

Effective Prefetch for Mark-Sweep GC

Robin Garner, Steve Blackburn, Daniel Frampton

Department of Computer Science
Australian National University



Talk Outline

- Tracing
- Prefetching
- Edge enqueueing
- Effective prefetch
- Results
- Other issues

Tracing Fundamentals

- Used by most collectors liveness
- Transitive closure from rootset
 - Trace visits:
 - **mark state** for each live object
 - **references** from each live object
 - Trace requires a **work set** (queue, deque, stack)
 - May determine visit order (DFS, BFS)
 - Classically holds unprocessed (gray) objects
- We ignore for simplicity:
 - Copying
 - Partial tracing (eg generations)

Tracing Loop

```
workset = rootset
```

```
while workset.hasEntry()  
    node = workset.remove()
```

```
    gcmmap = node.getGcMap()  
    for p in gcmmap.pointers  
        child = node.fields[p]
```

```
        if child.testAndMark()  
            workset.add(child)
```

Tracing Loop: Workset

Workset operations should (do) exhibit very good locality

```
workset = rootset
```

```
while workset.hasEntry()  
    node = workset.remove()  
  
    gcmmap = node.getGcMap()  
    for p in gcmmap.pointers  
        child = node.fields[p]  
  
        if child.testAndMark()  
            workset.add(child)
```

Tracing Loop: Traversal

Known to exhibit poor locality

```
workset = rootset
```

```
while workset.hasEntry()  
    node = workset.remove()
```

```
    gcmmap = node.getGcMap() // node metadata
```

```
    for p in gcmmap.pointers  
        child = node.fields[p] // node
```

```
        if child.testAndMark() // child metadata  
            workset.add(child)
```

Dense Mark State

Old idea: use bitmap to improve mark locality

```
workset = rootset

while workset.hasEntry()
    node = workset.remove()

    gcmmap = node.getGcMap() // node metadata
    for p in gcmmap.pointers
        child = node.fields[p] // node

        if child.testAndMark() // child metadata
            workset.add(child)
```

Dense Mark State

- We saw **no** performance advantage
- Why
 - Only effective if another object marked before metadata line displaced from cache
 - Side metadata strictly increases memory traffic
 - Cache traffic displaces metadata
 - Object scanning
 - Field touches
 - Work set operations (adding and removing)
- See paper for details

Prefetching

```
workset = rootset
```

```
while workset.hasEntry()  
    workset.prefetch(P)  
    node = workset.remove()
```

```
    gcmmap = node.getGcMap()           // node metadata
```

```
    for p in gcmmap.pointers  
        child = node.fields[p]       // node
```

```
        if child.testAndMark()       // child metadata  
            workset.add(child)
```

Prefetching: Boehm [ISMM2000]

Prefetch child before pushing

```
workset = rootset
```

```
while workset.hasEntry()  
    node = workset.remove()
```

```
    gcmmap = node.getGcMap()      // node metadata  
    for p in gcmmap.pointers  
        child = node.fields[p]   // node
```

```
        if child.testAndMark()   // child metadata  
            prefetch(child)  
            workset.add(child)
```

Prefetching: Reality #1

- Work sets are typically stacks
 - Cannot readily predict prefetch distance
- Cher et al [ASPLOS 04] solve this:
 - Introduce small FIFO in front of stack
 - Pop stack: a) prefetch and b) insert into FIFO
 - Dequeue from FIFO for use
 - FIFO depth implies prefetch distance
 - Results still not great

Prefetching: Reality #2

- Node and child accesses intermingled

```
workset = rootset
```

```
while workset.hasEntry()  
    workset.prefetch(P)  
    node = workset.remove()
```

```
gcmmap = node.getGcMap() // node metadata
```

```
for p in gcmmap.pointers  
    child = node.fields[p] // node
```

```
if child.testAndMark() // child metadata  
    workset.add(child)
```

Node & Edge Enqueueing

```
workset = rootset
```

```
while workset.hasEntry()  
  workset.prefetch(P)  
  node = workset.remove()
```

```
gcmmap = node.getGcMap()  
for p in gcmmap.pointers  
  child = node.fields[p]
```

```
if child.testAndMark()  
  workset.add(child)
```

holds unprocessed **nodes**

```
workset = rootset
```

```
while workset.hasEntry()  
  workset.prefetch(P)  
  node = workset.remove()
```

```
if node.testAndMark()  
  gcmmap = node.getGcMap()  
  for p in gcmmap.pointers  
    child = node.fields[p]  
    workset.add(child)
```

holds unprocessed **edges**

- More workset work

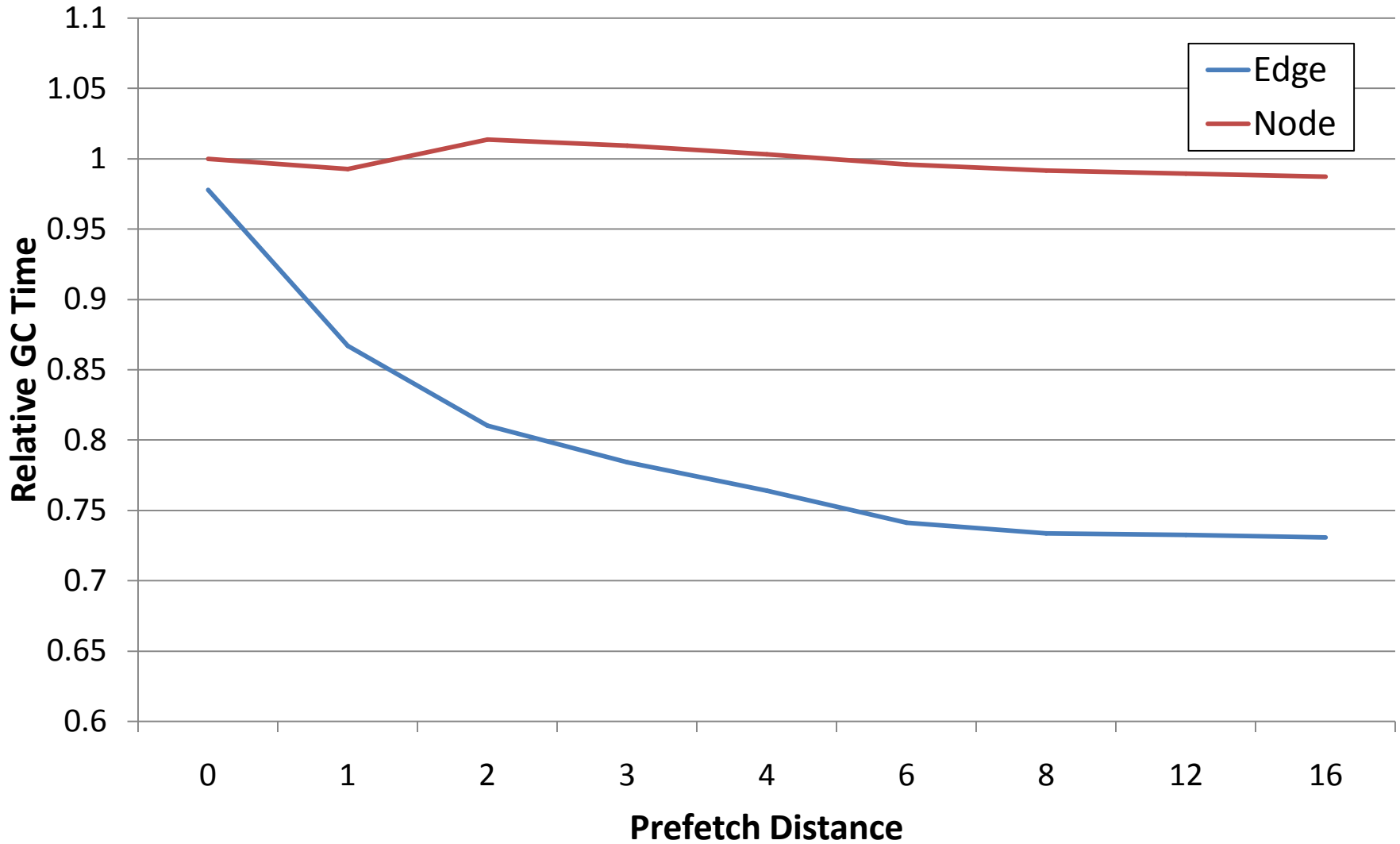
Edge Prefetching

```
workset = rootset

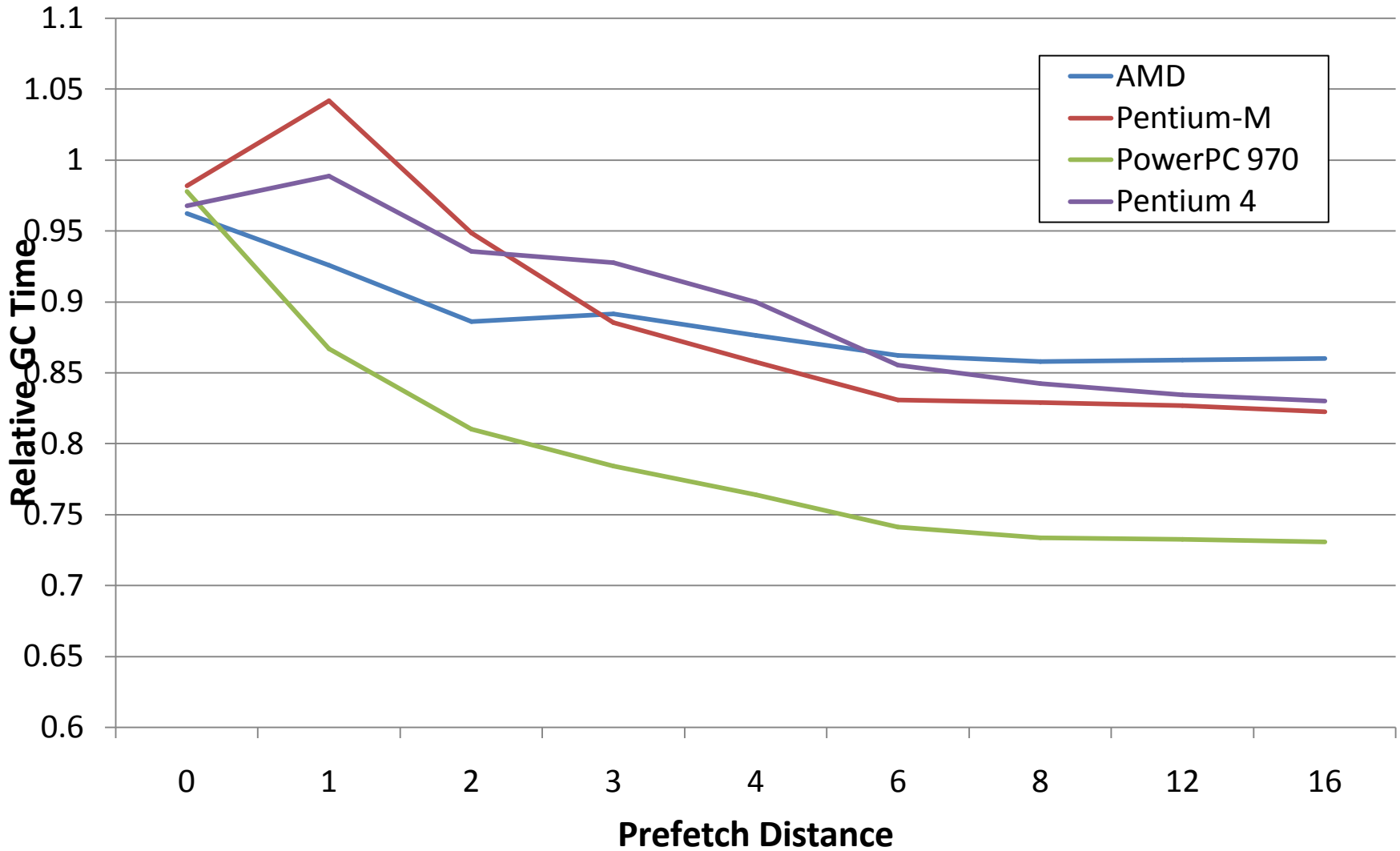
while workset.hasEntry()
  workset.prefetch(P)
  node = workset.remove()

  if node.testAndMark()           // node metadata
    gcmmap = node.getGcMap()      // node metadata
    for p in gcmmap.pointers
      child = node.fields[p]     // node
      workset.add(child)
```

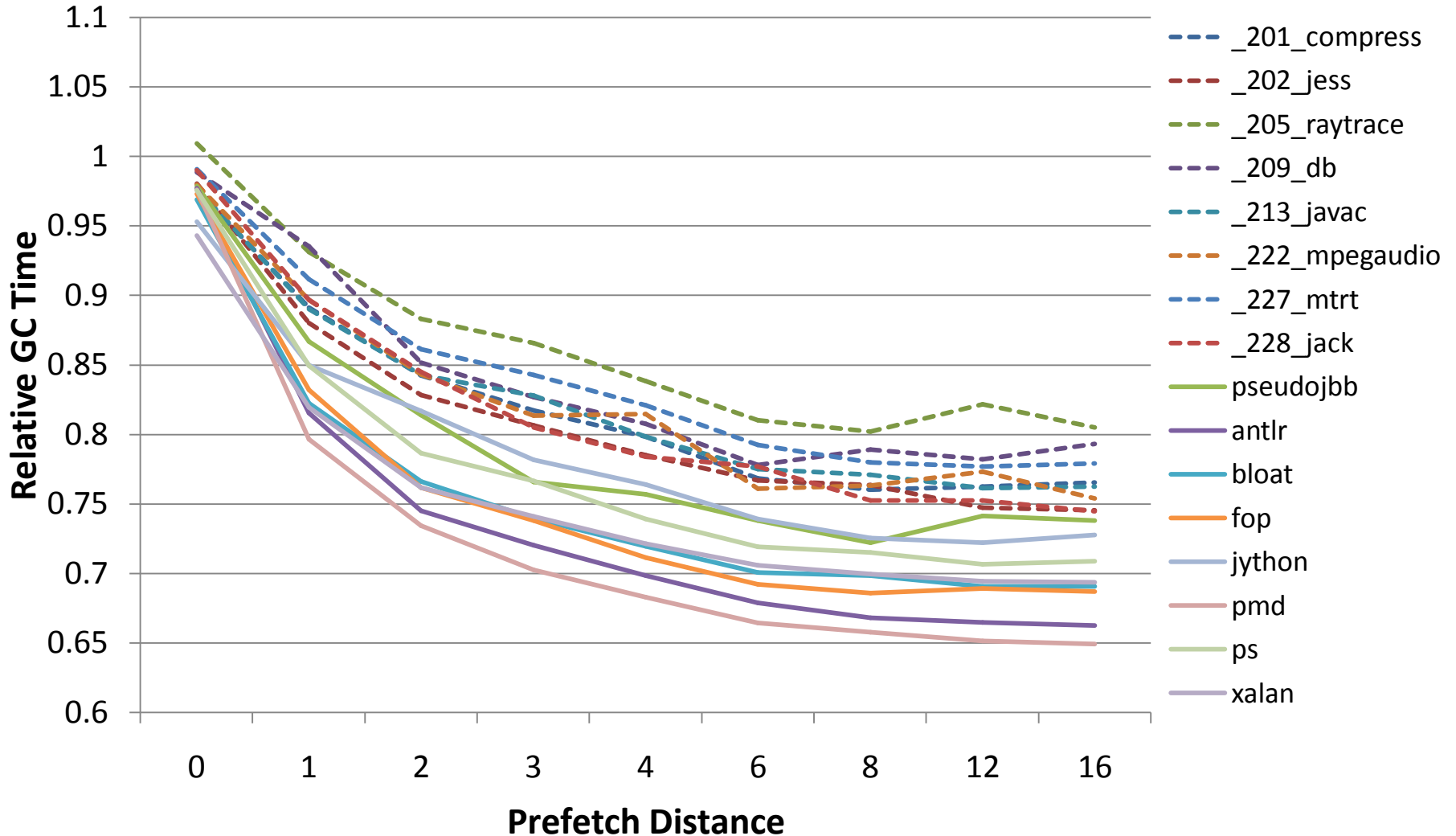
PPC970 GC Time



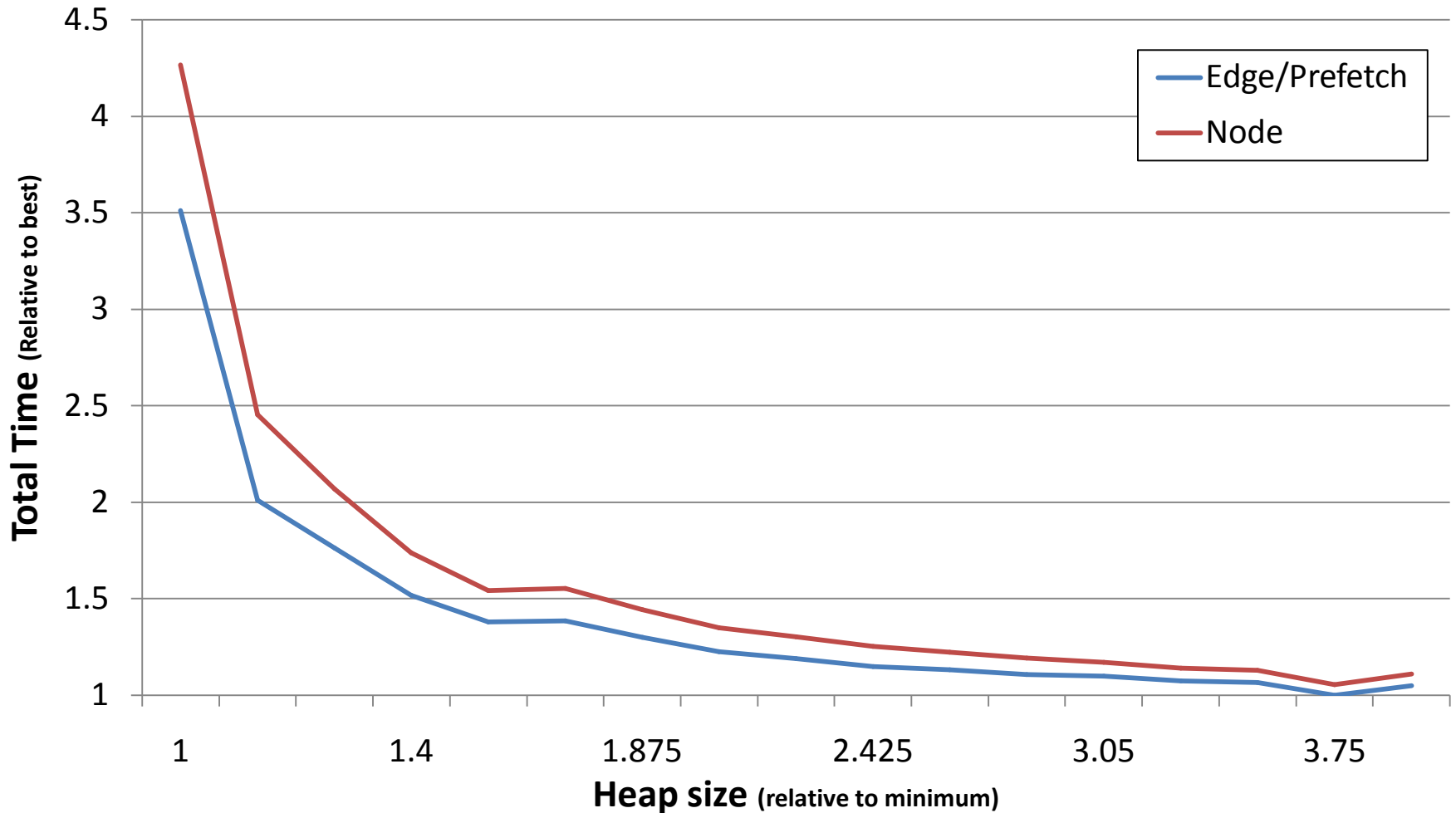
Performance by Architecture



PPC970 Benchmark Performance



PPC970 Total Time



Bottom Line

- Additional workset operations
 - More edges than nodes (1.7 per node)
- Improved locality
 - Exposes opportunities for prefetching
- Mean GC speedup of 22%, (min 17%, max 24%)

Also In The Paper

- Replay tracing
 - Comprehensive analysis methodology
- Comprehensive Evaluation
 - 4 architectures, 16 benchmarks
- Observations
 - Dense metadata gives no advantage
 - FIFO *appears* to help hardware load speculation
 - Prefetching *very* sensitive to quality of tracing loop
 - Poor tracing loop masks memory cost

Future Work

- Copying
 - Can prefetch be effective for copying GC?
 - Prerequisite for nursery collection

Conclusions

- Combined, edge enqueueing and FIFO prefetch yield significant tracing improvements
- Replay tracing allows detailed analysis of tracing performance
- New observations
 - Side metadata does not provide locality advantage

Questions?

- Patch available:

<http://cs.anu.edu.au/~Steve.Blackburn/pubs/abstracts.html>