A Relaxed Consistency based DSM for Asynchronous Parallelism

Keval Vora, Sai Charan Koduru, Rajiv Gupta
Motivation

- Graphs are popular
  - Graph Mining: Community Detection, Coloring
  - Graph Analytics: PageRank, Shortest Paths
- Real-world graphs are large
  - Orkut: 234M edges, 3M vertices
  - LiveJournal: 68M edges, 4.8M vertices
- Processing on distributed memory machines
  - Performance
  - Programmability
Graph Algorithms

› Vertex Centric
  › Computation written for a single vertex
  › Highly parallel execution

› Iterative
  › Terminate when values converge

› Network Bound
  › Computation is simple

Fetch(c)
Fetch(a)
Fetch(b)
c’ = f(c, a, b)
Store(c, c’)
Our Work

- Improve asynchronous execution
  - Make them faster
- Relax consistency to tolerate latencies
  - Tardis: remote fetch is ~2.3 times of local fetch
  - Allow use of stale values
Our Work

- Improve asynchronous execution
  - Make them faster
- Relax consistency to tolerate latencies
  - Tardis: remote fetch is ~2.3 times of local fetch
  - Allow use of stale values

Challenge: Tolerate latencies without delaying convergence
Weak Memory Models

- Delta Consistency [SPAA’97] [PPoPP’03]
  - Controls staleness using static threshold
Weak Memory Models

- Delta Consistency [SPAA’97] [PPoPP’03]
  - Controls staleness using static threshold

![Diagram showing iterations and remote fetches with high and low thresholds]
Weak Memory Models

- Delta Consistency [SPAA’97] [PPoPP’03]
  - Controls staleness using static threshold
Weak Memory Models

- Delta Consistency [SPAA’97] [PPoPP’03]
  - Controls staleness using static threshold

Delayed updates affect convergence
Weak Memory Models

- Delta Consistency [SPAA’97] [PPoPP’03]
  - Controls staleness using static threshold

![Diagram showing the impact of iterations and remote fetches on Delta Consistency with high and low thresholds. The diagram indicates that delayed updates affect convergence.](image-url)
Relaxed Consistency Protocol

- Tracks staleness to exploit it
  - Cached objects have a staleness value
- Best efforts to minimize stale objects
  - Refresh cached objects based on access pattern

- Provides programming support
  - Local writes must be immediately visible
  - Once an object is read by a thread, no earlier writes to it can be read by the same thread
Relaxed Consistency Protocol

- Current-hit
  - object in cache; staleness = 0
- Stale-hit
  - object in cache; 0 < staleness <= t
- Stale-miss
  - object in cache; staleness > t
- Cache-miss
  - object not in cache
Relaxed Consistency Protocol

- **Uncached**
  - Cache-Miss / Write: [Local Node] Staleness = 0
  - Evict: [Local Node]

- **Stale-Miss**
  - Evict: [Local Node]

- **Stale-Hit**
  - [Local Node]

- **Stale**
  - Invalidate: [Directory] ++Staleness

- **Shared**
  - Hit / Write: [Local Node]
  - Evict: [Local Node]

- **Stale**
  - Invalidate: [Directory] ++Staleness

- **Uncached**
  - Read: \(d^o = d^o \cup \{m_i\}\)

- **Shared**
  - Write: \(d^o = d^o \cup \{m_i\}\)
  - Evict: \(d^o = d^o \setminus \{m_i\}\)

- **Refresh**
  - [Local Node] Staleness = 0
Relaxed Consistency Protocol

- **Cache-Miss / Write**
  - [Local Node] Staleness = 0
  - Evict [Local Node]

- **Stale-Miss**
  - [Local Node] Staleness = 0
  - Evict [Local Node]

- **Stale-Hit**
  - [Local Node]

- **Shared**
  - Hit / Write [Local Node]
  - Invalidate [Directory] ++Staleness
  - Refresh [Local Node] Staleness = 0
  - Invalidation [Staleness]

- **Uncached**

- **Refresh**
  - Read
  - Write
  - Evict

- **Stale**
  - Uncached

- **Stale-Hit**

- **Stale-Miss**

- **Evict**

- **Shared**

- **Hit / Write**
  - [Local Node]

- **Write**
  - $d^0 = d^0 \cup \{m_i\}$

- **Evict**
  - $d^0 = d^0 \setminus \{m_i\}$

- **Refresh**
  - $d^0 = d^0 \cup \{m_i\}$

- **Stale-Hit**
  - [Local Node]
Relaxed Consistency Protocol

- **Uncached**
  - Evict [Local Node]
  - Stale-Miss [Local Node] Staleness = 0

- **Stale**
  - Invalidation [Directory] ++Staleness

- **Stale-Hit** [Local Node]
  - Evict [Local Node]

- **Shared**
  - Cache-Miss / Write [Local Node] Staleness = 0
  - Hit / Write [Local Node]
  - Invalidate [Directory] ++Staleness
  - Refresh [Local Node] Staleness = 0
  - Read d₀ = d₀ U {mᵢ}
  - Write d₀ = d₀ U {mᵢ}
  - Evict d₀ = d₀ \ {mᵢ}

- **Evict** [Local Node]
Relaxed Consistency Protocol

- **Cache-Miss / Write**
  - [Local Node]
  - Staleness = 0
  - Evict
  - [Local Node]
- **Hit / Write**
  - [Local Node]
  - Stale
  - Hit / Write
  - [Local Node]
  - Invalidate
  - [Directory]
  - ++Staleness
- **Read**
  - \( d^0 = d^0 \cup \{m_i\} \)
- **Write**
  - \( d^0 = d^0 \cup \{m_i\} \)
  - Evict
  - \( d^0 = d^0 \setminus \{m_i\} \)
- **Stale-Miss**
  - [Local Node]
  - Invalidate
  - [Directory]
  - ++Staleness
- **Stale-Hit**
  - [Local Node]
  - Stale
  - Refresh
  - [Local Node]
  - Staleness = 0
- **Uncached**
  - Evict
  - [Local Node]
  - Staleness = 0
Relaxed Consistency Protocol

- **Uncached**
  - Evict [Local Node]
  - Stale-Miss [Local Node] Staleness = 0

- **Stale**
  - Invalidation [Directory] ++Staleness
  - Stale-Hit [Local Node]
  - Evict [Local Node] Staleness = 0

- **Shared**
  - Hit / Write [Local Node]
  - Read $d^o = d^o \cup \{m_i\}$
  - Write $d^o = d^o \cup \{m_i\}$
  - Evict $d^o = d^o \setminus \{m_i\}$
  - Invalidation [Directory] ++Staleness

- **Hit / Write** [Local Node]
  - Staleness = 0

- **Cache-Miss / Write** [Local Node]
  - Staleness = 0

- **Stale**
  - Refresh [Local Node] Staleness = 0

- **Evict**
  - [Local Node]
  - Staleness = 0
Relaxed Consistency Protocol

Cache-Miss / Write
[Local Node]
Staleness = 0

Evict
[Local Node]

Stale-Miss
[Local Node]
Staleness = 0

Invalidate
[Directory]
++Staleness

Hit / Write
[Local Node]

Hit / Write
[Local Node]

Write
\[d^0 = d^0 \cup \{m_i\}\]

Read
\[d^0 = d^0 \setminus \{m_i\}\]

Refresh
[Local Node]
Staleness = 0

Evict
[Local Node]

Stale-Hit
[Local Node]

Invalidate
[Directory]
++Staleness

Stale
[Local Node]
Staleness = 0

Shared

Uncached
Relaxed Consistency Protocol

- **Uncached**
  - Evict [Local Node]
  - Staleness = 0

- **Stale**
  - Evict [Local Node]
  - Staleness = 0

- **Shared**
  - Hit / Write [Local Node]
  - Hit / Write [Local Node]

- **Invalidate** [Directory]
  - ++Staleness

- **Read**
  - $d^0 = d^0 \cup \{m_i\}$

- **Write**
  - $d^0 = d^0 \cup \{m_i\}$

- **Refresh** [Local Node]
  - Staleness = 0

- **Evict** [Local Node]
  - Stale-Hit

- **Stale-Miss** [Local Node]
  - Staleness = 0
  - Invalidate [Directory]
Relaxed Consistency Protocol

**Cache-Miss / Write**
- [Local Node]
  - Staleness = 0
- Evict
  - [Local Node]

**Hit / Write**
- [Local Node]

**Read**
- $d^0 = d^0 \cup \{m_i\}$

**Write**
- $d^0 = d^0 \cup \{m_i\}$

**Stale-Miss**
- [Local Node]
  - Staleness = 0
- Evict
  - [Local Node]

**Stale-Hit**
- [Local Node]

**Stale**
- [Local Node]
  - Staleness = 0
- Invalidate
  - [Directory]
    - ++Staleness
- Refresh
  - [Local Node]
    - Staleness = 0
- Evict
  - [Local Node]

**Stale**
- [Local Node]

**Shared**
- [Local Node]
  - Staleness = 0

**Evict**
- [Local Node]

**Invalidate**
- [Directory]
  - ++Staleness
Relaxed Consistency Protocol

- **Uncached**
  - Evict [Local Node]
  - Stale-Miss [Local Node] Staleness = 0

- **Stale**
  - Invalidation [Directory]
  - ++Staleness

- **Shared**
  - Cache-Miss / Write [Local Node] Staleness = 0
  - Hit / Write [Local Node]

- **Stale-Hit** [Local Node]
  - Invalidation [Directory]
  - ++Staleness

- **Hit** / Write [Local Node]
  - Evict

- **Read**
  - $d^0 = d^0 \cup \{m_i\}$
  - Evict $d^0 = d^0 \setminus \{m_i\}$

- **Write**
  - $d^0 = d^0 \cup \{m_i\}$

- **Refresh**
  - [Local Node] Staleness = 0
  - Evict

- **Invalidate**
  - [Directory]
Relaxed Consistency Protocol

Cache-Miss / Write
[Local Node]
Staleness = 0

Hit / Write
[Local Node]

Read
$d^0 = d^0 \cup \{m_i\}$

Write
$d^0 = d^0 \cup \{m_i\}$

Stale-Hit
[Local Node]

Stale-Miss
[Local Node]
Staleness = 0

Invalidate
[Directory]
++Staleness

Evict
[Local Node]

Shared

Refresh
[Local Node]
Staleness = 0

Invalidate
[Directory]
++Staleness

Stale

Evict
[Local Node]

Stale

Evict
[Local Node]

Uncached
Relaxed Consistency Protocol

**Cache-Miss / Write**
- [Local Node] Staleness = 0
- Evict [Local Node]

**Stale-Miss**
- [Local Node] Staleness = 0
- Evict [Local Node]
- Invalidate [Directory] ++Staleness

**Shared**
- Hit / Write [Local Node]
- Hit / Write [Local Node]
- Invalidate [Directory] ++Staleness
- Evict [Local Node]

**Stale**
- Stale-Hit [Local Node]
- Invalidate [Directory] ++Staleness

**Uncached**
- Evict [Local Node]

**Stale-Hit**
- [Local Node]

**Uncached**

**Shared**
- Read $d^o = d^o \cup \{m_i\}$
- Write $d^o = d^o \cup \{m_i\}$
- Evict $d^o = d^o \setminus \{m_i\}$

**Stale**
- Stale-Hit [Local Node]

**Stale**
- Stale-Hit [Local Node]

**Stale**
- Stale-Hit [Local Node]

**Stale**
- Stale-Hit [Local Node]

**Stale**
- Stale-Hit [Local Node]

**Stale**
- Stale-Hit [Local Node]
Relaxed Consistency Protocol

- Uncached
  - Evict [Local Node]
  - Stale-Miss [Local Node]
  - Stale-Hit [Local Node]

- Stale
  - Evict [Local Node]
  - Invalidation [Directory]
  - Staleness = 0

- Stale-Hit
  - Evict [Local Node]

- Stale-Miss
  - Evict [Local Node]
  - Staleness = 0

- Shared
  - Hit / Write [Local Node]
  - Hit / Write [Local Node]
  - Refresh [Local Node]
  - Staleness = 0

- Read
  - d₀ = d₀ U {m₁}
  - Evict d₀ = d₀ \ {m₁}

- Write
  - d₀ = d₀ U {m₁}

- Invalidation
  - [Directory]
  - ++Staleness

- Invalidation
  - [Directory]
  - ++Staleness
Implementation

- Similar to dyDSM [Koduru et al. 2013]
  - Object based
  - Protocol relaxes strict consistency
  - Graphs are partitioned using METIS [SISC 99]

- Runtime
  - Single Writer Model
  - Refresher threads block on refresh-queues
  - Compute threads populate refresh-queues
Performance

Pokec:
30M edges
1.6M vertices

AtmosModl:
10M edges
1.4M vertices

RCP 48.7% faster than SCP and 56% faster than best Stale-n
Performance

RCP blocks for 41% of remote fetches

Best Stale-n blocks for 85% of remote fetches
RCP requires 49% more iterations.

Stale-2/Stale-3 require 146/176% more iterations.
Performance

97.4% of values have staleness 0; 2.2% of values have staleness 1
RCP performs better for non power-law graphs

- RCP is orthogonal to GraphLab

<table>
<thead>
<tr>
<th></th>
<th>SSSP</th>
<th>PR</th>
<th>GC</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orkut</td>
<td>RCP</td>
<td>161.88</td>
<td>822.95</td>
<td>92.79</td>
</tr>
<tr>
<td></td>
<td>GL</td>
<td>239.4</td>
<td>829.3</td>
<td>248.66</td>
</tr>
<tr>
<td>Live-Journal</td>
<td>RCP</td>
<td>21.73</td>
<td>343.96</td>
<td>17.44</td>
</tr>
<tr>
<td></td>
<td>GL</td>
<td>15.7</td>
<td>295.1</td>
<td>x</td>
</tr>
<tr>
<td>Pokec</td>
<td>RCP</td>
<td>9.47</td>
<td>169.47</td>
<td>8.81</td>
</tr>
<tr>
<td></td>
<td>GL</td>
<td>8.7</td>
<td>159.9</td>
<td>173.47</td>
</tr>
<tr>
<td>Higgs-Twitter</td>
<td>RCP</td>
<td>2.50</td>
<td>15.64</td>
<td>3.60</td>
</tr>
<tr>
<td></td>
<td>GL</td>
<td>5.5</td>
<td>x</td>
<td>263.45</td>
</tr>
</tbody>
</table>
Conclusion

- Relaxing consistency is useful
  - With controlled use of staleness
- Prior DSMs:
  - Efficient (delta coherence & strict consistency)
- Graph Processing Frameworks
  - Easier to code (Pregel, GraphLab & PowerGraph)