



Kent Academic Repository

Ugawa, Tomoharu, 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.

Downloaded from

<https://kar.kent.ac.uk/45210/> The University of Kent's Academic Repository KAR

The version of record is available from

This document version

UNSPECIFIED

DOI for this version

Licence for this version

UNSPECIFIED

Additional information

Best poster prize, APLAS 2014

Versions of research works

Versions of Record

If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts

If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in *Title of Journal*, Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries

If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our [Take Down policy](https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies) (available from <https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies>).

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

Tomoharu Ugawa

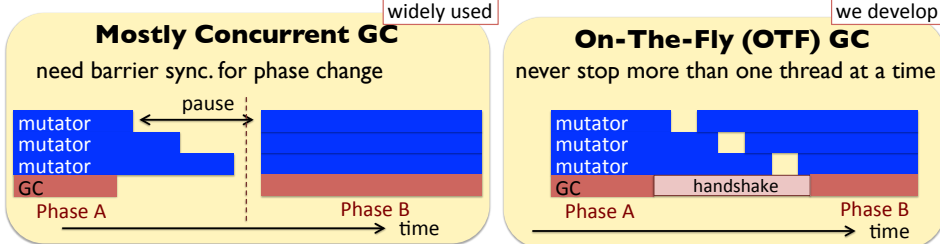


Richard E. Jones
Carl G. Ritson
University of Kent
Computing

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

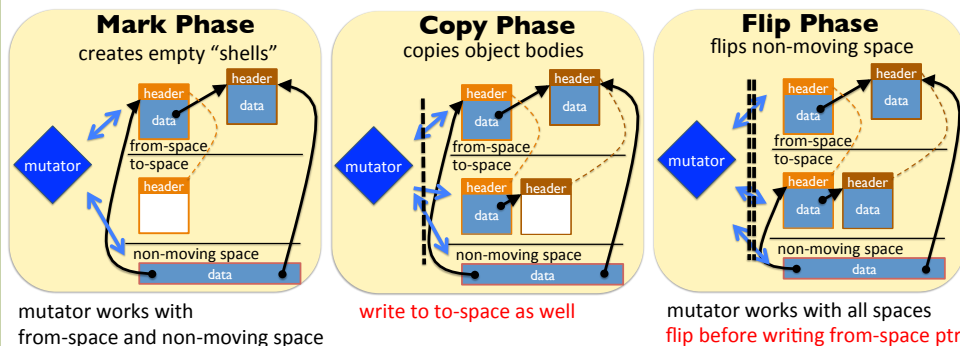
1. Implemented Sapphire OTF GC on widely-used Java VM (Jikes RVM)
2. Developed general framework for OTF, parallel GC
3. Identified a pattern of lagged phase change and fixed a bug in Sapphire
4. Developed efficient concurrent copying method using transactions
5. Support subtleties such as `Object.hashCode()`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

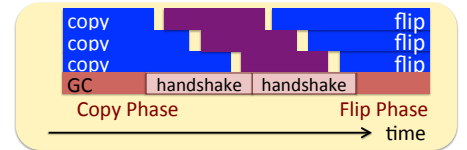
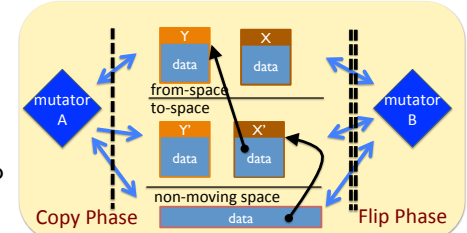
Mutator A in copy phase

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

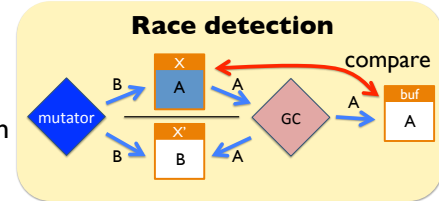
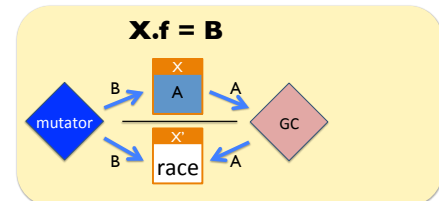
Our solution: **copy-fence-verify** per object

```
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;
```

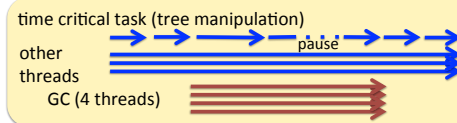
- fewer synchronisation
- sequential memory access

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

