

# Kent Academic Repository

## Full text document (pdf)

### Citation for published version

Ugawa, Tomoharu and Jones, Richard E. and Ritson, Carl G. (2014) An On-The-Fly Copying Garbage Collection Framework for Jikes RVM. In: 12th Asian Symposium on Programming Languages and Systems, 17-19 November 2014, Singapore.

### DOI

### Link to record in KAR

<http://kar.kent.ac.uk/45210/>

### Document Version

UNSPECIFIED

#### Copyright & reuse

Content in the Kent Academic Repository is made available for research purposes. Unless otherwise stated all content is protected by copyright and in the absence of an open licence (eg Creative Commons), permissions for further reuse of content should be sought from the publisher, author or other copyright holder.

#### Versions of research

The version in the Kent Academic Repository may differ from the final published version.

Users are advised to check <http://kar.kent.ac.uk> for the status of the paper. **Users should always cite the published version of record.**

#### Enquiries

For any further enquiries regarding the licence status of this document, please contact:

[researchsupport@kent.ac.uk](mailto:researchsupport@kent.ac.uk)

If you believe this document infringes copyright then please contact the KAR admin team with the take-down information provided at <http://kar.kent.ac.uk/contact.html>

# An On-The-Fly Copying Garbage Collection Framework for Jikes RVM

Tomoharu Ugawa

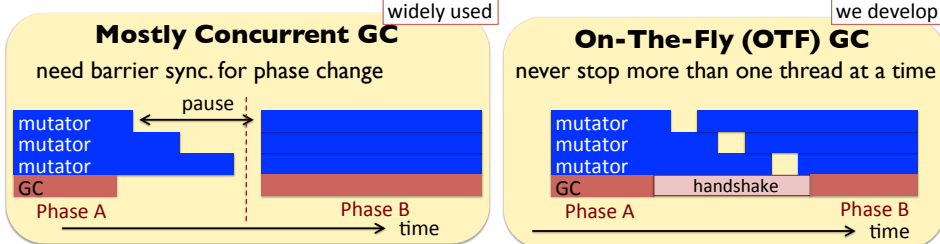


Richard E. Jones  
Carl G. Ritson  
University of Kent  
Computing

## Motivation

GC pauses are undesirable for modern enterprise

➔ Eliminate GC pauses from multi-threaded applications



Challenge on OTF GC: designing correct and efficient write barrier

## Contributions

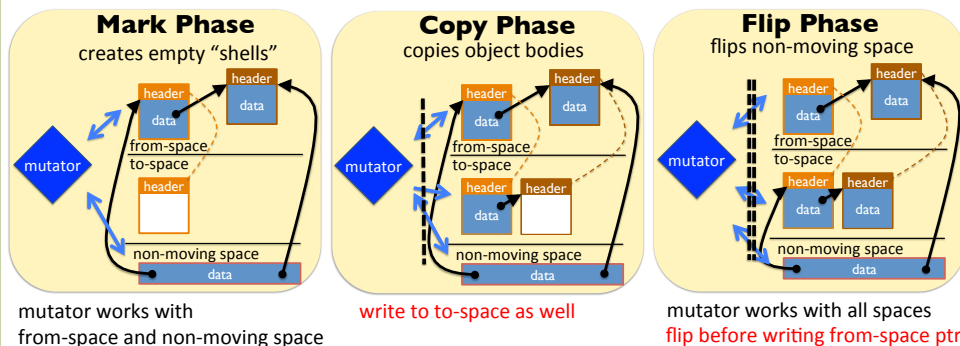
1. Implemented Sapphire OTF GC on widely-used Java VM (Jikes RVM)
2. Developed general framework for OTF, parallel GC
3. Identified a pattern of lagged phase change and fixed a bug in Sapphire
4. Developed efficient concurrent copying method using transactions
5. Support subtleties such as `Object.hashCode()` and weak references

## 1. Sapphire [Hudson & Moss, 2001]

The only known on-the-fly copying GC, but no full-scale implementation exists

Replication: create semantically equivalent replica behind mutators

**write barrier** enforces invariant: no to-space → from-space pointer



## 2. Lagged Phase Change

Different phases require different invariants

**Sapphire's bug**

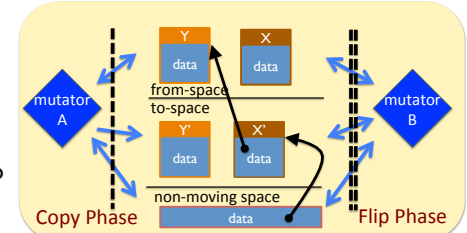
Mutator A in copy phase

INV: no non-moving → to-space

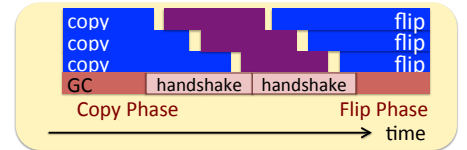
Mutator B in flip phase

INV: no new non-moving → from-space

1. B stores pointer to to-space object X' to non-moving space
2. A loads X' from non-moving space
3. A stores pointer to from-space object Y to a slot of X'



We introduce **intermediate states** to prevent conflicts between invariants of adjacent phases



## 3. Concurrent Copy

Sapphire: compare-and-swap per word

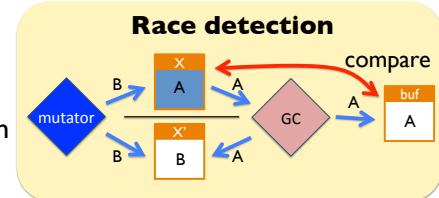
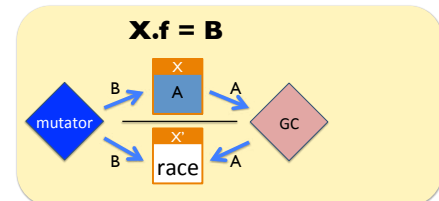
Our solution: **copy-fence-verify** per object

```
copy-object(X, X')
for(f : fields(X))
{ buf.f = X.f; X'.f = forward(X.f); }
fence;
for (f : fields(X))
if (X.f != buf.f) fallback;
```

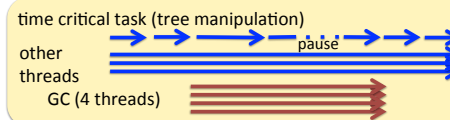
- fewer synchronisation
- sequential memory access

Can use HW transaction for race detection

- transaction setup was heavier than fence
- ➔ similar throughput to SW transaction



## Evaluation Result



- Long pauses were very rare (observed regardless of GC)
- Write barrier slowed down mutators to roughly half speed

