A Fully Parallel LISP2 Compactor with preservation of the Sliding Properties

Xiao-Feng Li, Ligang Wang, Chen Yang

Intel China Research Center

2008-08-01

Agenda

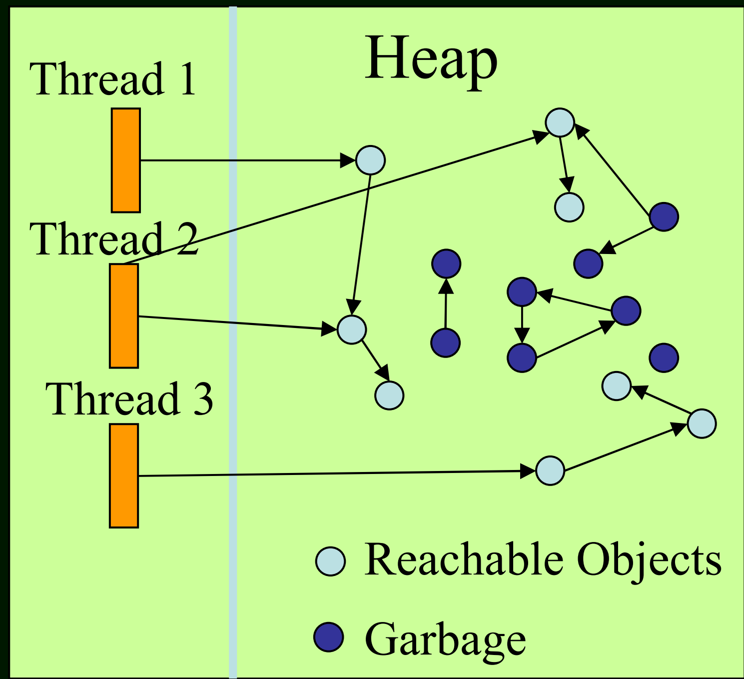
- LISP2 Sliding Compactor
- Parallel LISP2 Compactor
- Working in Apache Harmony
- Evaluations
- Summary and On-going work

Agenda

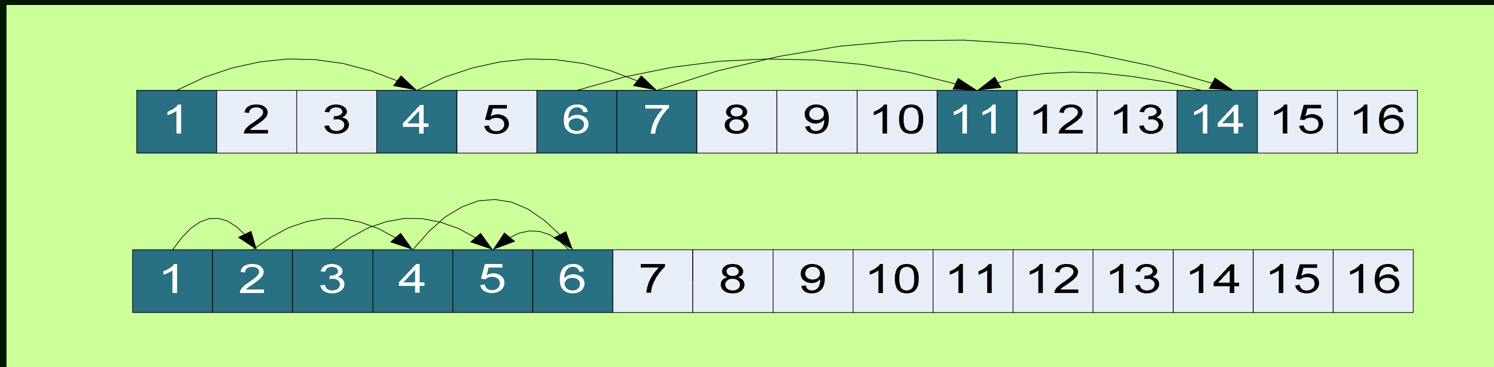
- LISP2 Sliding Compactor
- Parallel LISP2 Compactor
- Working in Apache Harmony
- Evaluations
- Summary and On-going work

One Slide on Garbage Collection

- GC is universally available in modern runtime systems
- Reachability analysis
  - Traverse object connection graph from application's context
  - Commonly used



Sliding Compactor



- Properties
  - In-place collection: little extra space required
  - Heap de-fragmentation: high heap utilization
  - Sliding compaction: Object order preservation
  - Contiguous free space: Bump-pointer alloc

Is Sliding Compactor Good?

- Criteria for stop-the-world GC
  - Allocation performance
  - Mutation performance
  - Collection performance
    - Pause time
  - Memory requirement

Sliding Compactor: Pros

- Allocation performance
  - Bump-pointer allocation → Fast
- Mutation performance
  - Object order preservation & Bump-pointer allocation

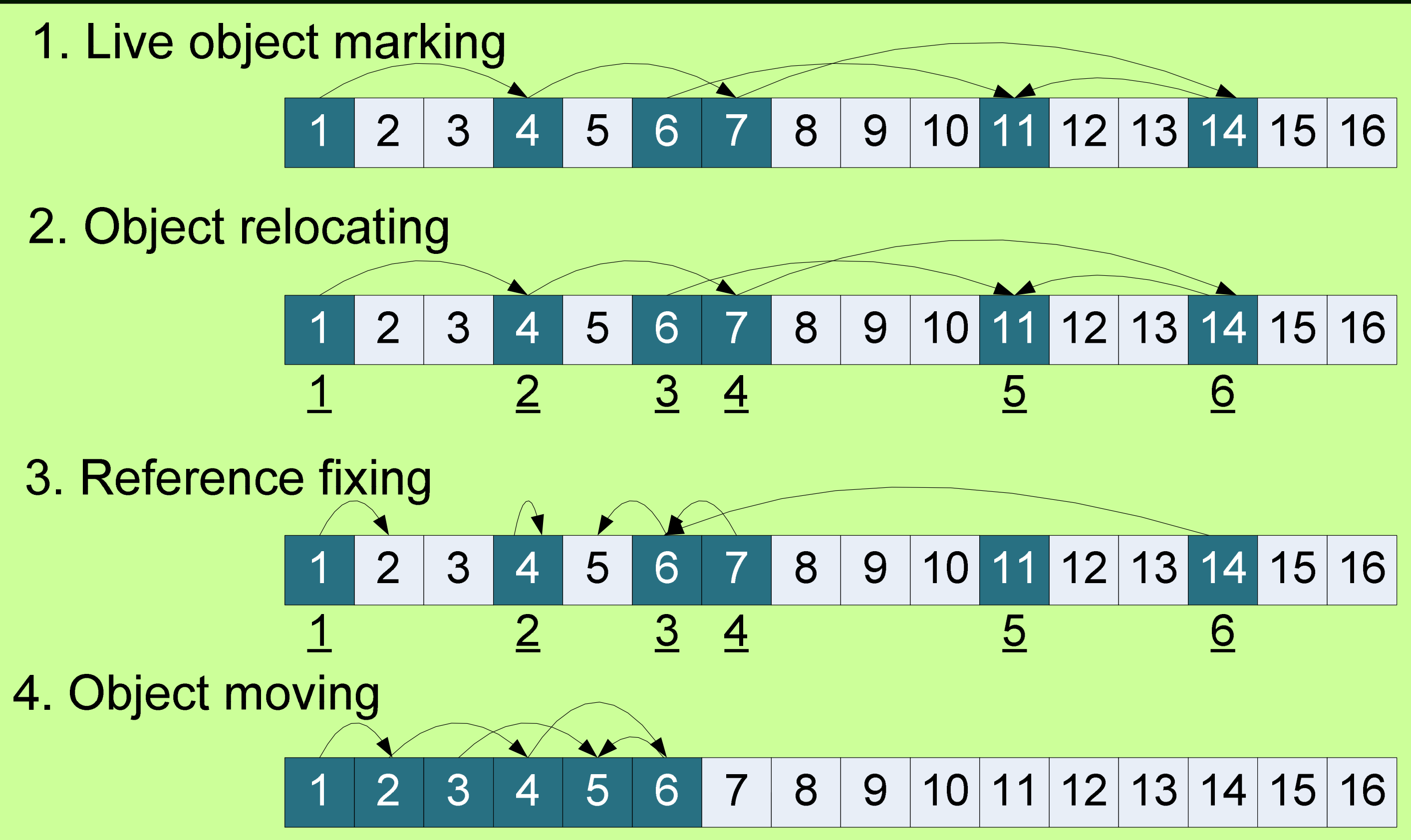
  → good locality & prefetch opportunity
- Memory requirement
  - In-place collection & heap de-defragmentation

  → Small footprint

Sliding Compactor: Cons

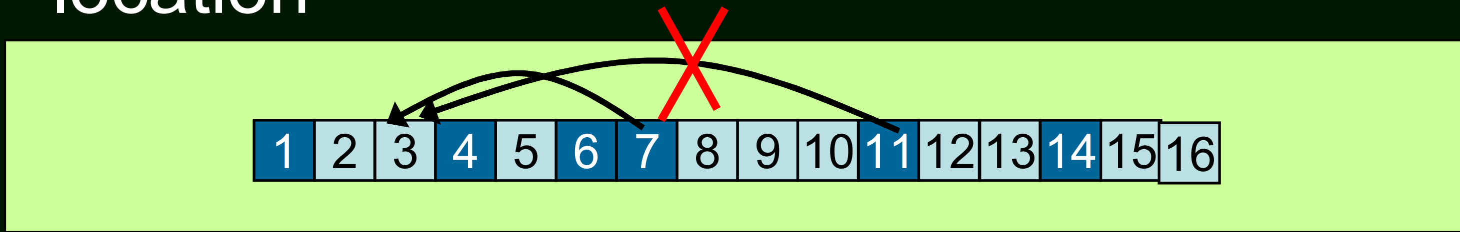
- Collection performance
  - Trade time for the advantages …
  - Even though, sliding compactor is widely used
    - For entire heap collection
- Collection performance can be improved
  - Parallelization is one of the approaches
- This work
  - Parallelization of LISP2 Compactor

LISP2 Sliding Compactor

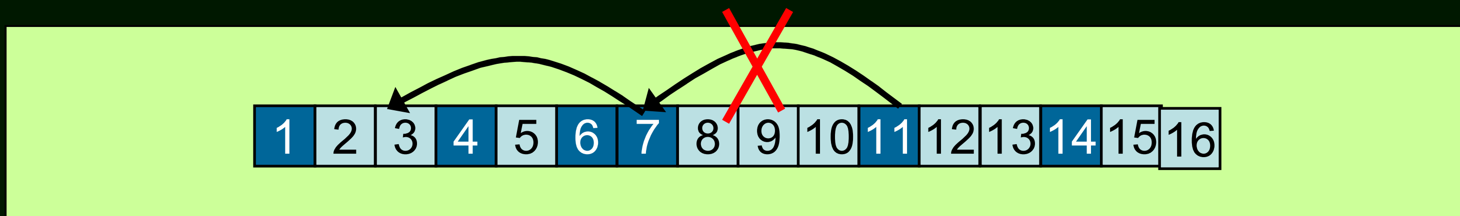


Difficulties in Parallelization

- To keep the sliding properties
  - Two collectors may compete for same target location



  - One collector may overwrite another collector's data

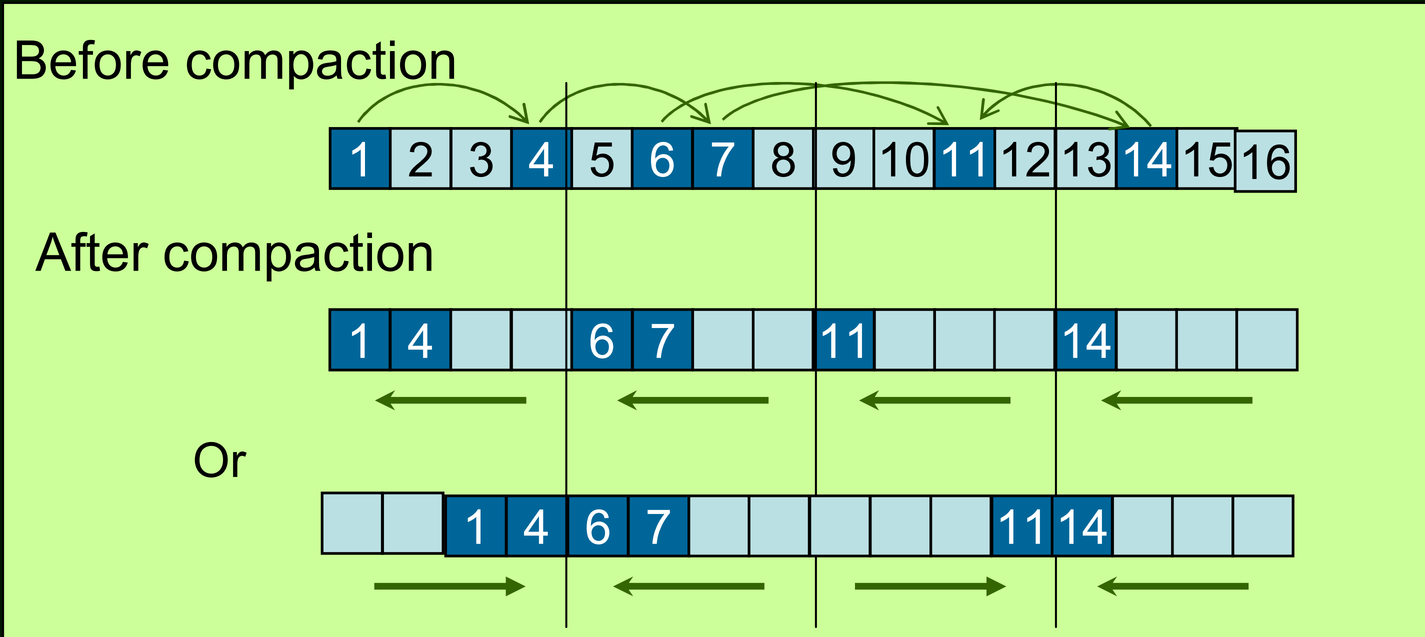


More Difficulties for Scalability

- Irregular program, ordered list of blocks
  - Load balance
    - One collector should not stay idle when there are still tasks remaining
  - Parallelization efficiency
    - One collector should not repeat any work done by another collector
  - Synchronization overhead
    - Should avoid long time in critical section or spinning

Prior Parallel LISP2 Compactor

- [Flood-Detlefs-Shavit-Zhang 2001]
  – Idea: Heap is divided to n regions, and each region is compacted independently



Our Parallel LISP2 Compactor

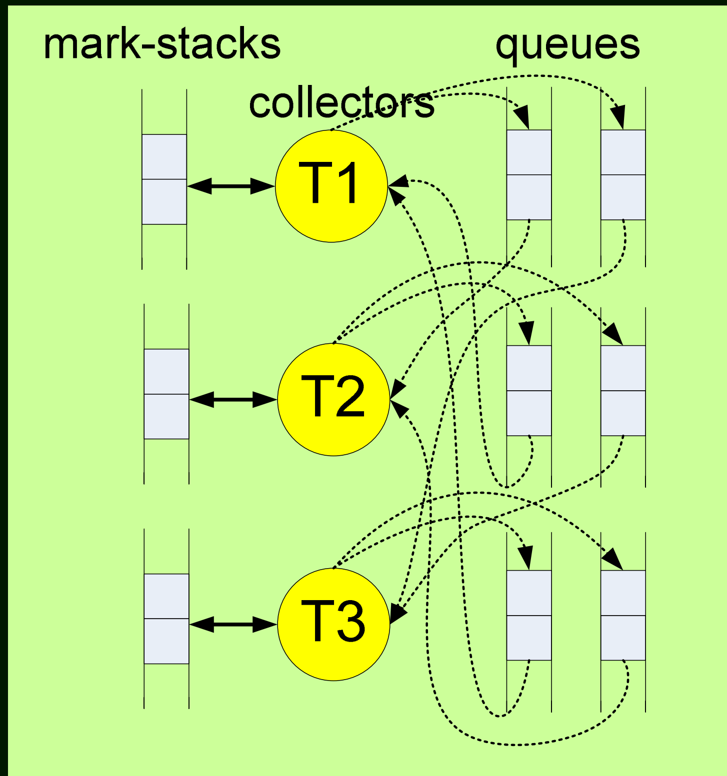
- Parallel granularity: heap block
  - Source block / target block
  - One block has the two roles
    - Ordered list of source block or target block?
- Key idea
  - Relocating phase: ordered list of source blocks
  - Moving phase: ordered list of target blocks
  - Connected through a dependence list

Agenda

- LISP2 Sliding Compactor
- Parallel LISP2 Compactor
- Working in Apache Harmony
- Evaluations
- Summary and On-going work

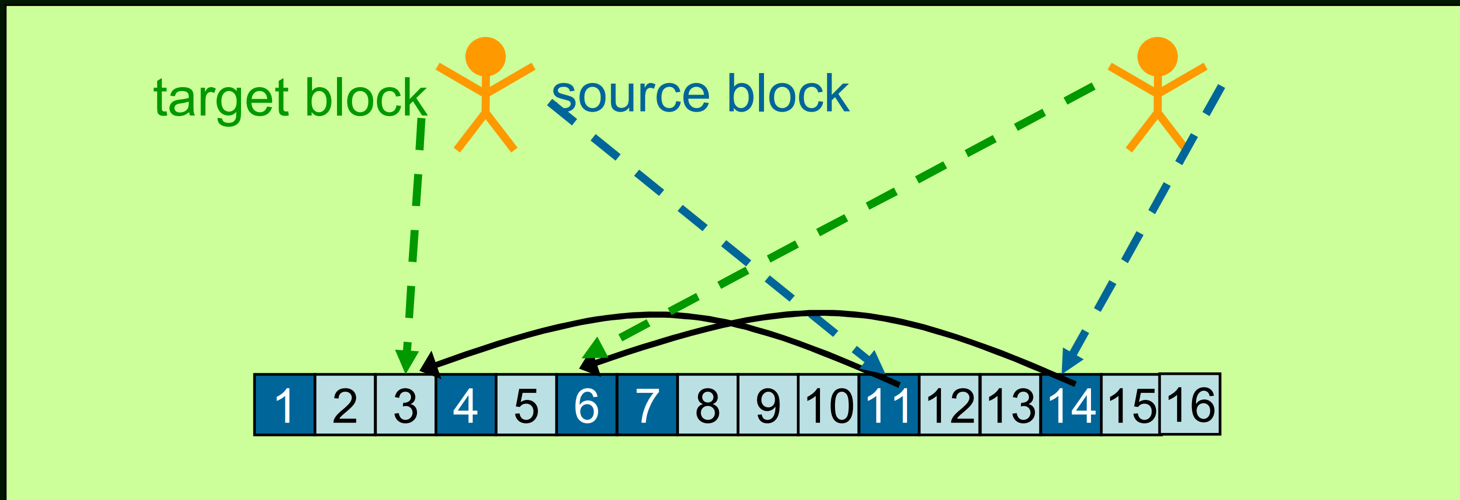
1. Parallel Live Object Marking

- Traverse object connect. graph in parallel
  – Depth-first traversal
  – For load balance, a collector pushes its extra tasks to other collectors



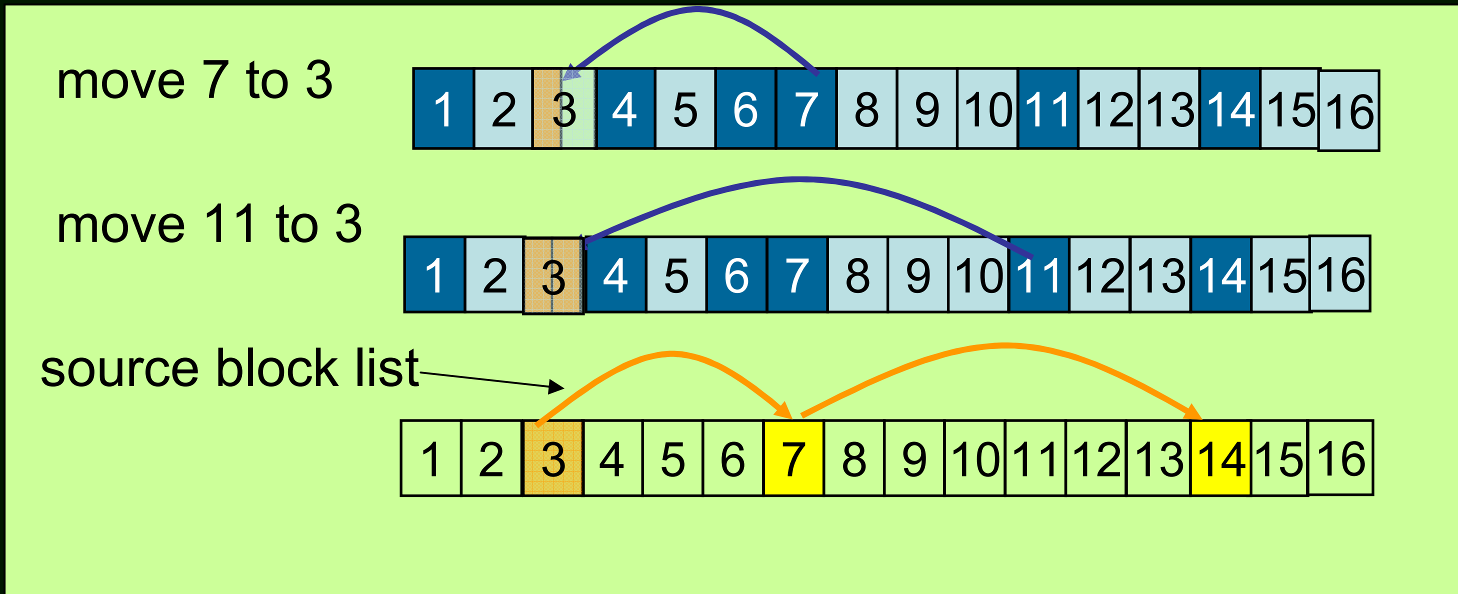
2. Parallel Object Relocating

- In any time, a collector always holds a source block and a target in hands
  - For each live object in source block, computes its target address

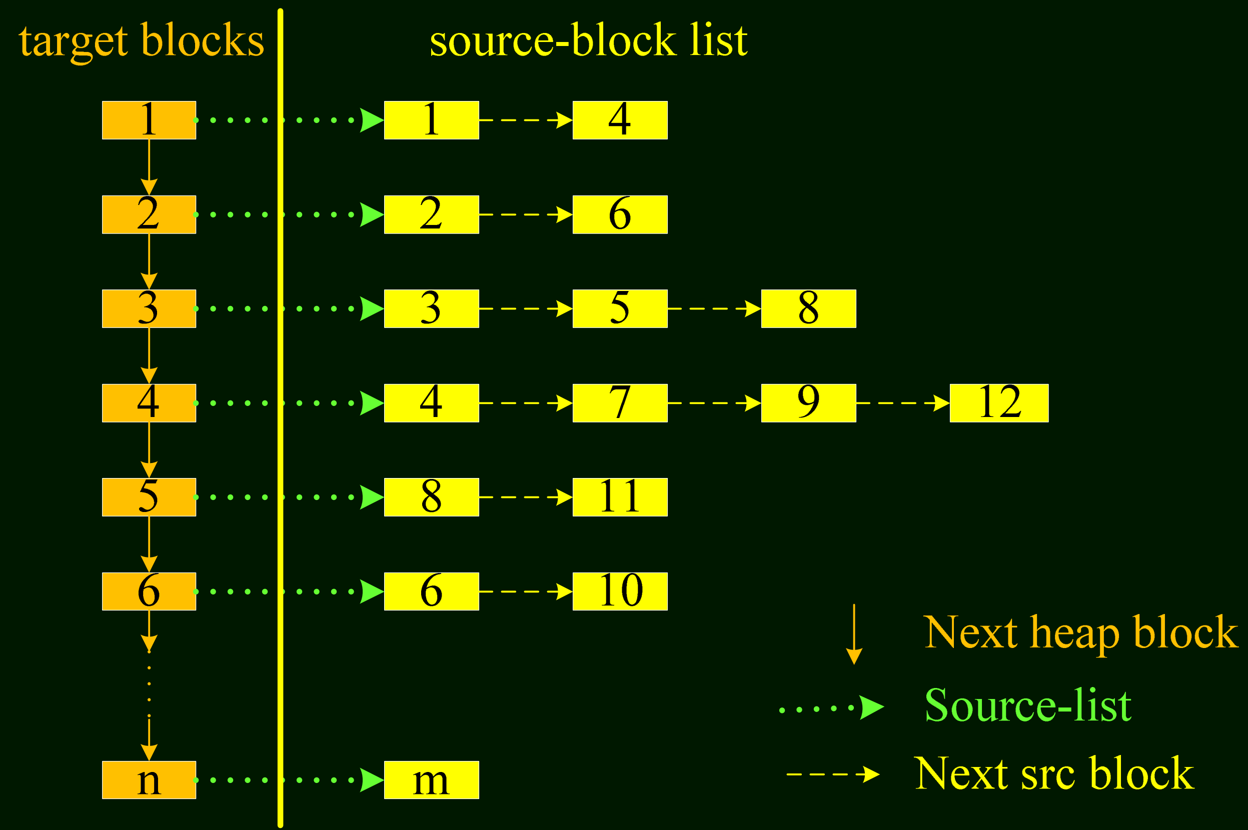


Metadata Maintained

- Collectors maintain a source-block list for each target-block (dependence list)
  - Recording its data sources



Example: Source-Block List

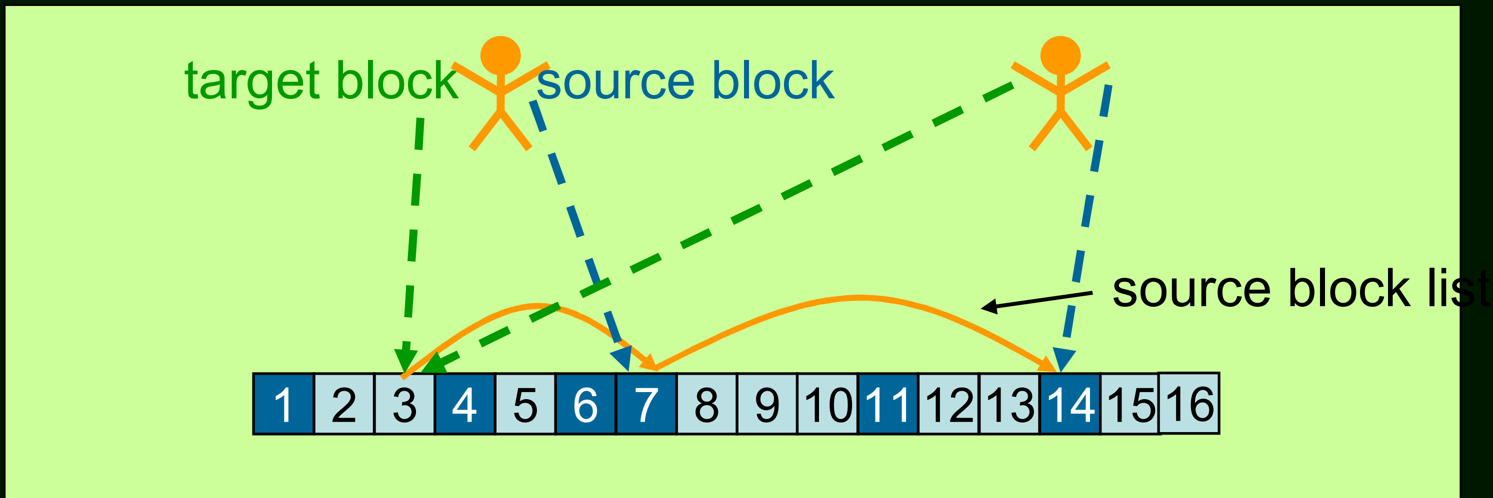


3. Parallel Reference Fixing

- For each object, reference fixing is a local operation
  - The collectors grab blocks atomically from the heap and fix the references locally
  - Inherently highly parallel

4. Parallel Object Moving

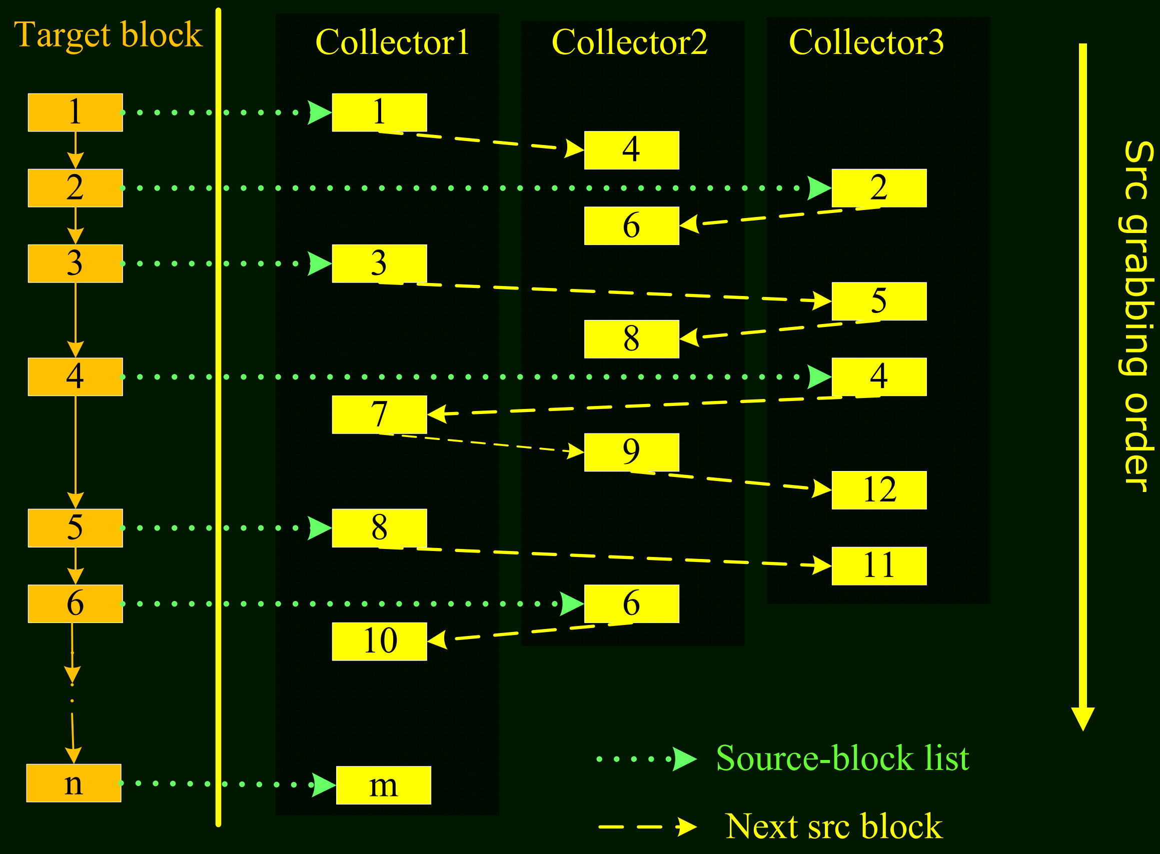
- The collectors grab target blocks in address order
  - Move in the live objects from the blocks in source block list



Metadata Maintained

- To avoid a source block is overwritten before its data are moved away
  - A flag in source block
    - Indicating if its data are moved out
    - Implemented by *target-count*, recording the number of target blocks of a source block
  - Possible values of *target-count* : 0,1,2
    - 0 : no useful data (all dead or moved)
    - Decremented once copied to a target block

Example: Object Moving



Synchronization Control

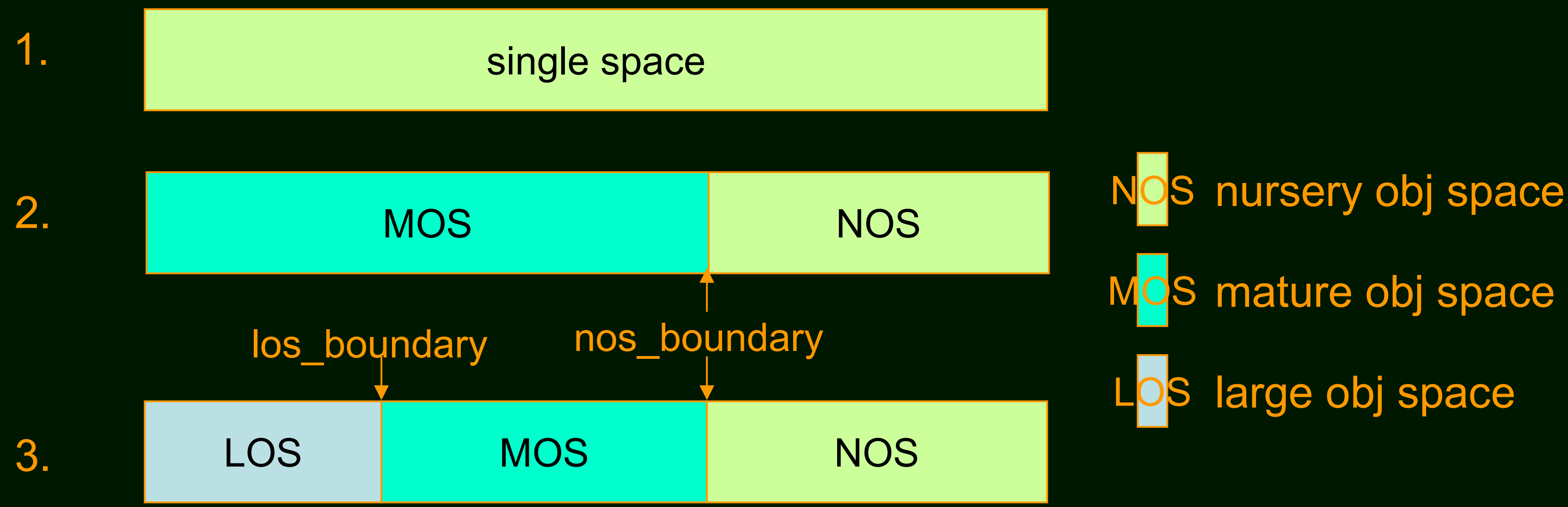
- In object relocating phase
  - Collectors atomically grab source blocks from the heap in address order
- In object moving phase
  - Collectors atomically grab target blocks from the heap in address order

Agenda

- LISP2 Sliding Compactor
- Parallel LISP2 Compactor
- Working in Apache Harmony
- Evaluations
- Summary and On-going work

GC in Real JVM

- GC toolkit in Apache Harmony
  - Generational, parallel, concurrent
- Heap configurations

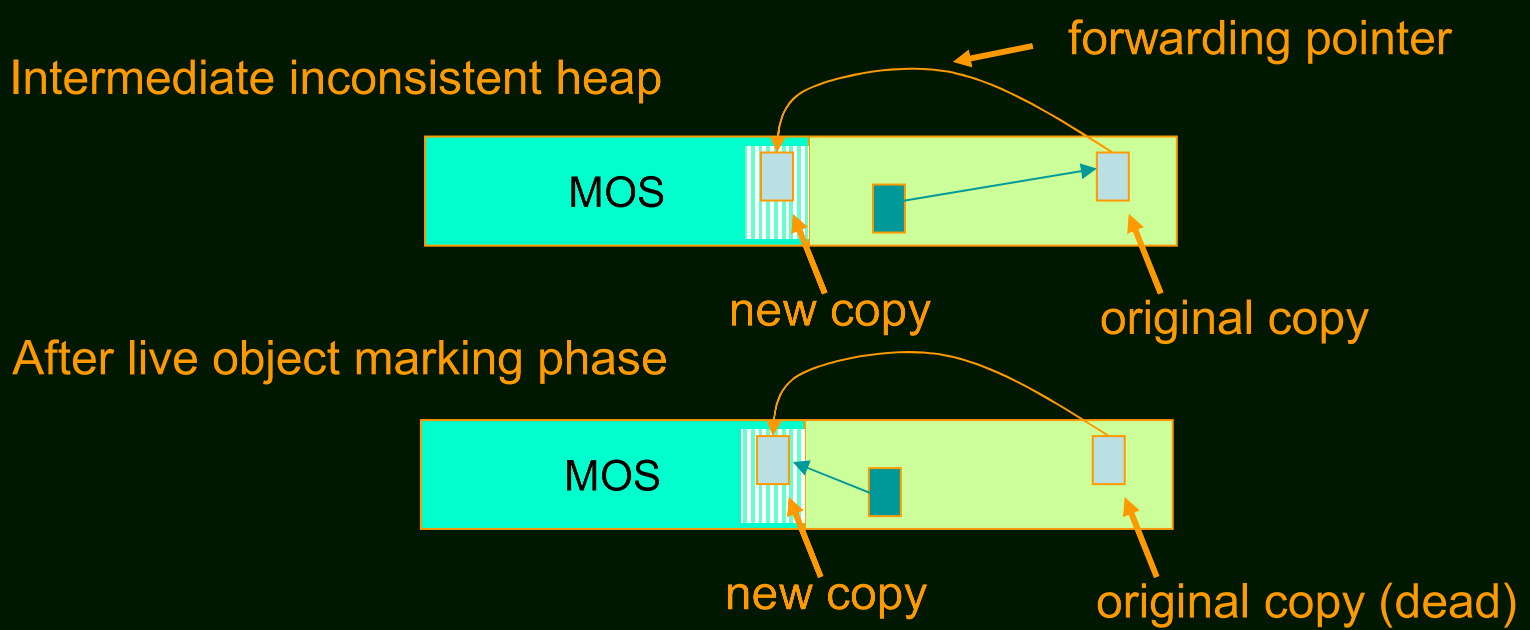


NOS + MOS

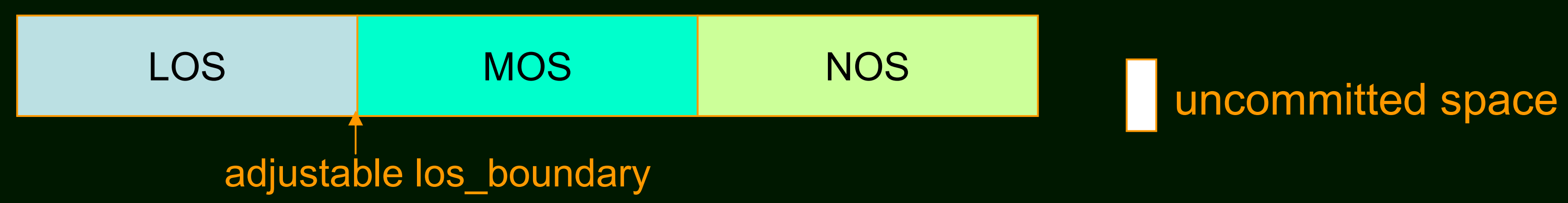
- High-end of MOS should be adjustable
  - Can be satisfied trivially due to sliding compaction nature

Fallback Compaction

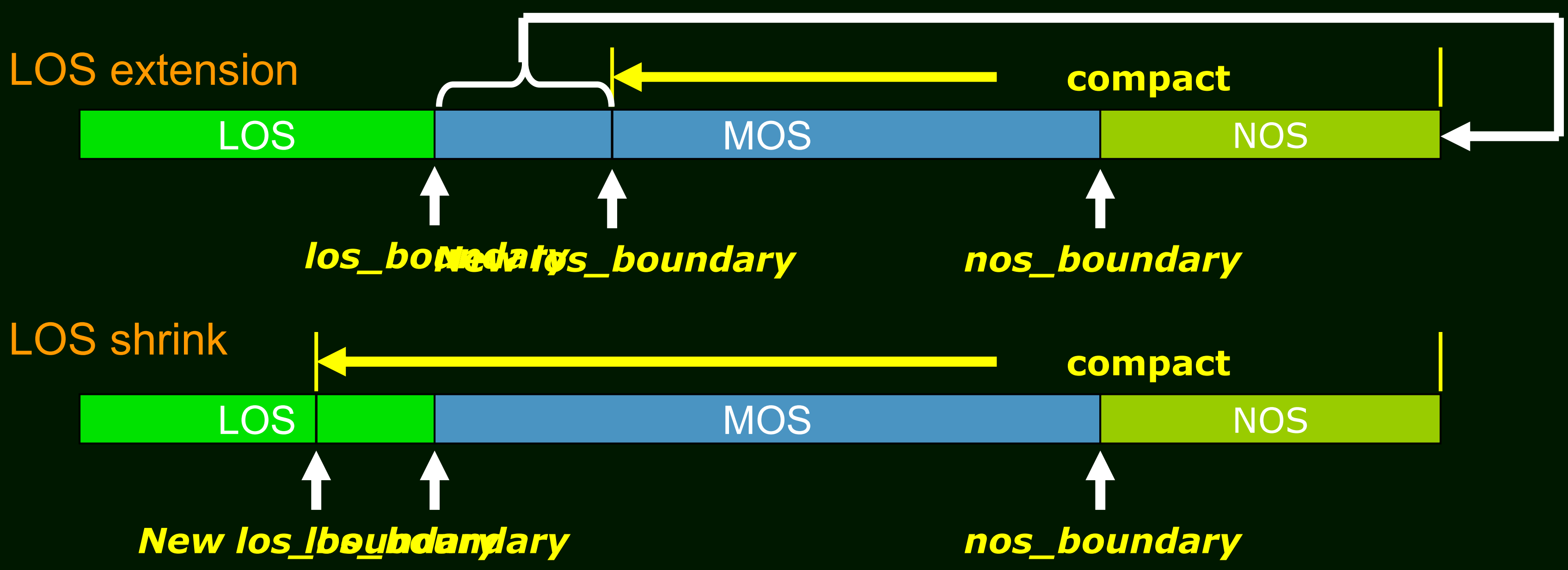
- Copy reserve is inadequate to accommodate NOS survivors
  - Fall back to entire-heap compaction



NOS+MOS+LOS

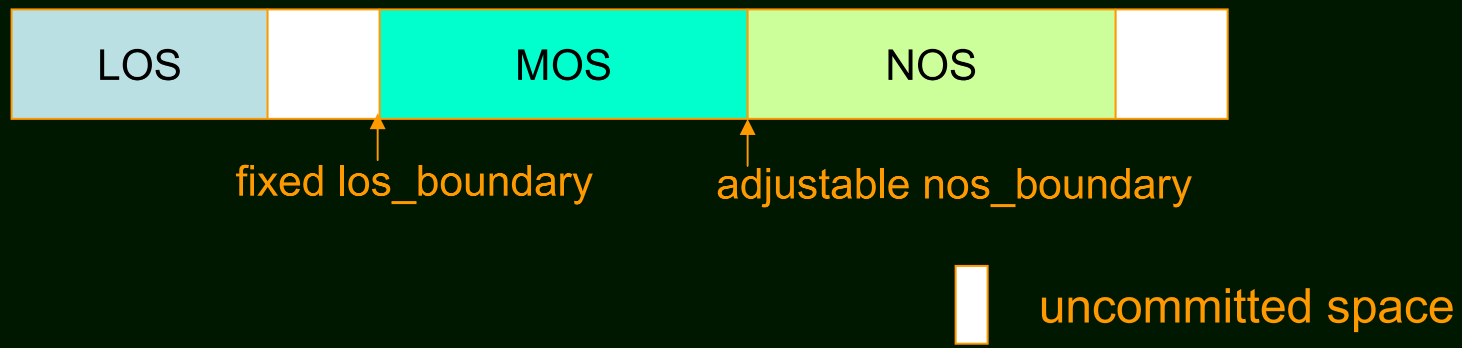


LOS
MOS
NOS
adjustable los_boundary
uncommitted space



LOS extension
compact
LOS
MOS
NOS
los_boundary
New los_boundary
nos_boundary
LOS shrink
compact
LOS
MOS
NOS
New los_boundary
los_boundary
nos_boundary

Harmony GC Default Setting



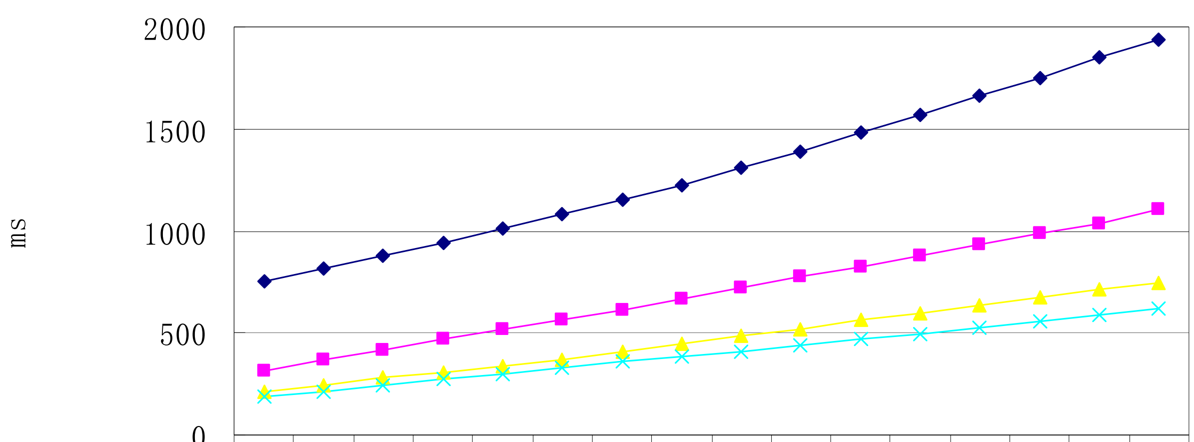
- Switch back to adjustable los_boundary when virtual address space is not enough

Agenda

- LISP2 Sliding Compactor
- Parallel LISP2 Compactor
- Working in Apache Harmony
- Evaluations
- Summary and On-going work

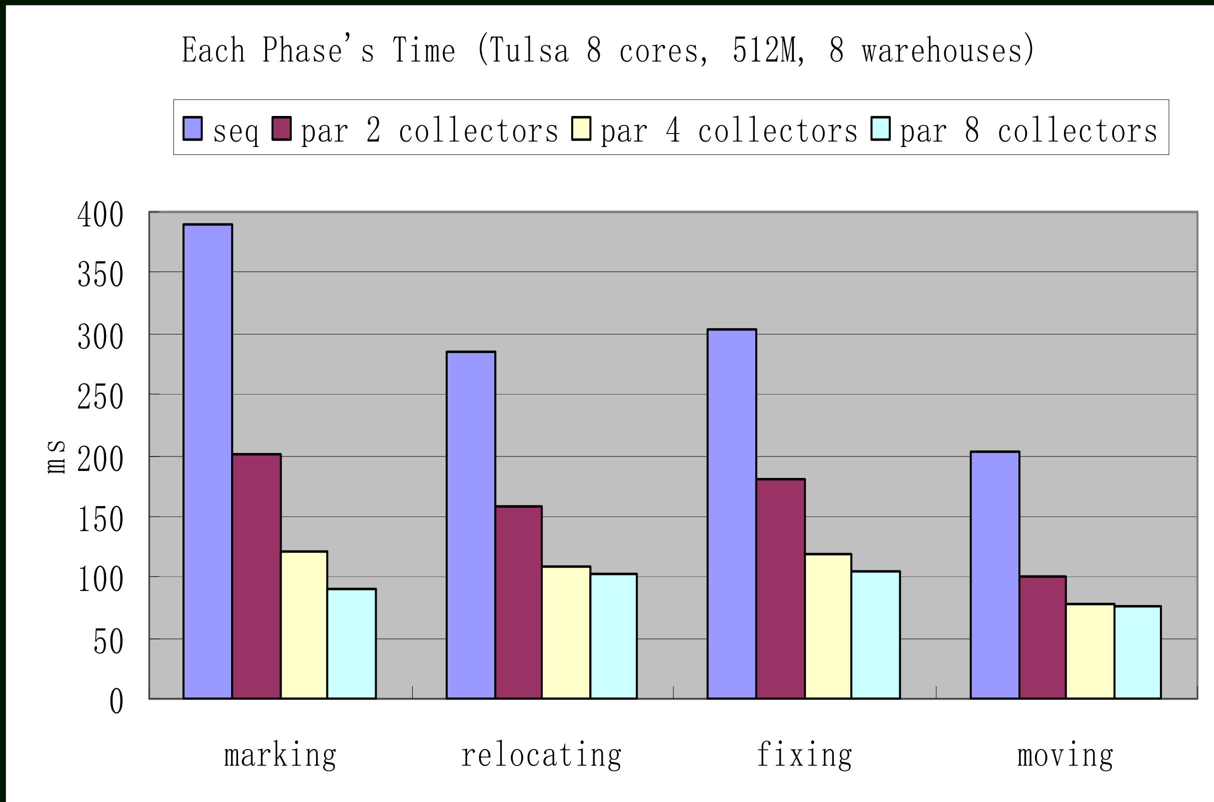
GC Time with SPECJBB2005

SPECjbb2005 GC Time (Tulsa 8 cores, 512M heap)

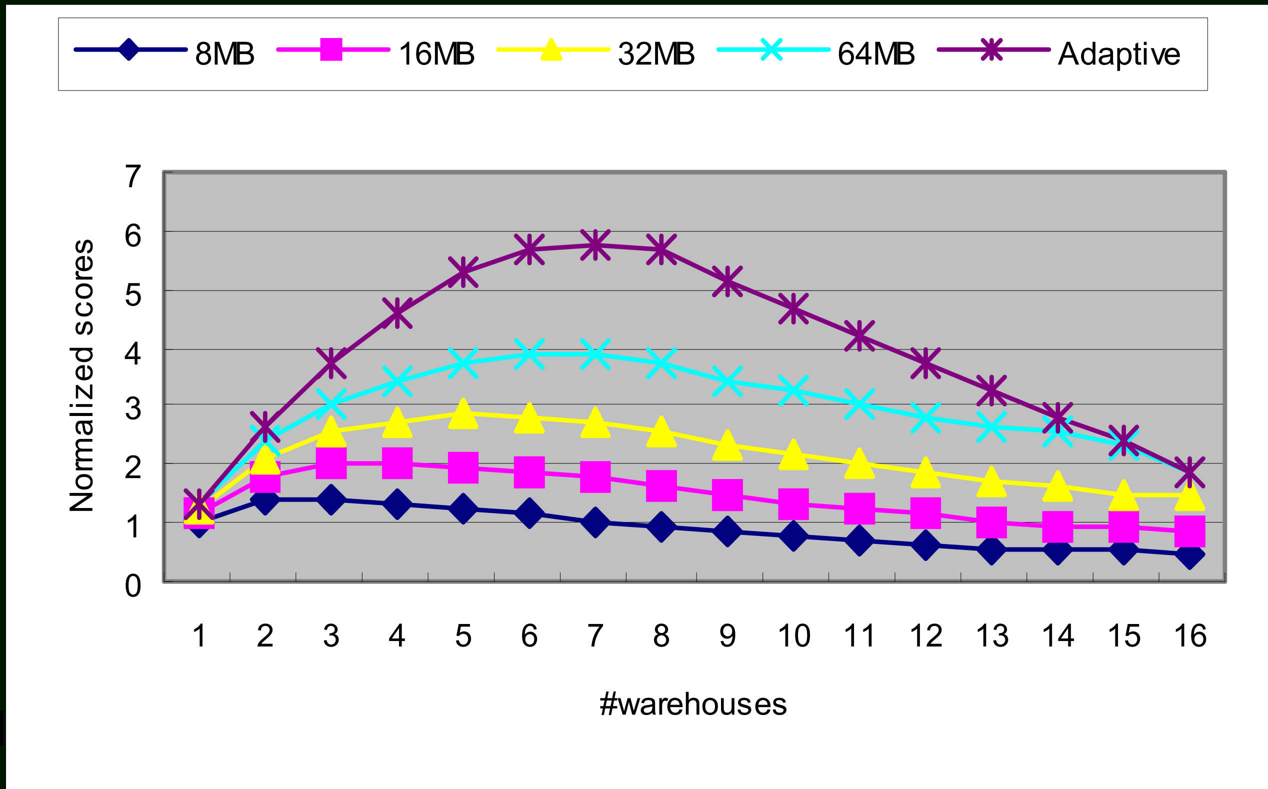


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
seq	751	814	880	944	1012	1082	1152	1227	1307	1392	1483	1569	1663	1752	1848	1941
par 2 collectors	316	367	415	467	516	565	615	664	721	773	823	881	933	989	1038	1103
par 4 collectors	211	242	279	306	341	371	405	445	483	521	563	595	637	671	711	749
par 8 collectors	186	215	242	272	301	329	358	384	410	439	469	497	528	556	589	619

Phase Time with SPECJBB2005



Perf. with Different NOS Size



N

Related Work

- Parallel LISP2 compactor
  - Flood et al, JVM2001
- Three-phase compactor
  - Abuaiadh et al, OOPSLA2004
- Compressor
  - Kermany and Petrank, PLDI2006
- Mapping collector
  - Wegiel and Krintz, ASPLOS2008

Summary

- A parallel LISP2 compactor is proposed
  - Methodology of irregular program parallelization
  - Demonstrated the performance
  - Integrated into Apache Harmony GC toolkit