Concurrent GC
Leveraging Transactional Memory

Phil McGachey
Ali-Reza Adl-Tabatabai
Richard L. Hudson
Vijay Menon
Bratin Saha
Tatiana Shpeisman
Introduction

- Moore’s Law leading to multi-cored chips
- Harder to exploit than raw CPU power
- Concurrent programming becomes common
- Need simplified programming models
Multi-Cored GC

• Stopping the world is unfeasible
• Overhead of pausing operation
• Non-parallel code in collector
• Growing heap sizes
• Clear call for concurrent garbage collection
Concurrency Control

- More threads lead to more interactions
- Locks are already difficult to reason about
- Push for transactional memory
- Transactional Integrity
  - Strong atomicity - TM system responsible
  - Weak atomicity - Programmer responsible
Synergy

- Shared mechanisms
- GC must observe modifications to objects
- TM must detect conflicts
- Barriers are required for both systems
- We leverage the overlap
- Treat “to-space” objects as speculative
Our Goals

- Leverage strong atomicity infrastructure for GC
- Target desktop applications
  - Games, multimedia, VOIP
- Not hard realtime
- Focus on pause time
  - Aim to keep 90% of pauses under 1ms
Implementing Atomicity

- STM with strong atomicity
- No “non-transactional” memory accesses
- Version number for conflict resolution
- Writes increment on commit
- Objects in transactions are write-locked
- Lock can be anonymous
The GC Algorithm

- Don’t stop the world
- Threads paused one at a time
- Minimize work during each pause
- Copy a portion of the heap per GC cycle
- Designed to support parallelism
- 1 GC thread per 10 application threads
- Not necessary for current desktops
Mark Phase

• Pause threads one at a time
• Scan stack area
  • Runtime stacks
  • TM data structures
• Concurrently mark the heap
• Iterate until all reachable objects are marked
• Barrier prevents writes of unmarked pointers
Copy Phase

- Collect small region of heap
- Don’t pause the application
- Copy objects transactionally
- Read barrier follows forwarding pointers
- Write barrier updates pointers
Flip Phase

- Update pointers to forwarded objects
- Pause each thread individually
  - Scan stack area
  - Update forwarded pointers
- Concurrently flip the heap
- Same barriers as the copy phase
Pauses

- Phase changes
- Mark phase
  - Pause each thread to scan stack
  - Pause to guarantee no unmarked objects
- Flip phase
  - Pause to find unflipped pointers on stacks
  - Pause to guarantee no unflipped objects
Concurrent Copying

GC Thread

Application Thread
Concurrent Copying

GC Thread
Begin Copy

Application Thread
Concurrent Copying

GC Thread
- Begin Copy
- Copy Field A

Application Thread
Concurrent Copying

GC Thread
- Begin Copy
- Copy Field A
- Copy Field B

Application Thread
Concurrent Copying

GC Thread
- Begin Copy
- Copy Field A
- Copy Field B
- Copy Field C

Application Thread
- Write to Field A
Concurrent Copying

GC Thread
- Begin Copy
- Copy Field A
- Copy Field B
- Copy Field C
- Write Forwarding Ptr

Application Thread
- Write to Field A
Concurrent Copying

GC Thread
- Begin Copy
- Copy Field A
- Copy Field B
- Copy Field C
- Write Forwarding Ptr

Application Thread
- Write to Field A
- Read from Field A
Atomic Copying

• Copy operation must be atomic
• Wrap each object copy in a transaction
  • Strong atomicity avoids lost update
• Prohibitively expensive
• Build on the STM infrastructure
• Favor application thread in conflicts
Transactional Copying

GC Thread

Application Thread

Store version #
Transactional Copying

GC Thread
- Store version #
- Copy Field A

Application Thread
Transactional Copying

GC Thread
- Store version #
- Copy Field A
- Copy Field B

Application Thread
Transactional Copying

GC Thread

Store version #
Copy Field A
Copy Field B
Copy Field C

Application Thread

Write to Field A
Transactional Copying

GC Thread
- Store version #
- Copy Field A
- Copy Field B
- Copy Field C
- Compare version #

Application Thread
- Write to Field A
Transactional Copying

GC Thread
- Store version #
- Copy Field A
- Copy Field B
- Copy Field C
- Compare version #

Application Thread
- Write to Field A
- Read from Field A
Barrier Synergy

- Strong atomicity barriers:
  - Logs reads and writes
  - Follows forwarding pointers
- Concurrent GC barriers
  - Prevents writes of unmarked pointers
  - Follow forwarding pointers
Experiments

- SPEC JVM98
- SPECjbb2000
- Atomicjbb
- AtomicTSP, AtomicOO7
AtomicJBB
All workloads
## Outliers

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>&lt; 1ms</th>
<th>1..10 ms</th>
<th>10...100 ms</th>
<th>&gt; 100 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>201_compress</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>202_jess</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>209_db</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>213_javac</td>
<td>99.43%</td>
<td>0.57%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>222_mpegaudio</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>227_mtrt</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>_228_jack</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>SPECjbb</td>
<td>99.72%</td>
<td>0.14%</td>
<td>0.14%</td>
<td>0.00%</td>
</tr>
<tr>
<td>AtomicOO7</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>AtomicTSP</td>
<td>100.0%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Atomicjbb</td>
<td>85.00%</td>
<td>12.50%</td>
<td>2.14%</td>
<td>0.36%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>98.92%</td>
<td>0.85%</td>
<td>0.21%</td>
<td>0.02%</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td>≥ 90%</td>
<td>≤ 9%</td>
<td>≤ 0.9%</td>
<td>≤ 0.1%</td>
</tr>
</tbody>
</table>
# Pauses per GC

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Mark Phase</th>
<th>Flip Phase</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>201_compress</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>202_jess</td>
<td>2.6</td>
<td>2.0</td>
<td>4.6</td>
</tr>
<tr>
<td>209_db</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>213_javac</td>
<td>2.7</td>
<td>2.0</td>
<td>4.7</td>
</tr>
<tr>
<td>222_mpegaudio</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>227_mtrt</td>
<td>3.7</td>
<td>2.9</td>
<td>6.6</td>
</tr>
<tr>
<td>_228_jack</td>
<td>2.1</td>
<td>2.0</td>
<td>4.1</td>
</tr>
<tr>
<td>SPECjbb</td>
<td>5.7</td>
<td>2.7</td>
<td>8.4</td>
</tr>
<tr>
<td>AtomicOO7</td>
<td>3.6</td>
<td>2.0</td>
<td>5.6</td>
</tr>
<tr>
<td>AtomicTSP</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>Atomicjbb</td>
<td>4.0</td>
<td>2.0</td>
<td>6.0</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>2.9</strong></td>
<td><strong>2.1</strong></td>
<td><strong>5.1</strong></td>
</tr>
<tr>
<td>Benchmark</td>
<td>Mark Phase</td>
<td>Copy Phase</td>
<td>Flip Phase</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>201_compress</td>
<td>0.19%</td>
<td>0.08%</td>
<td>0.26%</td>
</tr>
<tr>
<td>202_jess</td>
<td>0.91%</td>
<td>0.37%</td>
<td>1.18%</td>
</tr>
<tr>
<td>209_db</td>
<td>0.43%</td>
<td>0.14%</td>
<td>0.44%</td>
</tr>
<tr>
<td>213_javac</td>
<td>0.82%</td>
<td>0.17%</td>
<td>0.82%</td>
</tr>
<tr>
<td>222_mpegaudio</td>
<td>0.22%</td>
<td>0.08%</td>
<td>0.27%</td>
</tr>
<tr>
<td>227_mtrt</td>
<td>1.81%</td>
<td>0.61%</td>
<td>2.02%</td>
</tr>
<tr>
<td>_228_jack</td>
<td>0.77%</td>
<td>0.36%</td>
<td>0.58%</td>
</tr>
<tr>
<td>SPECjbb</td>
<td>1.27%</td>
<td>0.27%</td>
<td>1.02%</td>
</tr>
<tr>
<td>AtomicOO7</td>
<td>0.04%</td>
<td>0.01%</td>
<td>0.03%</td>
</tr>
<tr>
<td>AtomicTSP</td>
<td>0.51%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Atomicjbb</td>
<td>1.37%</td>
<td>0.52%</td>
<td>2.39%</td>
</tr>
<tr>
<td>Average</td>
<td>0.76%</td>
<td>0.24%</td>
<td>0.82%</td>
</tr>
</tbody>
</table>