

# Rubah


Efficient, General-purpose Dynamic Software Updating for Java


Luís Pina 

luis.pina@ist.utl.pt

Michael Hicks 

mwh@cs.umd.edu

 INESC-ID/Instituto Superior Técnico  
Lisbon, Portugal

 University of Maryland  
College Park, MD, USA

June 28, 2013

5th Workshop on Hot Topics in Software Upgrades  
**HotSWUp'13**

# Motivation

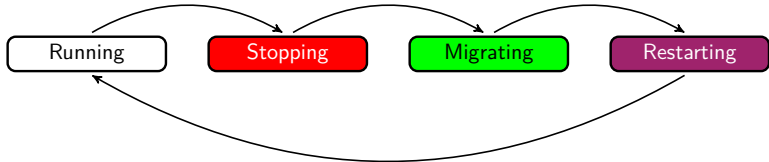
DSU systems for Java

		JVM	
		Custom	Stock
Overhead	High	JDrums	DUSC, DuSTM
	Low	JVolve, DCE-VM	???

# Goals

- Design Rubah outside the JVM
- Low overhead

## Rubah overview



## Rubah overview

Running

Avoid proxies

Stopping

Update points  
(Kitsune)

Migrating

Update class

Restarting

Control flow  
migration  
(Kitsune)

# Updated Program

## H2 Database Engine

- Open source SQL DBMS written in pure Java
- Use releases as versions
  - 1.2.121
  - 1.2.122
  - 1.2.123
- TPC-C benchmark
  - In-memory tables

## Stopping safely

```
01 class TcpServerThread extends Thread {
02
03     void run() {
04
05         // Negotiate protocol params with client
06
07
08
09         while (!stop) {
10
11             // Process client's commands
12
13         }
14     }
15 }
```

## Stopping safely

```
01 class TcpServerThread extends Thread {
02     Transfer transfer;
03     void run() {
04
05         transfer = new Transfer(clientSocket);
06         // Negotiate protocol params with client
07         transfer.init();
08
09         while (!stop) {
10
11             // Process client's commands
12             process();
13         }
14     }
15 }
```



## Stopping safely

```
01 class TcpServerThread extends Thread {
02     Transfer transfer;
03     void run() {
04
05         transfer = new Transfer(clientSocket);
06         // Negotiate protocol params with client
07         transfer.init();
08
09         while (!stop) {
10             Rubah.update("process");
11             // Process client's commands
12             process();
13         }
14     }
15 }
```

## How much manual effort?

[illegible]

## How much manual effort?

## Update points

[illegible]

## Migrating the control flow

```
01 class TcpServerThread extends Thread {
02     Transfer transfer;
03     void run() {
04
05         transfer = new Transfer(clientSocket);
06         // Negotiate protocol params with client
07         transfer.init();
08
09         while (!stop) {
10             Rubah.update("process");
11             // Process client's commands
12             process();
13         }
14     }
15 }
```

## Migrating the control flow

```
01 class TcpServerThread extends Thread {
02     Transfer transfer;
03     void run() {
04         if (!Rubah.isUpdating()) {
05             transfer = new Transfer(clientSocket);
06             // Negotiate protocol params with client
07             transfer.init();
08         }
09         while (!stop) {
10             Rubah.update("process");
11             // Process client's commands
12             process();
13         }
14     }
15 }
```

## How much manual effort?

## Update points

[illegible]

## How much manual effort?

## Update points

+

## Control flow migration

[illegible]

## How much manual effort?

## Update points

+

## Control flow migration

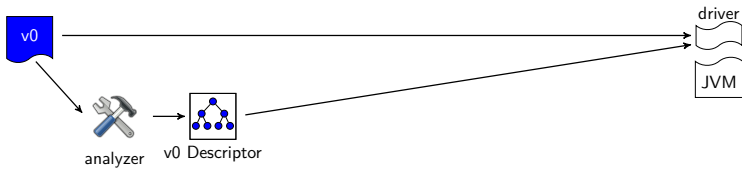
+

## NIO, misc

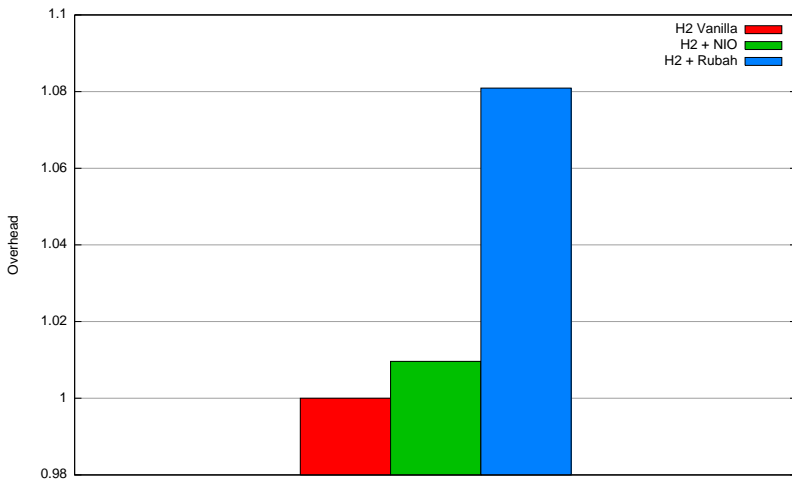
[illegible]



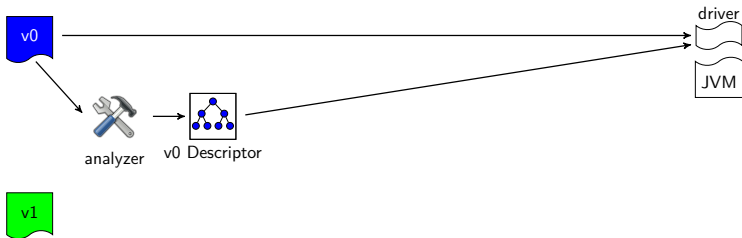
# Tools



## Steady state overhead



# Tools



# Tools

## Update example

*Version 1.2.121*

```
01 class PageStore {
02     private int writeCount;
03
04
05
06     public Data readPage(int pageId) {
07         Data page;
08         // Read page from store
09
10         return page;
11     }
12
13     public void writePage(int pageId, Data data) {
14         // Write page in store
15         writeCount++;
16     }
17 }
```

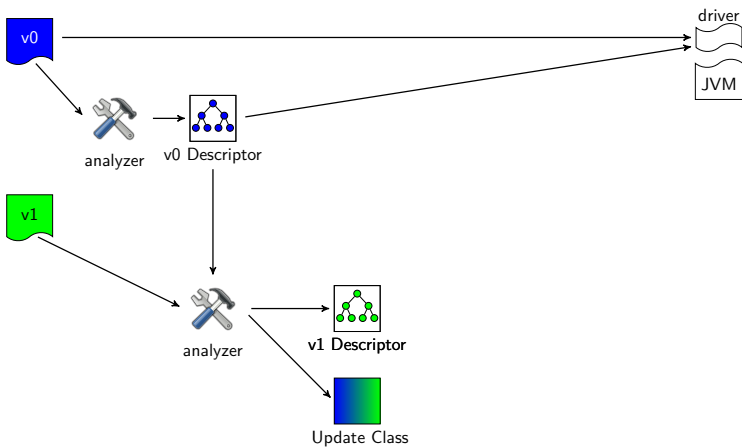
# Tools

## Update example

*Version 1.2.122*

```
01 class PageStore {
02     private int writeCount;
03     private int writeCountBase;
04     private int readCount;
05
06     public Data readPage(int pageId) {
07         Data page;
08         // Read page from store
09         readCount++;
10         return page;
11     }
12
13     public void writePage(int pageId, Data data) {
14         // Write page in store
15         writeCount++;
16     }
17 }
```

# Tools



# Tools

## Stub Update class

```
01 class UpdateClass {
02
03     void convert(v0.PageStore o0, v1.PageStore o1) {
04         // New fields
05         o1.writeCountBase = 0;
06         o1.readCount = 0;
07
08
09     }
10     ...
11 }
```

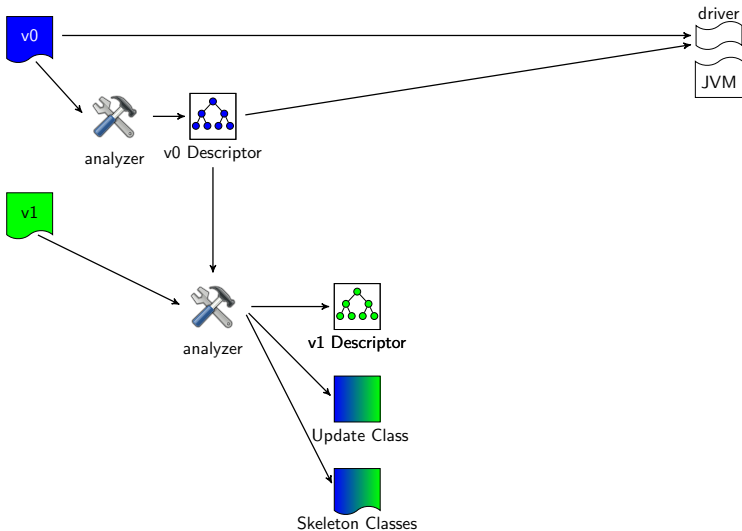
# Tools

## Stub Update class

```
01 class UpdateClass {
02
03     void convert(v0.PageStore o0, v1.PageStore o1) {
04         // New fields
05         o1.writeCountBase = 0;
06         o1.readCount = 0;
07
08
09     }
10     ...
11 }
```



# Tools



# Tools

## Skeleton Classes

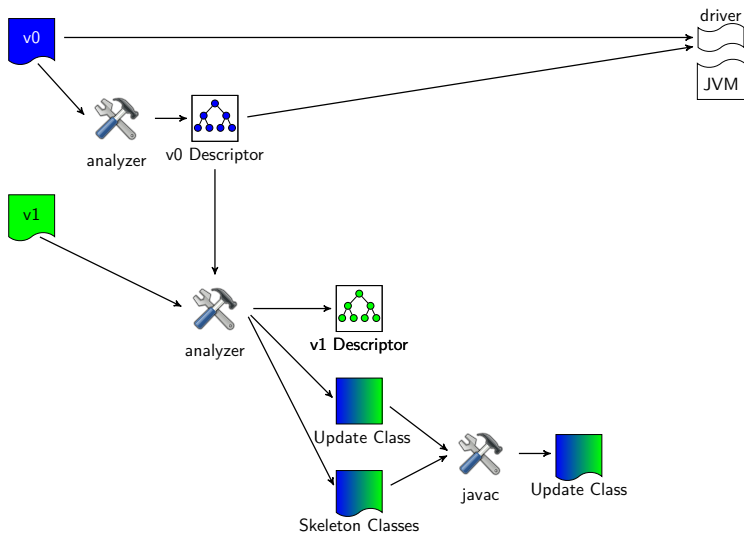
```
01 class v0.PageStore {
02     public int writeCount;
03
04     public Data readPage(int id) { return null; }
05
06     public void writePage(int id, Data data) { return; }
07 }
08
09 class v1.PageStore {
10     public int writeCount;
11     public int writeCountBase;
12     public int readCount;
13
14     public Data readPage(int id) { return null; }
15
16     public void writePage(int id, Data data) { return; }
17 }
```

# Tools

## Skeleton Classes

```
01 class v0.PageStore {
02     public int writeCount;
03
04     public Data readPage(int id) { return null; }
05
06     public void writePage(int id, Data data) { return; }
07 }
08
09 class v1.PageStore {
10     public int writeCount;
11     public int writeCountBase;
12     public int readCount;
13
14     public Data readPage(int id) { return null; }
15
16     public void writePage(int id, Data data) { return; }
17 }
```

# Tools



# Tools

## Update class

```
01 class UpdateClass {
02
03     void convert(v0.PageStore o0, v1.PageStore o1) {
04         // New fields
05         o1.writeCountBase = 0;
06         o1.readCount = 0;
07
08
09     }
10     ...
11 }
```

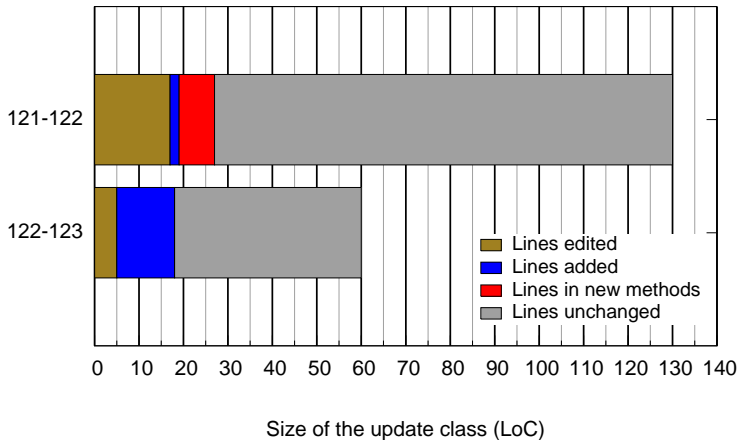
# Tools

## Update class

```
01 class UpdateClass {
02
03     void convert(v0.PageStore o0, v1.PageStore o1) {
04         // New fields
05         o1.writeCountBase = o0.writeCount;
06         o1.readCount = 0;
07
08         o1.writeCount = 0;
09     }
10     ...
11 }
```

# Tools

## Update class manual effort



## Update class manual effort

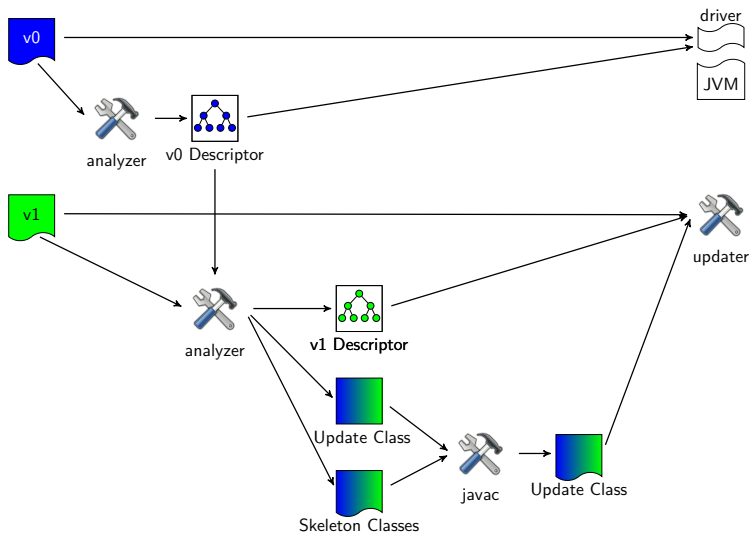
1.2.121 - 1.2.122

1.2.122 - 1.2.123

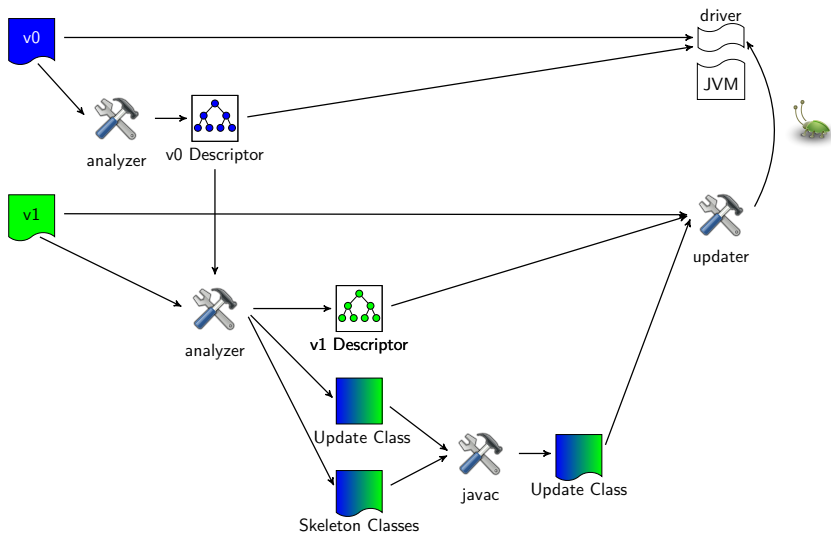
[illegible][illegible]



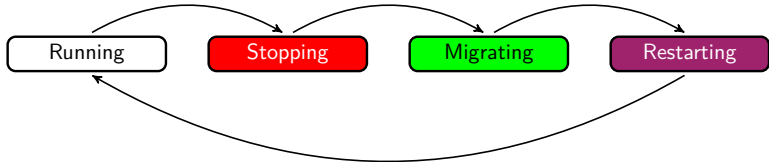
# Tools



# Tools



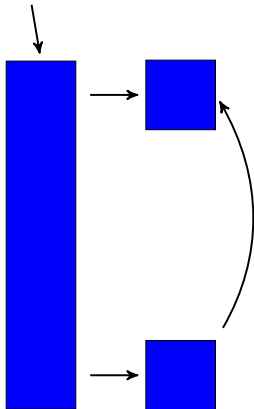
## Update process



# Migrating the program state

Custom heap traversal

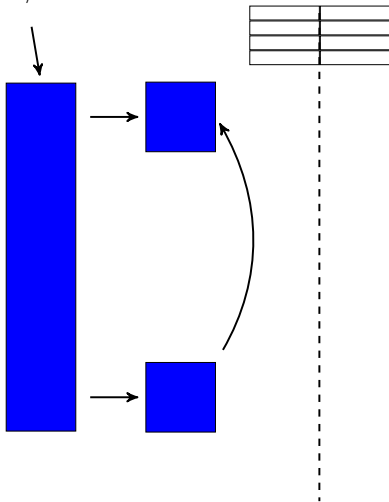
Live thread/Static field



# Migrating the program state

Custom heap traversal

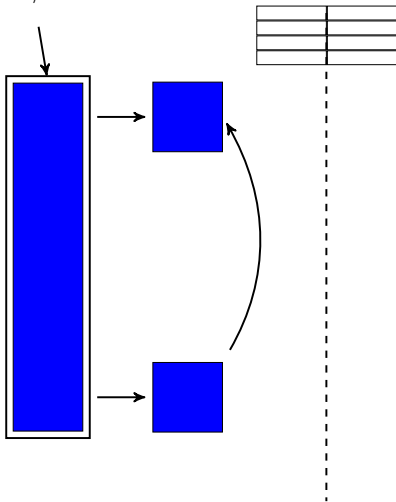
Live thread/Static field



# Migrating the program state

Custom heap traversal

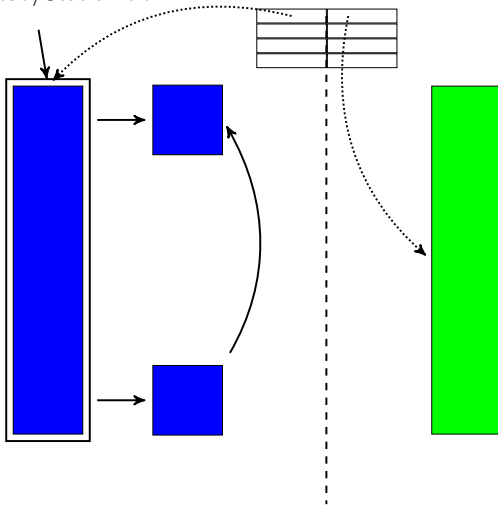
Live thread/Static field



# Migrating the program state

Custom heap traversal

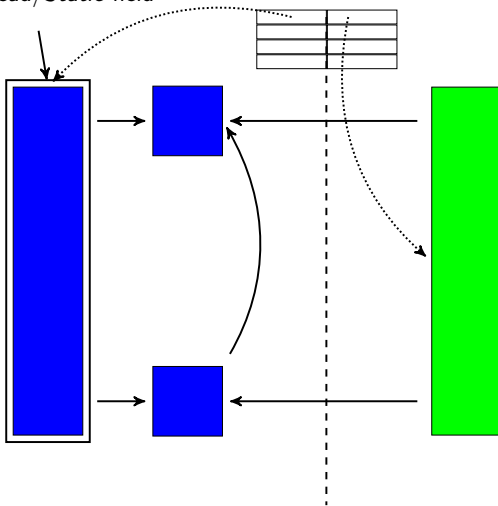
Live thread/Static field



# Migrating the program state

Custom heap traversal

Live thread/Static field

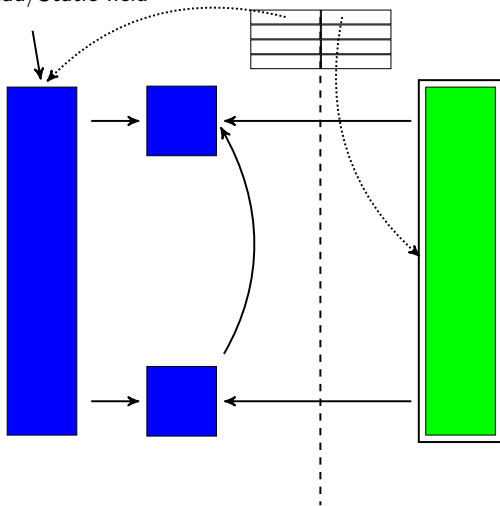




# Migrating the program state

Custom heap traversal

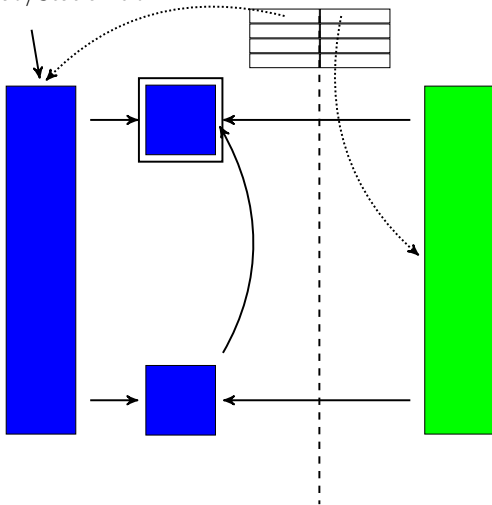
Live thread/Static field



# Migrating the program state

Custom heap traversal

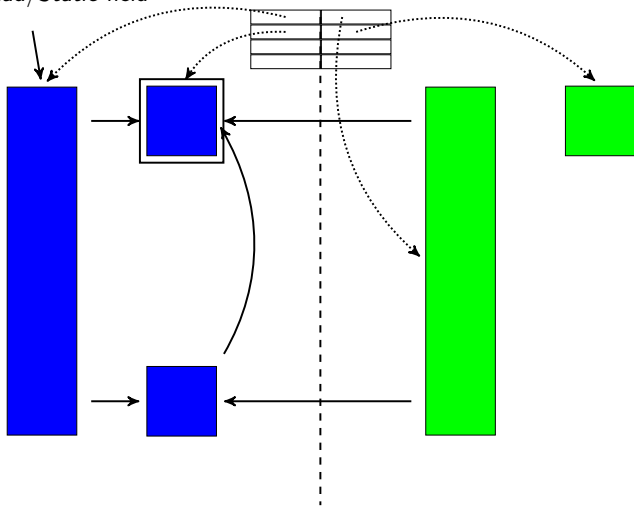
Live thread/Static field



# Migrating the program state

Custom heap traversal

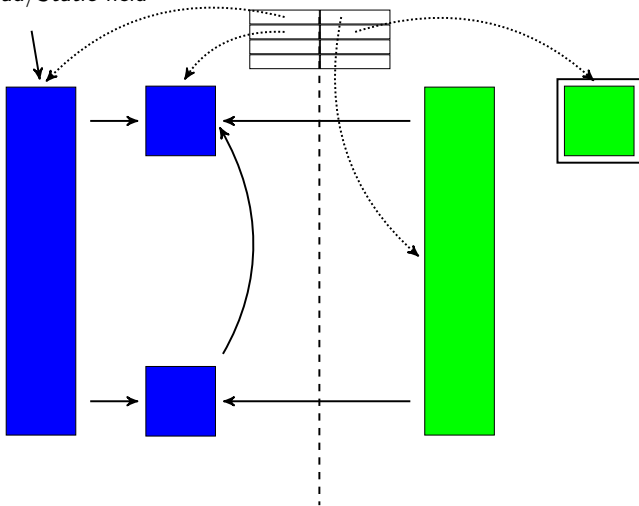
Live thread/Static field



# Migrating the program state

Custom heap traversal

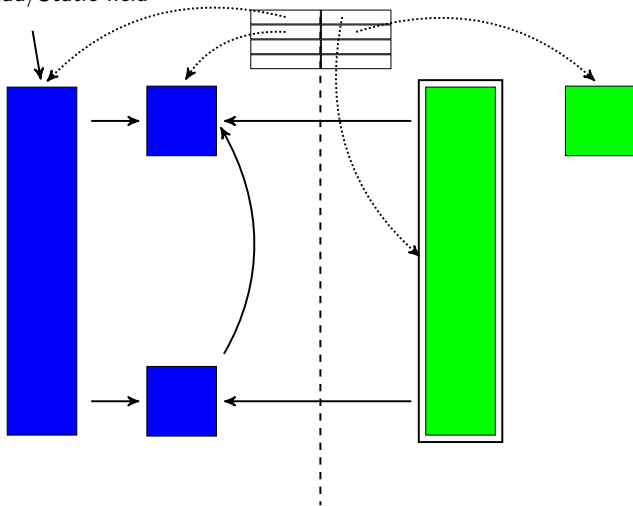
Live thread/Static field



# Migrating the program state

Custom heap traversal

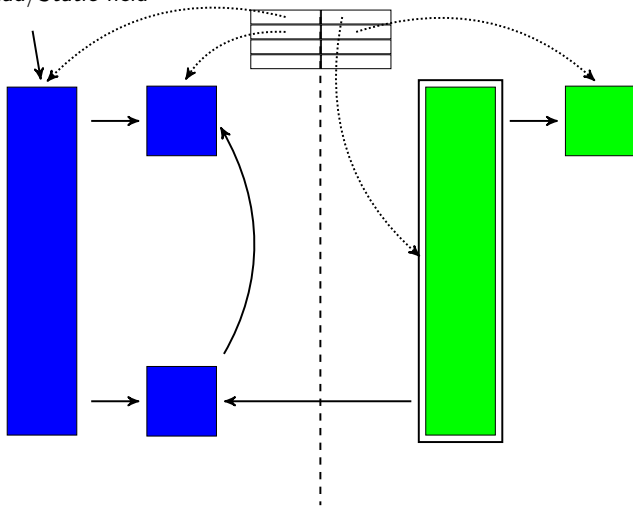
Live thread/Static field



## Migrating the program state

## Custom heap traversal

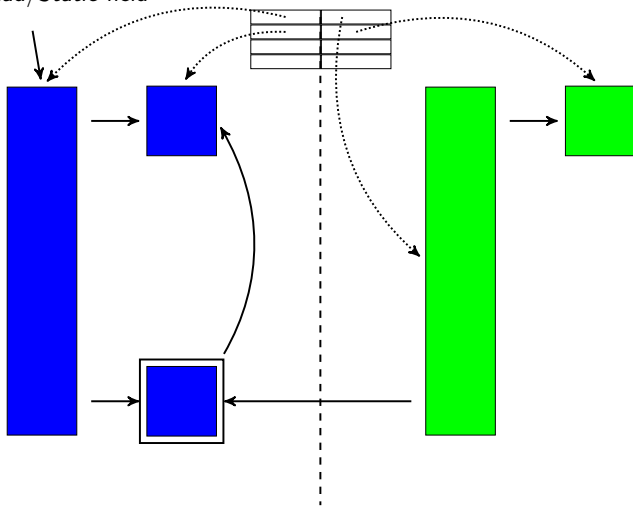
## Live thread/Static field



# Migrating the program state

Custom heap traversal

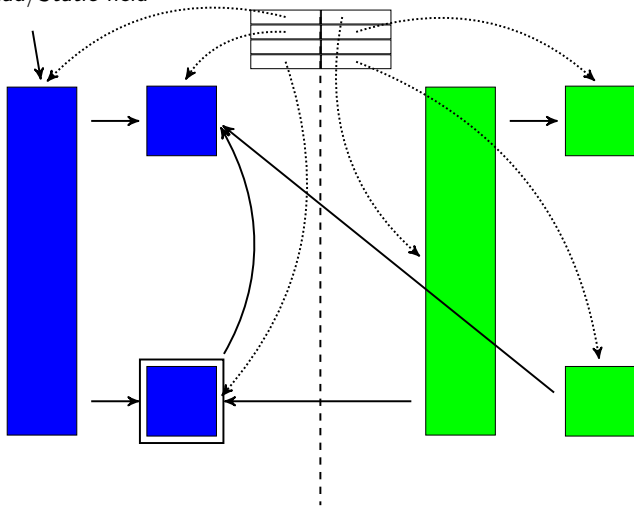
Live thread/Static field



# Migrating the program state

Custom heap traversal

Live thread/Static field

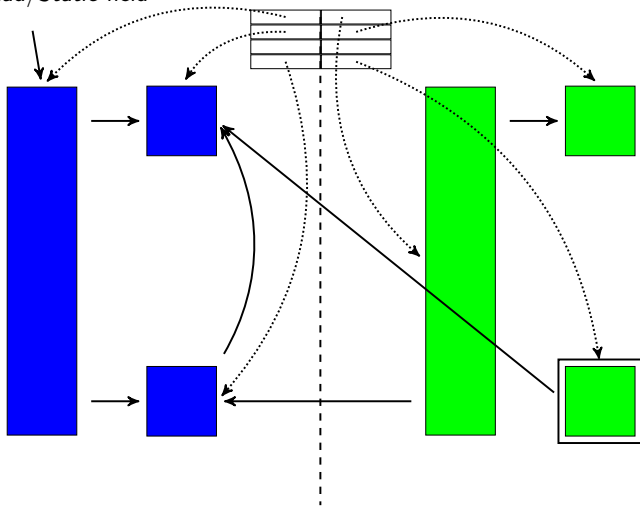




# Migrating the program state

Custom heap traversal

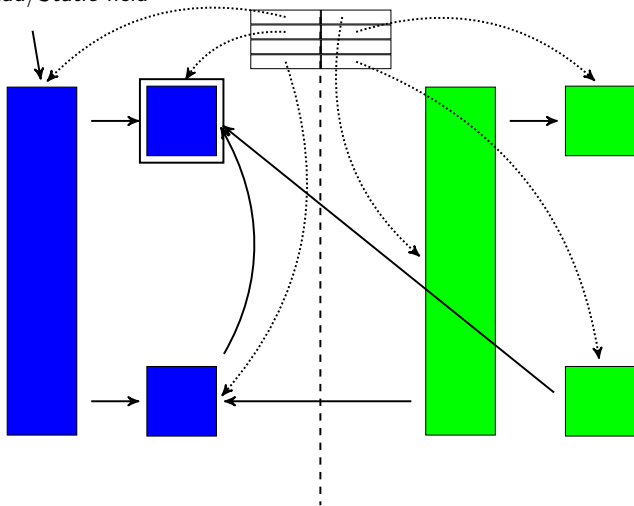
Live thread/Static field



# Migrating the program state

Custom heap traversal

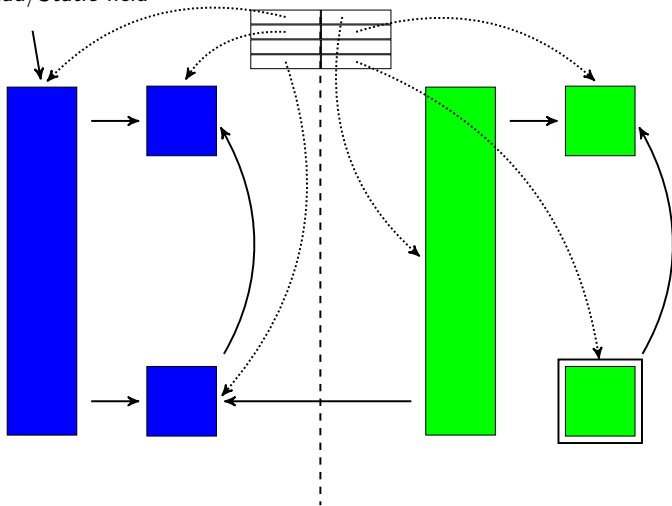
Live thread/Static field



# Migrating the program state

Custom heap traversal

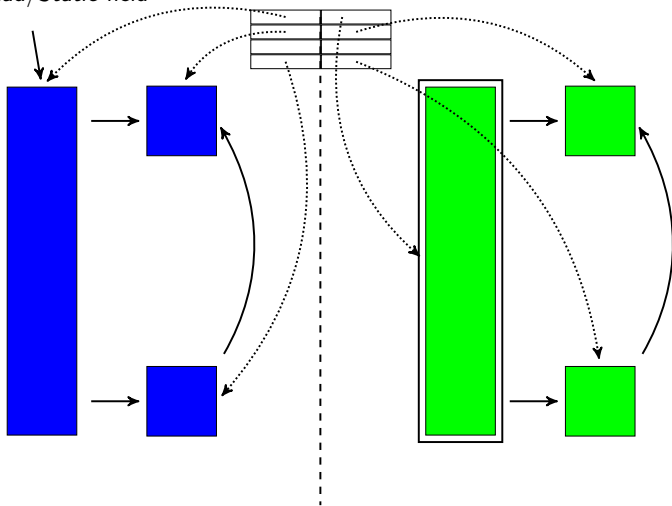
Live thread/Static field



# Migrating the program state

Custom heap traversal

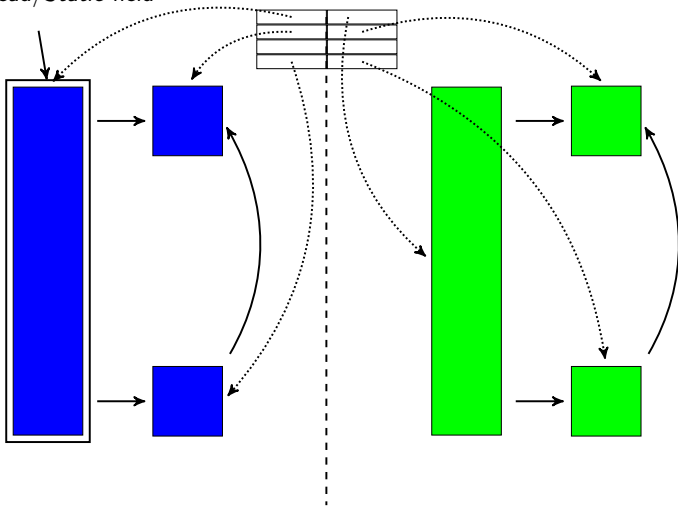
Live thread/Static field



# Migrating the program state

Custom heap traversal

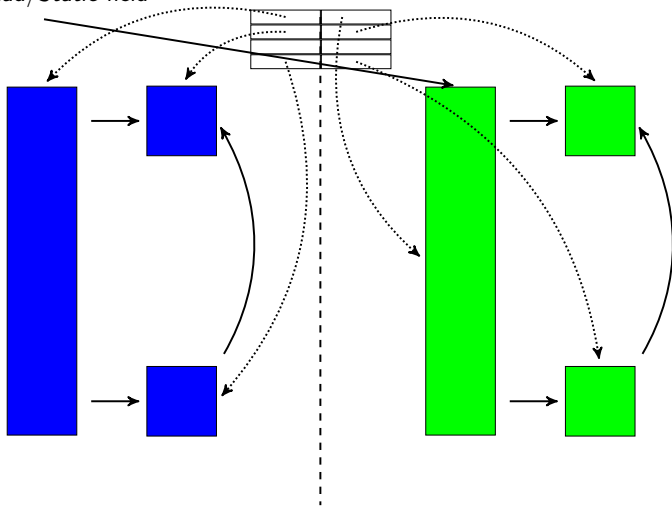
Live thread/Static field



# Migrating the program state

Custom heap traversal

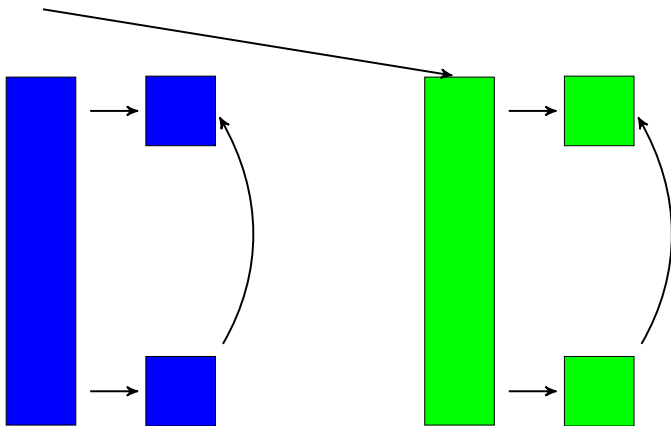
Live thread/Static field



# Migrating the program state

Custom heap traversal

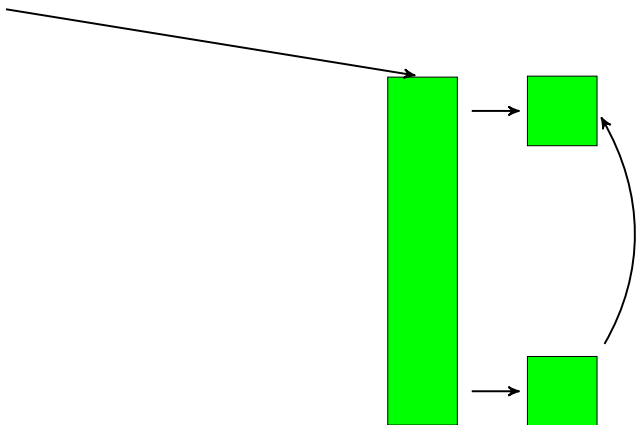
Live thread/Static field



# Migrating the program state

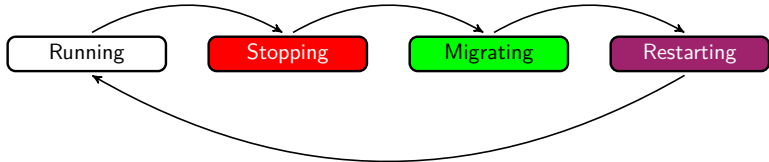
Custom heap traversal

Live thread/Static field

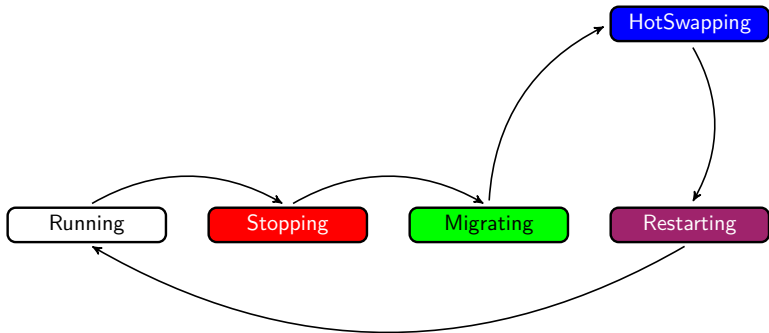




## Update process



## Update process



# HotSwap

```
01 class ClientClass {  
02     void method() {  
03         ...  
04         PageStore__0 store;  
05         ...  
06         store.readPage();  
07         ...  
08     }  
09 }
```

# HotSwap

```
01 class ClientClass {  
02     void method() {  
03         ...  
04         PageStore__0 store;  
05         ...  
06         store.readPage();  
07         ...  
08     }  
09 }
```

# HotSwap

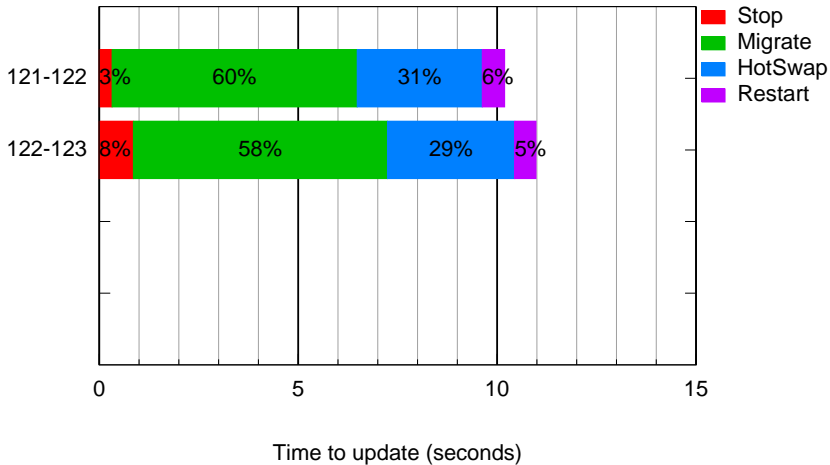
```
01 class ClientClass {  
02     void method() {  
03         ...  
04         PageStore__0 store;  
05         ...  
06         store.readPage();  
07         ...  
08     }  
09 }
```

# HotSwap

```
01 class ClientClass {  
02     void method() {  
03         ...  
04         PageStore__1 store;  
05         ...  
06         store.readPage();  
07         ...  
08     }  
09 }
```

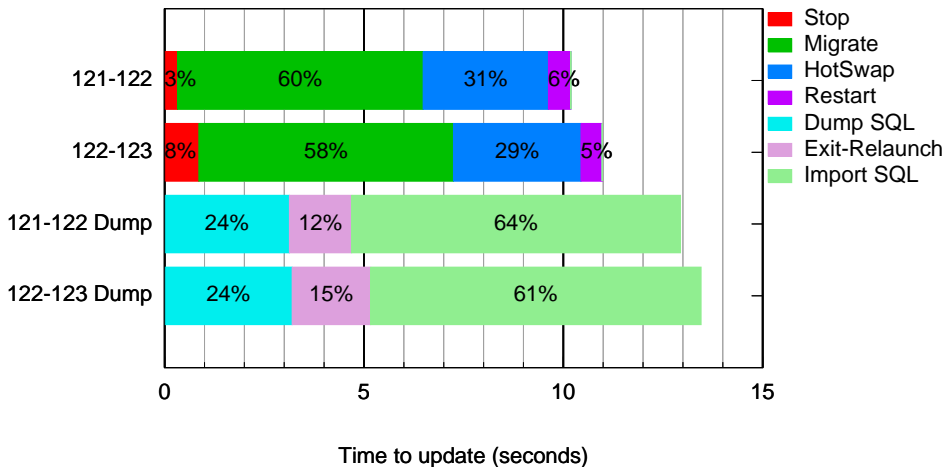
# Update process

## Performance



# Update process

## Performance





## Conclusion

- Rubah designed to use a stock JVM
- Low overhead on steady state
- Long update pause

## Next steps

- Improve update pause times
  - Minimal JVM support (Efficient HotSwap?)
  - Lazy updates
- Further improve steady state performance (8% > 0%)
- More programs, more updates
  - Game server
  - BitTorrent client

# Rubah


Efficient, General-purpose Dynamic Software Updating for Java


Luís Pina 

luis.pina@ist.utl.pt

Michael Hicks 

mwh@cs.umd.edu

 INESC-ID/Instituto Superior Técnico  
Lisbon, Portugal

 University of Maryland  
College Park, MD, USA

June 28, 2013

5th Workshop on Hot Topics in Software Upgrades  
**HotSWUp'13**