ASPIRE
Exploiting Asynchronous Parallelism in Iterative Algorithms using a Relaxed Consistency based DSM

Keval Vora, Sai Charan Kodurru, Rajiv Gupta

OOPSLA’ 14 – Portland, Oregon
October 24, 2014
Motivation

- Iterative algorithms
  - PDE Solvers: Heat Simulation
  - Graph Analysis: PageRank, Community Detection
- Real-world datasets are typically large
  - Social Networks, Genome Graphs
- Processing on distributed memory machines
  - Performance
  - Programmability
Outline

- Iterative Algorithms
- Overview of ASPIRE
- Existing Weak Memory Models
- Relaxed Consistency Protocol
- Evaluation
- Conclusion
Iterative Algorithms

- Data Centric
  - Computation written for a single element
  - Terminate when values converge
  - Highly parallel execution

- Network Bound
  - Computation is simple

```
Fetch(c)
Fetch(a)
Fetch(b)
c' = f(c, a, b)
Store(c, c')
```
Iterative Algorithms

- Data Centric
  - Computation written for a single element
  - Terminate when values converge
  - Highly parallel execution

- Network Bound
  - Computation is simple

```
Fetch(c)
Fetch(a)
Fetch(b)
c' = f(c, a, b)
Store(c, c')
```
Execution Models

- Bulk Synchronous Parallel (BSP) [CACM’90]
  - Disjoint computation and communication
  - Computation based on previous iteration
- Asynchronous Parallelism [Baudet 1978]
  - Overlap computation and communication
  - Computation based on current iteration
  - Known to be faster than BSP
ASPIRE

- Improve asynchronous execution
  - Make them faster
- Tolerate network latencies
  - Tardis: remote fetch is ~2.3 times of local fetch
- Relaxing consistency
  - Allow use of stale values
- Without affecting convergence
  - Minimize use of stale values
ASPIRE

- Improve asynchronous execution
  - Make them faster

- Tolerate network latencies

- Tardis: remote fetch is ~2.3 times of local fetch

- Relaxing consistency
  - Allow use of stale values
  - Without affecting convergence
  - Minimize use of stale values

**Challenge:** Relax consistency without delaying convergence
Weak Memory Models

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

![Diagram]

- Remote Fetches
- Iterations
- High Threshold
- Low Threshold
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
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

Delayed updates affect convergence
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**
- **Evict**
- **Stale-Miss**
  - Staleness = 0
- **Stale-Hit**
  - Staleness = 0
- **Shared**
- **Hit / Write**
  - [Local Node]
- **Invalidate**
  - [Directory]
  - ++Staleness
- **Stale**
- **Invalidate**
  - [Directory]
  - ++Staleness
- **Stale-Miss**
  - [Local Node]
- **Evict**
  - [Local Node]
- **Read**
  - $d^o = d^o \cup \{m_i\}$
- **Write**
  - $d^o = d^o \cup \{m_i\}$
- **Refresh**
  - [Local Node]
  - Staleness = 0
- **Evict**
  - [Local Node]
Relaxed Consistency Protocol

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

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

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

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

- **Evict**
  - Shared
  - [Local Node]

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

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

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

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

- **Uncached**
  - Evict [Local Node]
  - Stale-Miss [Local Node] Staleness = 0
- **Evict** [Local Node]
- **Stale-Miss** [Local Node] Staleness = 0
  - Invalidation [Directory] ++Staleness
- **Stale** [Local Node] Staleness = 0
  - Invalidation [Directory] ++Staleness
- **Stale-Hit** [Local Node]
- **Shared**
  - Hit / Write [Local Node]
  - Uncached
  - Shared
- **Refresh** [Local Node] Staleness = 0
  - Invalidation [Directory] ++Staleness
- **Read**
  - $d^o = d^o \cup \{m_i\}$
- **Stale-Miss** [Local Node] Staleness = 0
  - Evict [Local Node]
  - Invalidation [Directory] ++Staleness
- **Evict** [Local Node]
Relaxed Consistency Protocol

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

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

- **Stale**
  - Staleness = 0

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

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

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

- **Evict**
  - [Local Node]

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

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

- **Evict**
  - $d^o = d^o \setminus \{m_i\}$
Relaxed Consistency Protocol

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

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

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

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

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

- **Hit**: 
  - Write 
    - \( d^o = d^o \cup \{m_i\} \) 
  - Read 
    - \( d^o = d^o \cup \{m_i\} \) 
  - Evict 
    - \( d^o = d^o \setminus \{m_i\} \)
Relaxed Consistency Protocol

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

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

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

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

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

- **Evict** [Local Node]

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

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

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

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

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

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

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

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

- **Shared**

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

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

- **Invalidated**
  - [Directory]
  - ++Staleness

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

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

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

Cache-Miss / Write

Staleness = 0

Hit / Write

Staleness = 0

Read

\[ d^o = d^o \cup \{m_i}\]
Relaxed Consistency Protocol

- **Cache-Miss / Write**
  - [Local Node]
  - Staleness = 0
  - Evict
  - Stale-Miss
    - [Local Node]
    - Staleness = 0
- **Hit / Write**
  - [Local Node]
  - Hit
  - Shared
  - Stale
    - Stale-Hit
      - [Local Node]
      - Invalidation
        - [Directory]
        - ++Staleness
- **Read**
  - d^o = d^o \ U \ {m_i}
  - Evict
  - Shared
- **Write**
  - d^o = d^o \ U \ {m_i}
  - Refresh
    - [Local Node]
    - Staleness = 0
    - Invalidation
      - [Directory]
      - ++Staleness

- **Evict**
  - [Local Node]
  - Stale
  - [Local Node]
  - Hit
  - Shared
  - Stale
  - Invalidation
    - [Directory]
    - ++Staleness
Relaxed Consistency Protocol

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

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

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

- **Stale**
  - Invalidate [Directory]
  - Staleness = 0

- **Shared**
  - Hit / Write [Local Node]
  - Write $d^o = d^o \cup \{m_i\}$

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

- **Refresh** [Local Node]
  - $d^o = d^o \cup \{m_i\}$

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

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

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

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

- Runtime
  - Single Writer Model
  - Refresh strategy using producer-consumer model
  - Termination semantics
Experimental Setup

- Tardis @ UCR
  - 16-node cluster running CentOS 6.3
- 8 benchmarks
  - WS, HS, PR, SSSP, CC, CD, NP, GC
- 10 real-world inputs
  - SNAP dataset collection
  - UFL sparse matrix collection
Execution Time

Pokec:
- 30M edges
- 1.6M vertices

AtmosModl:
- 10M edges
- 1.4M vertices

RCP 48.7% faster than SCP+RW and 56% faster than best Stale-n
Remote Fetches

RCP blocks on 41.8% of remote fetches (7.5% for PR, WS & HS)

Best Stale-n blocks on 85.6% of remote fetches
Iterations

RCP requires 49.5% more iterations

Stale-2/Stale-3 require 146/176% more iterations
Staleness Percentage

97.4% of values have staleness 0; 2.2% of values have staleness 1
Graph Processing Frameworks

- Distributed memory
  - GraphLab [VLDB’12], Pregel [SIGMOD’10], PowerGraph [OSDI’12]

- Shared memory
  - Ligra [PPoPP’13], Grace [CIDR’13], Galois[PLDI’07]

- Out-of-core
  - GraphChi [OSDI’12], X-Stream [SOSP’13]
GraphLab

- RCP compares favorably with GraphLab
- RCP is orthogonal to these frameworks
  - Can be used in their asynchronous engines
Other Experiments

- Different inputs
- Overhead study
- Design choices
  - Invalidates v/s updates
  - Sensitivity to object size
  - Sensitivity to writes
  - Sensitivity to communication delay
- Different cluster sizes
Conclusion

- Relaxing consistency is useful
  - With controlled use of staleness
  - Using refresh strategy
- Efficient than prior DSMs based on
  - Strict consistency
  - Delta consistency
- Processing Frameworks
  - Easier to code (Pregel)
  - Compares favorably (GraphLab)
Thanks

› GRASP
  › http://grasp.cs.ucr.edu

› Acknowledgements