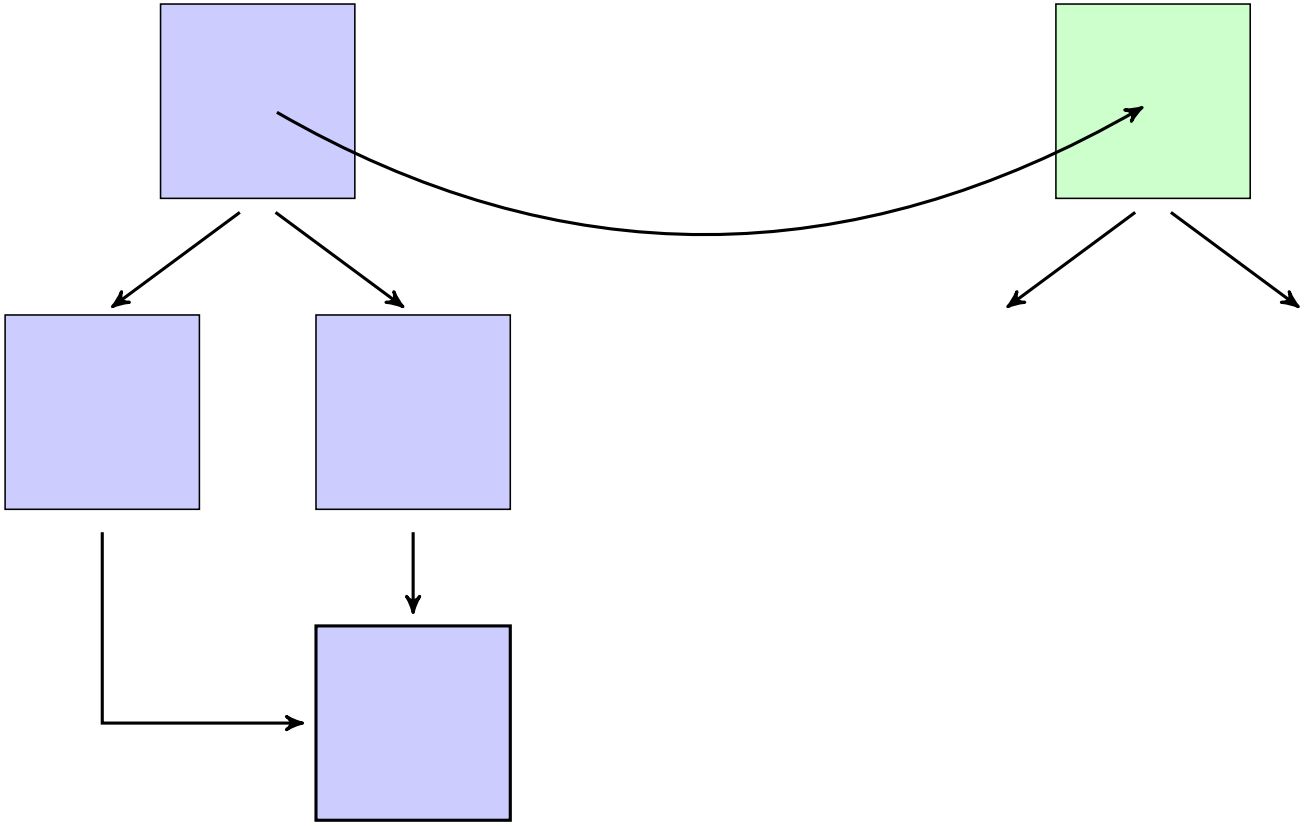
Migrating the state

Parallel migration



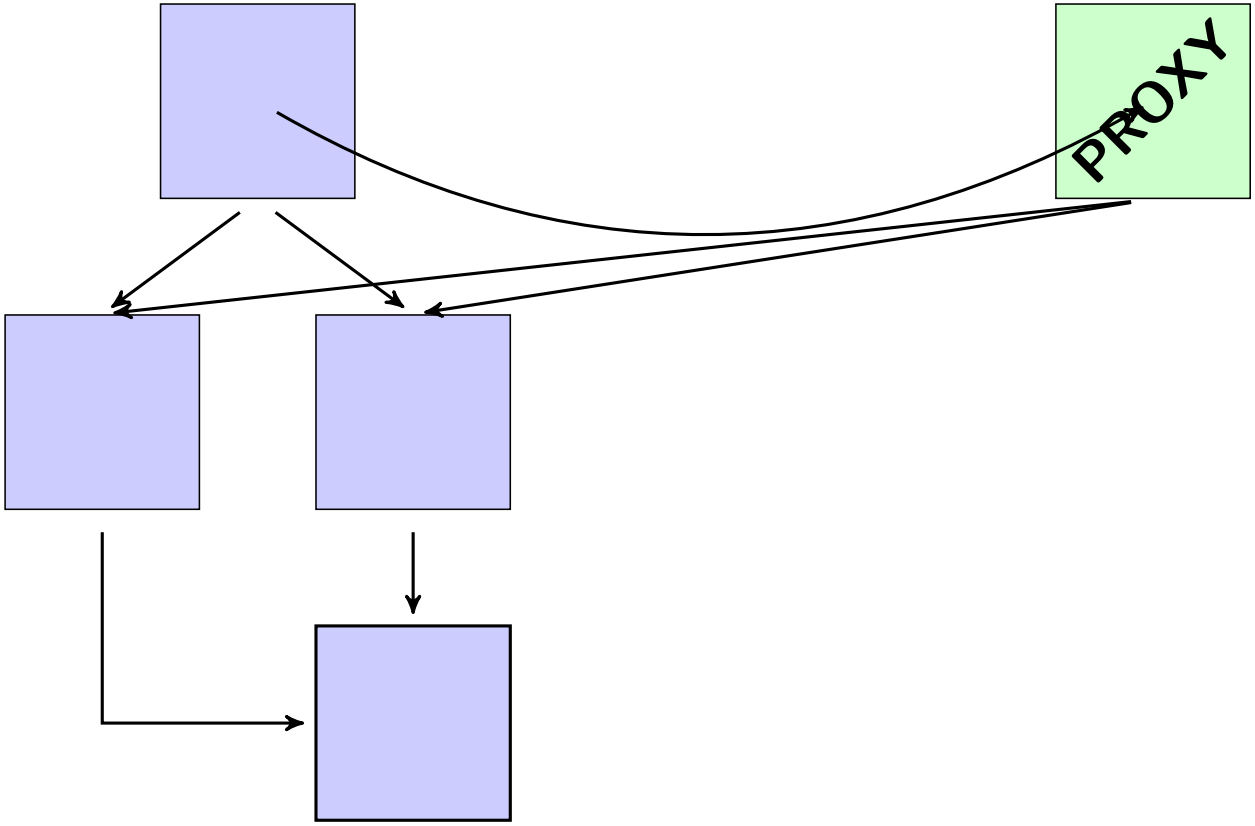
Migrating the state

Lazy migration



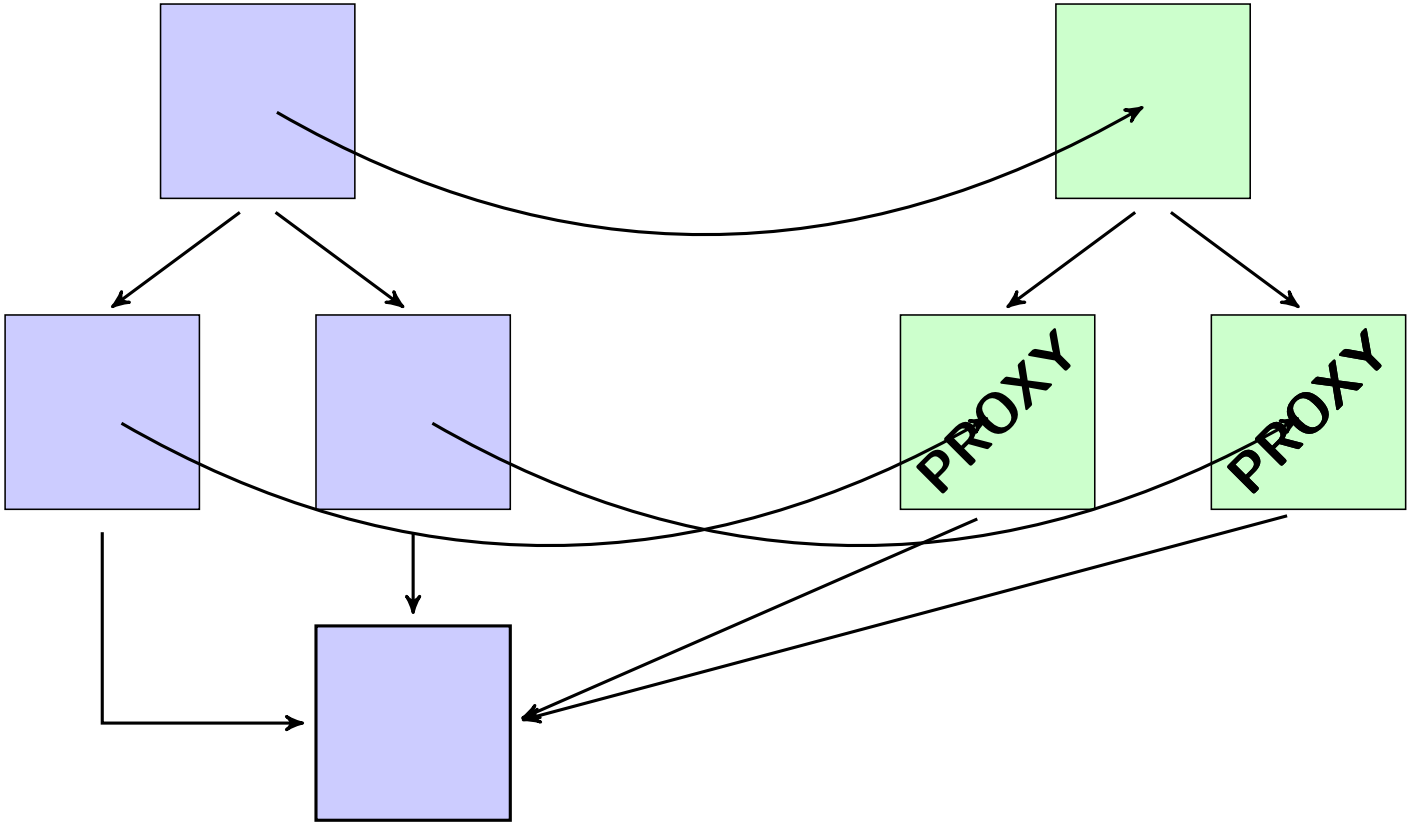
Migrating the state

Lazy migration



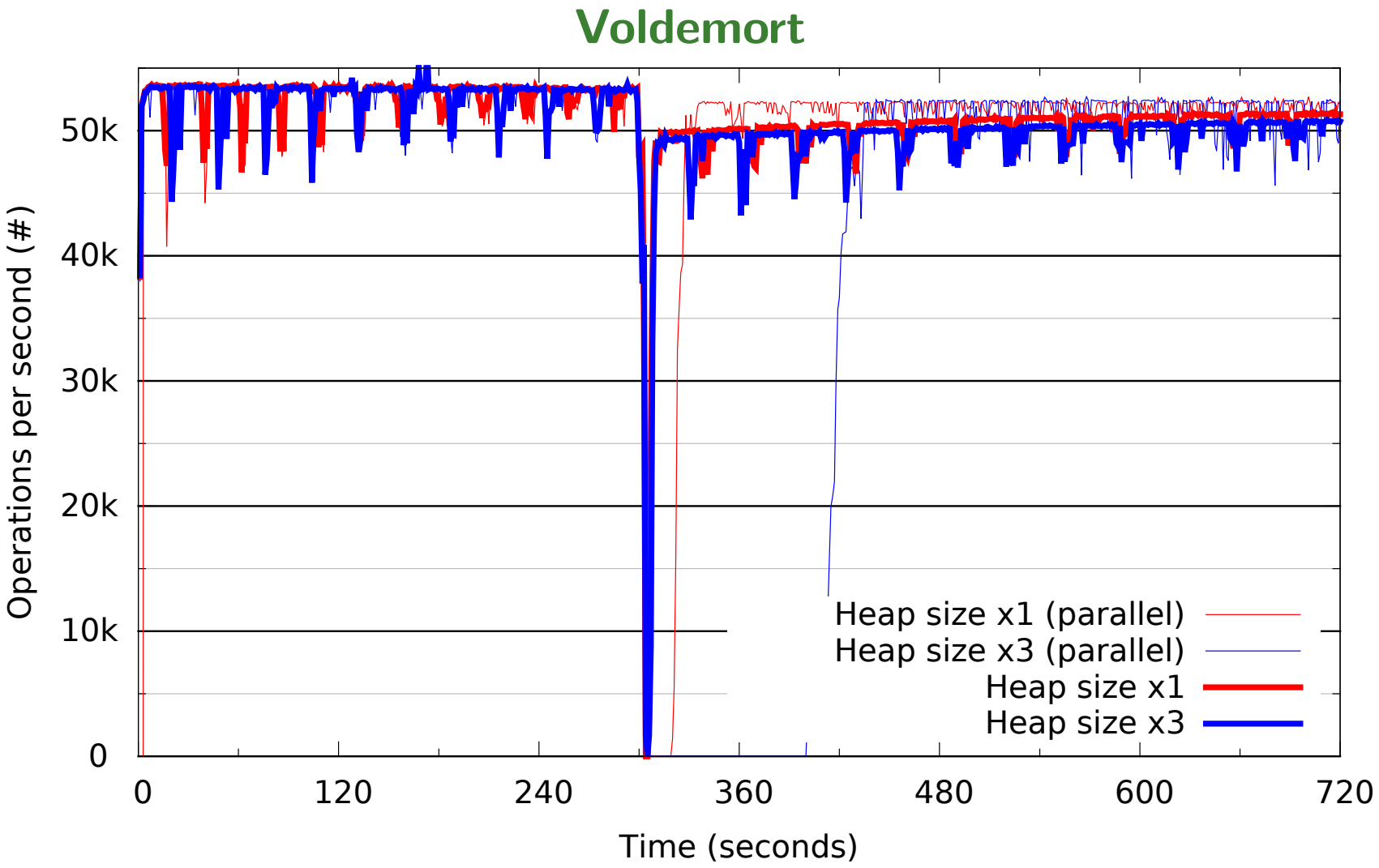
Migrating the state

Lazy migration



Migrating the state

Lazy migration



Migrating the state

Update Pause

Benchmark	Heap Size	Pause (sec)	
		Parallel	*Lazy*
Voldemort	x1	10.7	2.2
	x2	19.1	2.2
	x3	107.4	2.4
H2	x1	9.0	3.1
	x2	15.3	3.7
	x4	30.9	3.7

Rubah

- Efficient / Non-disruptive
- Stock JVM
- Flexible

https://github.com/plum-umd/rubah

Thank You!

Rubah

- Efficient / Non-disruptive
  - No steady-state overhead
  - Parallel/lazy algorithms
- Stock JVM
  - Optimizations rely on internals
    - Unsafe memory access, object memory layout
    - Tested on Oracle HotSpot
    - Adapatable to different JVMs (OpenJDK, IBM Jikes)
- Flexible
  - Unrestricted class updates
  - 13 versions of 5 real-world applications

https://github.com/plum-umd/rubah