Leak Pruning

Michael Bond Kathryn McKinley

The University of Texas at Austin

Deployed Software Fails

Deployed Software Fails

- Driverless truck
  - 10,000 lines of C#

http://www.codeproject.com/KB/showcase/IfOnlyWedUsedANTSProfiler.aspx

Deployed Software Fails

- Driverless truck
  - 10,000 lines of C#
- Leak: past obstacles remained reachable

http://www.codeproject.com/KB/showcase/IfOnlyWedUsedANTSProfiler.aspx

Deployed Software Fails

- Driverless truck
  - 10,000 lines of C#
- Leak: past obstacles remained reachable
- No immediate symptoms

  "This problem was pernicious because it only showed up after 40 minutes to an hour of driving around and collecting obstacles."

http://www.codeproject.com/KB/showcase/IfOnlyWedUsedANTSProfiler.aspx

Deployed Software Fails

- Driverless truck
  - 10,000 lines of C#
- Leak: past obstacles remained reachable
- No immediate symptoms

  "This problem was pernicious because it only showed up after 40 minutes to an hour of driving around and collecting obstacles."
- Quick "fix": restart after 40 minutes

http://www.codeproject.com/KB/showcase/IfOnlyWedUsedANTSProfiler.aspx

Deployed Software Fails

- Driverless truck
  - 10,000 lines of C#
- Leak: past obstacles remained reachable
- No immediate symptoms

  "This problem was pernicious because it only showed up after 40 minutes to an hour of driving around and collecting obstacles."
- Quick "fix": restart after 40 minutes
- Environment sensitive
  - More obstacles in deployed setting
  - Unresponsive after 28 minutes

http://www.codeproject.com/KB/showcase/IfOnlyWedUsedANTSProfiler.aspx

Deployed Software Fails

- Driverless truck
  - 10,000
- Leak: pa hable
- No immediate symptoms

  "This problem was p wed up after 40 minutes to an hour of driving cles."
- Quick "fix": nutes
- Environment sensitive
  - More obst g
  - Unrespons

Different environments & inputs → different behavior

Hard to fix before deployment

Uncertainty in deployed systems

Tolerating Memory Leaks

- Deployed systems have leaks
  - Critical systems need immediate help
- **Leak pruning** tolerates bad effects
  - Reclaims memory automatically
  - High precision & low overhead
  - Bounds resources
  - Preserves semantics

Outline

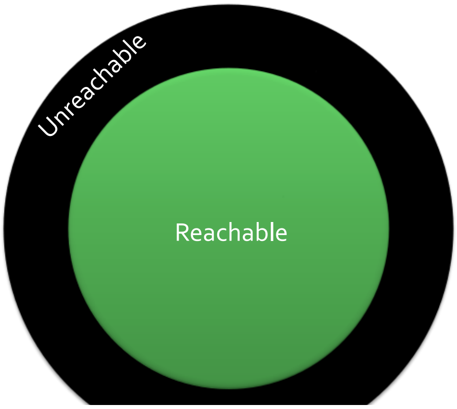
- Why tolerate leaks
- Why leaks are so bad
- How leak pruning works
- How leak pruning predicts leaked objects
- Evaluation

Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them

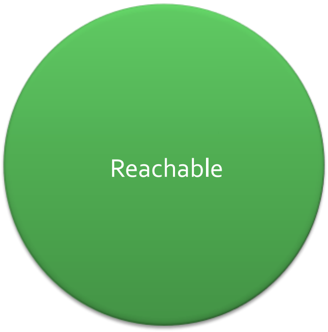
Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them



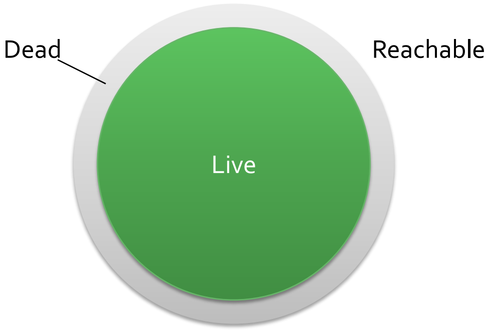
Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them



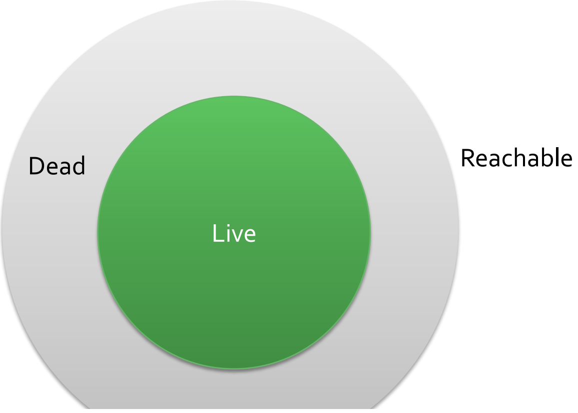
Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them



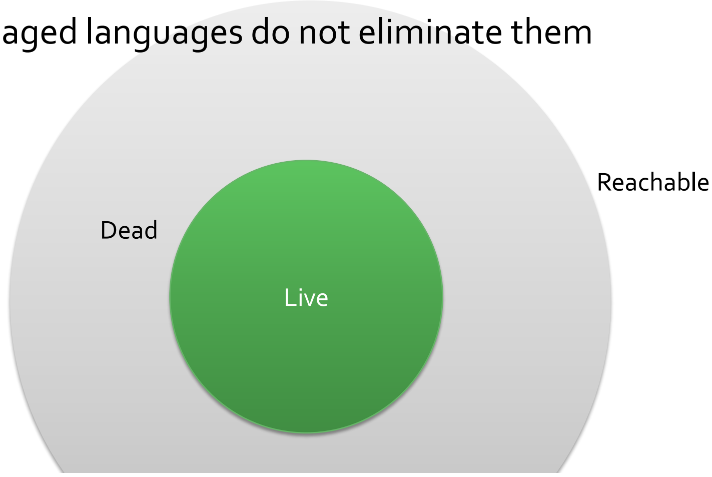
Memory Leaks in Deployed Systems

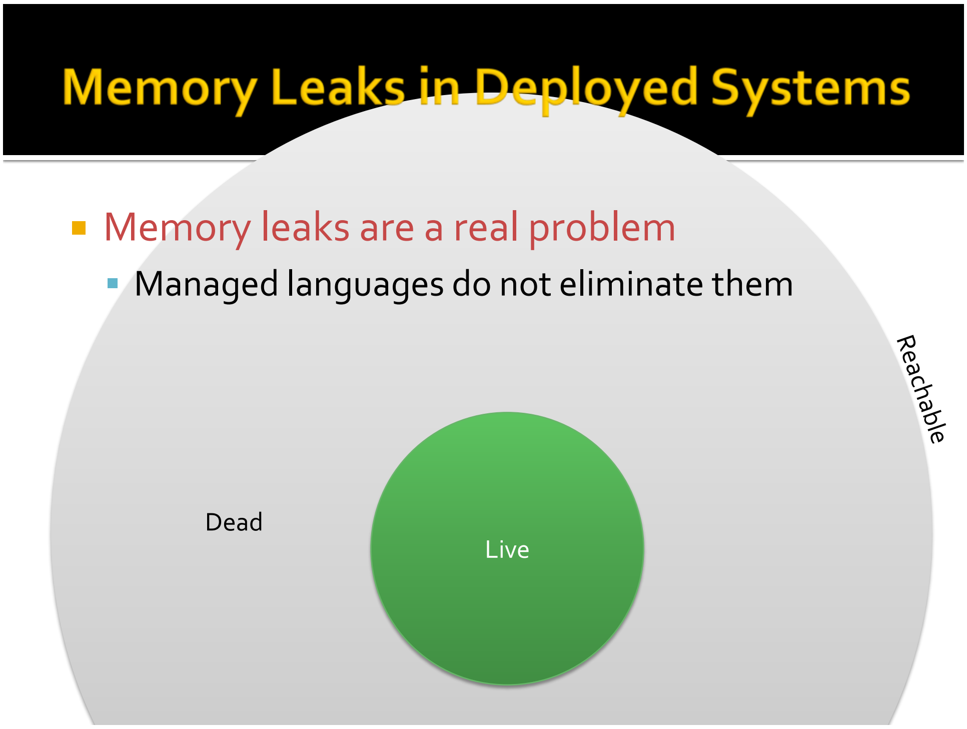
- Memory leaks are a real problem
  - Managed languages do not eliminate them



Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them





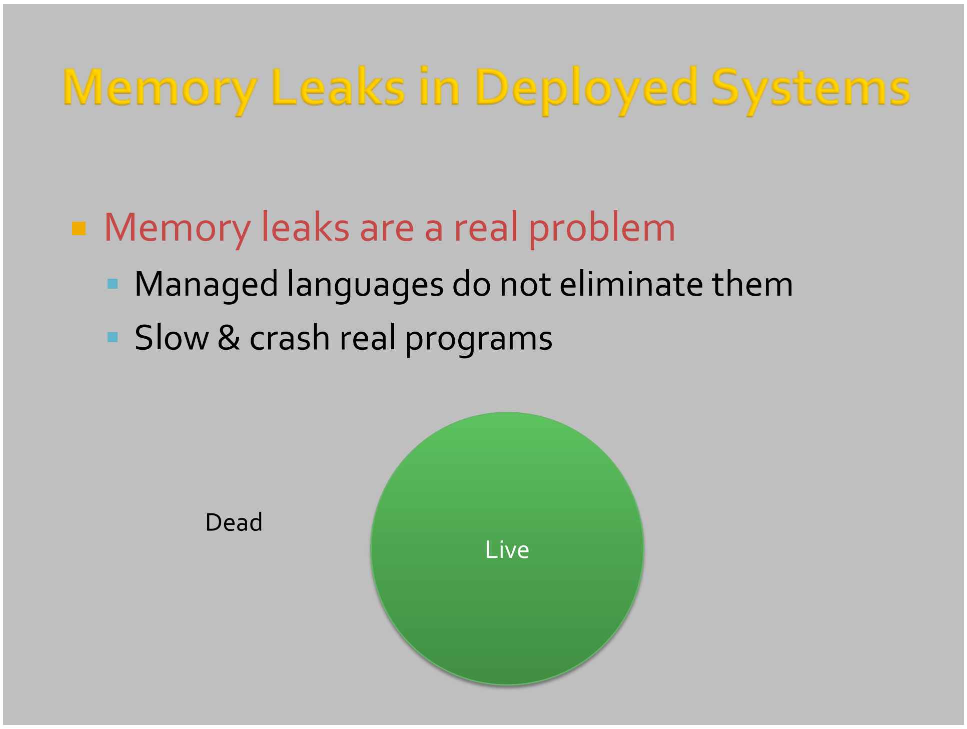
Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them

Reachable

Dead

Live



Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them
  - Slow & crash real programs

Dead

Live

Memory Leaks in Deployed Systems

- Memory leaks are a real problem
  - Managed languages do not eliminate them
  - Slow & crash real programs
  - Unacceptable for some applications

Memory Leaks in Deployed Systems

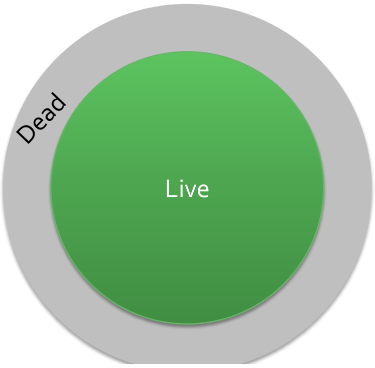
- Memory leaks are a real problem
  - Managed languages do not eliminate them
  - Slow & crash real programs
  - Unacceptable for some applications
- Fixing leaks is hard
  - Leaks take time to materialize
  - Failure far from cause

Outline

- Why tolerate leaks
- Why leaks are so bad
- How leak pruning works
- How leak pruning predicts leaked objects
- Evaluation

Reclaiming Memory while Preserving Semantics

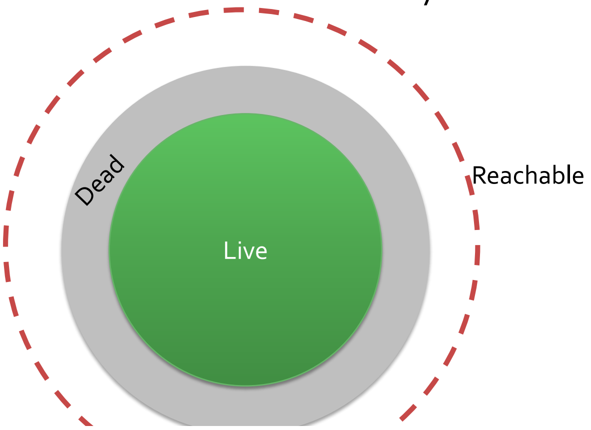
- Garbage collection based on liveness



Reachable

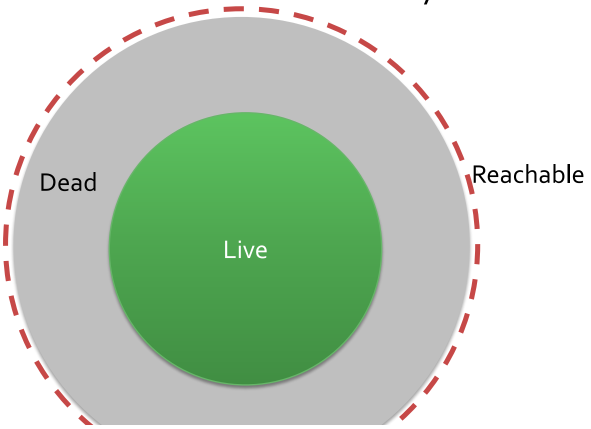
Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Take action when memory exhausted



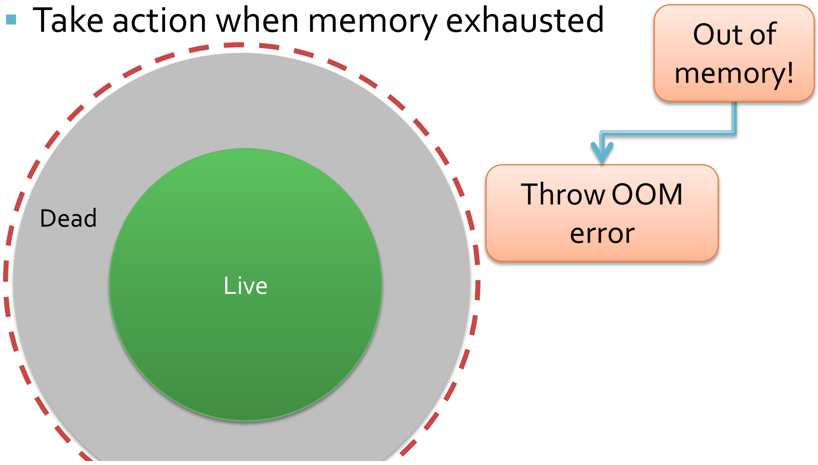
Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Take action when memory exhausted



Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Take action when memory exhausted



Dead

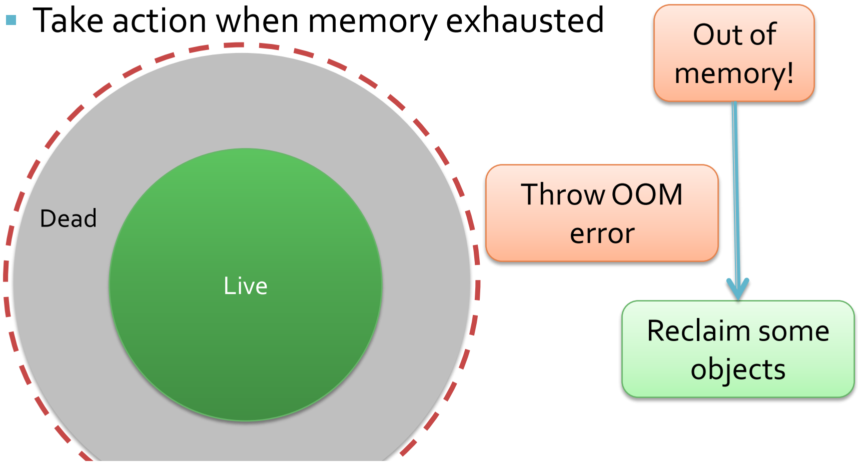
Live

Out of memory!

Throw OOM error

Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Take action when memory exhausted



Dead

Live

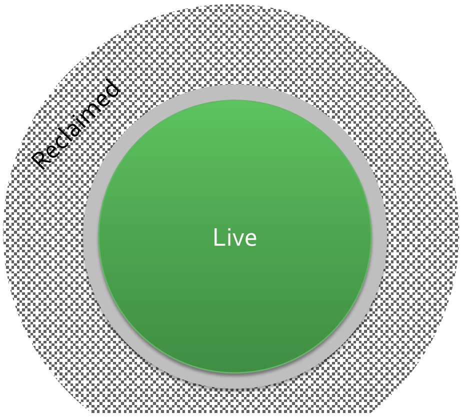
Out of memory!

Throw OOM error

Reclaim some objects

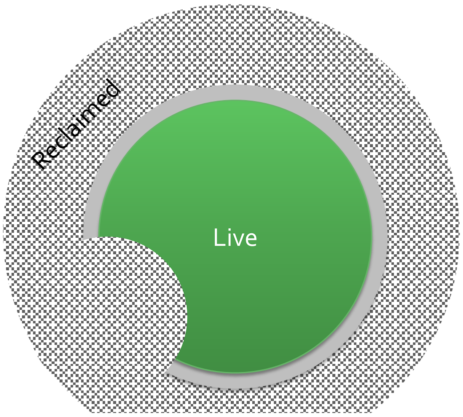
Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Reclaim predicted dead objects



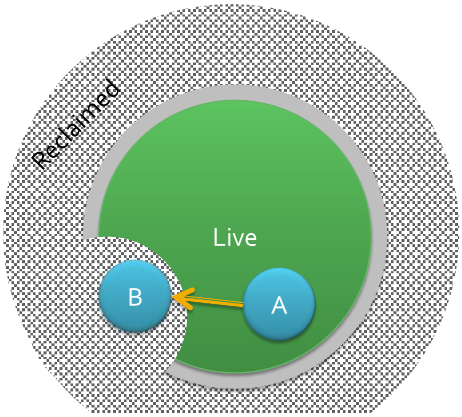
Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Reclaim predicted dead objects



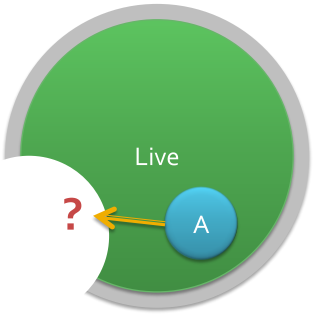
Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Reclaim predicted dead objects



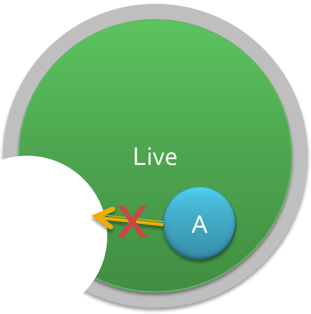
Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Reclaim predicted dead objects



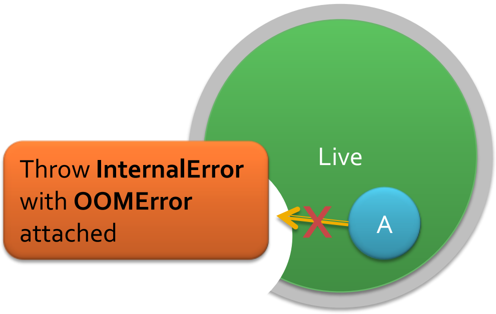
Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Reclaim predicted dead objects
  - *Poison* references to reclaimed objects



Reclaiming Memory while Preserving Semantics

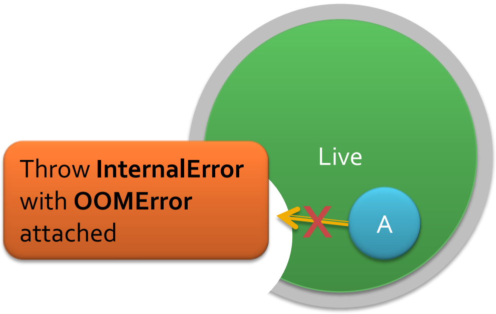
- Garbage collection based on liveness
  - Reclaim predicted dead objects
  - *Poison* references to reclaimed objects



Throw **InternalError** with **OOMError** attached

Reclaiming Memory while Preserving Semantics

- Garbage collection based on liveness
  - Reclaim predicted dead objects
  - *Poison* references to reclaimed objects



Throw **InternalError** with **OOMError** attached

Worst case: defers fatal errors

Best case: keeps leaky programs running much longer or indefinitely

Outline

- Why tolerate leaks
- Why leaks are so bad
- How leak pruning works
- How leak pruning predicts leaked objects
- Evaluation

Predicting Dead Objects

- Predicting the future
  - Leaked objects → not used again
  - Highly *stale* objects → *likely* leaked

  [Chilimbi & Hauswirth ’04] [Qin et al. ’05] [Bond & McKinley ’06]

Predicting Dead Objects

- Predicting the future
  - Leaked objects → not used again
  - Highly *stale* objects → *likely* leaked
- Alternative: offload to disk

  [**Melt**, Bond & McKinley ’08]
  [**LeakSurvivor**, Tang et al. ’08]
  [**Panacea**, Goldstein et al. ’07, Breitgand et al. ’07]
  - Tolerates mispredictions
  - Eventually exhausts disk (if disk at all)

Predicting Dead Objects

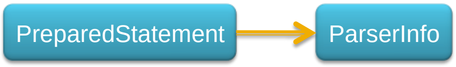
- Predicting the future
  - Leaked objects → not used again
  - Highly *stale* objects → *likely* leaked
- Alternati
  [**Melt**, Bo
  [**LeakSur**
  [**Panacea**
  '07]
  - Tolera
  - Eventually exhausts disk (if disk at all)

Need high precision

One misprediction: program terminates

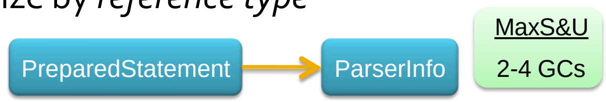
Predicting Dead Data Structures

- Identify references to prune
  - Roots of leaked data structures
  - Categorize by *reference type*



Predicting Dead Data Structures

- Identify references to prune
  - Roots of leaked data structures
  - Categorize by *reference type*
- Criteria

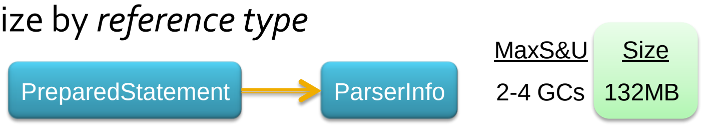


MaxS&U
2-4 GCs

  - Highly stale references
    - More stale than *most stale instance accessed previously*

Predicting Dead Data Structures

- Identify references to prune
  - Roots of leaked data structures
  - Categorize by *reference type*
- Criteria

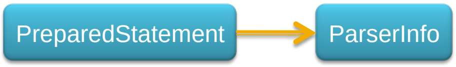


MaxS&U	Size
2-4 GCs	132MB

  - Highly stale references
    - More stale than *most stale instance accessed previously*
  - Reference type keeping most bytes reachable

Predicting Dead Data Structures

- Identify references to prune
  - Roots of leaked data structures
  - Categorize by *reference type*



MaxS&U	Size
2-4 GCs	132MB

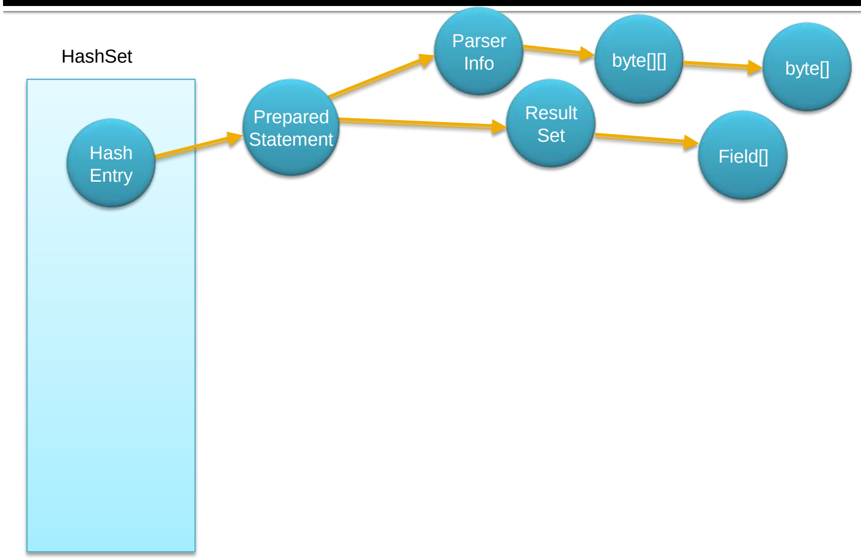
- Criteria
  - Highly stale references
    - More stale than *most stale instance accessed previously*
  - Reference type keeping most bytes reachable

Piggyback on GC: two-phase transitive closure

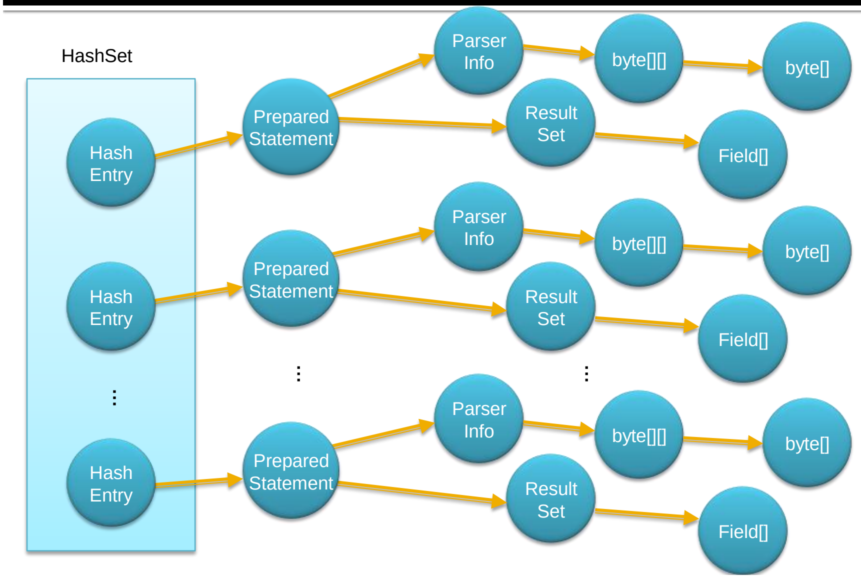
MySQL Leak

- Based on colleague's JDBC application
- Leak: SQL statements remain in set

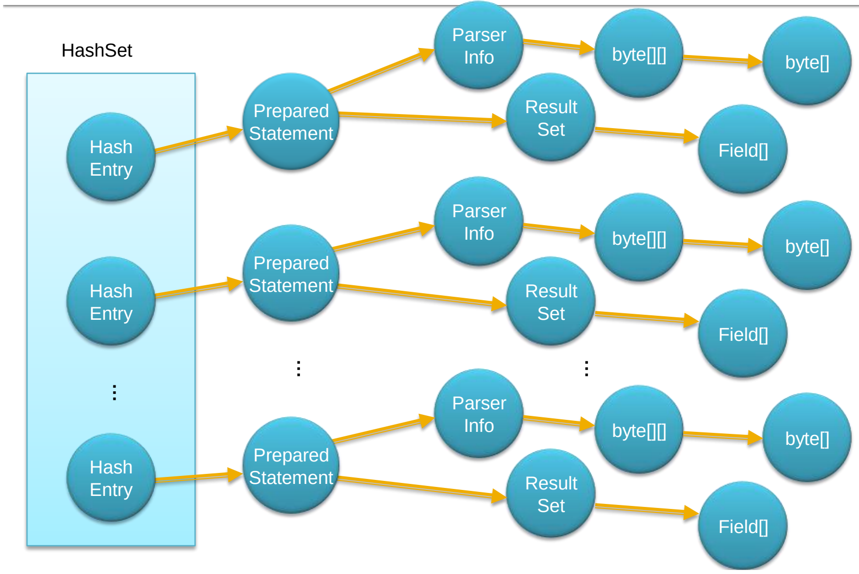
MySQL Leak



MySQL Leak

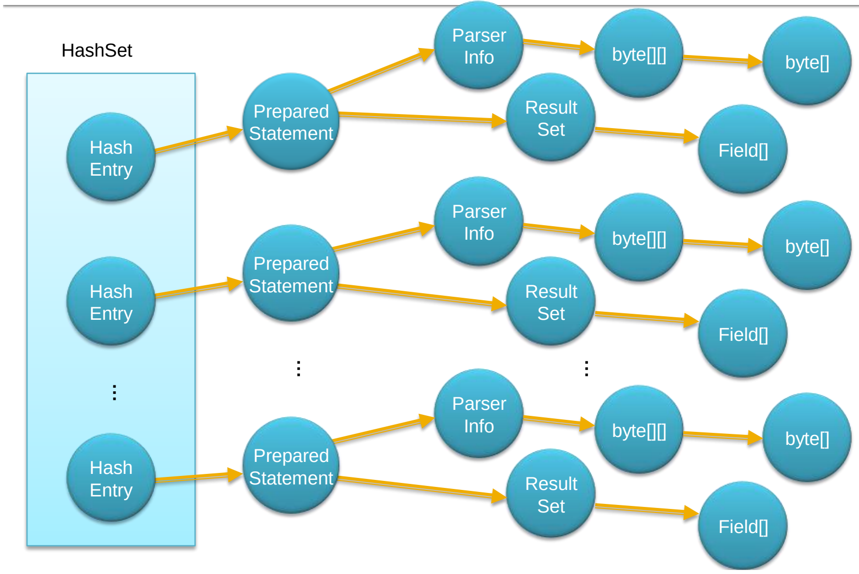


Transitive Closure: Phase One



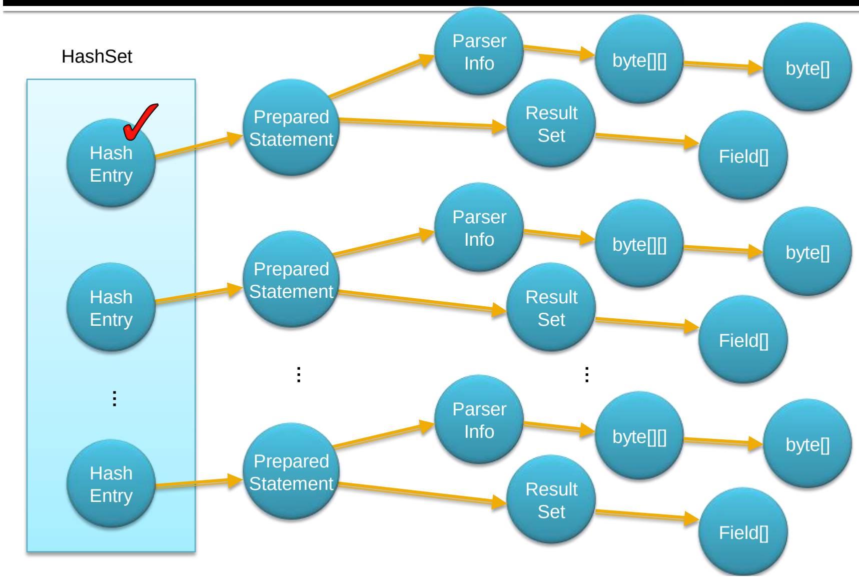
Transitive Closure: Phase One

		MaxS&U
HashEntry →	PreparedStatement	16-32
PreparedStatement →	ParserInfo	0-1
PreparedStatement →	ResultSet	0-1



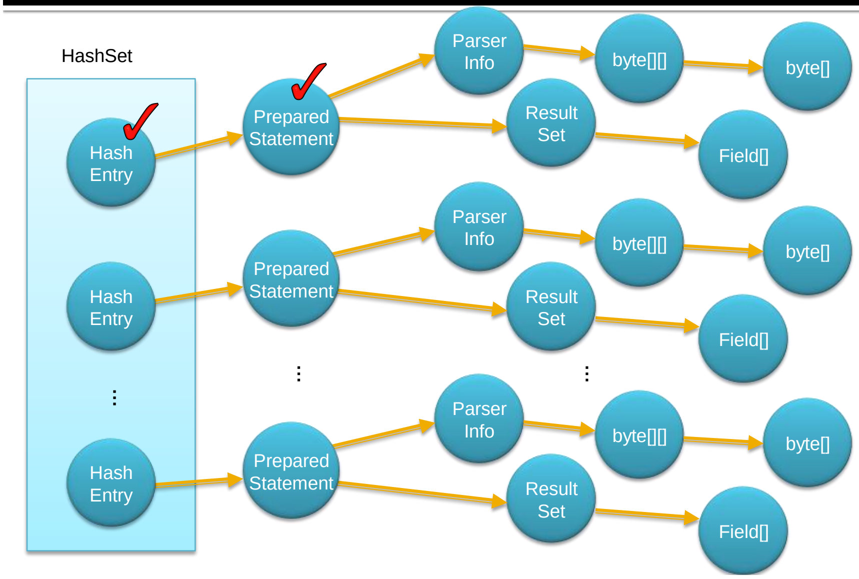
Transitive Closure: Phase One

	MaxS&U
HashEntry → PreparedStatement	16-32
PreparedStatement → ParserInfo	0-1
PreparedStatement → ResultSet	0-1



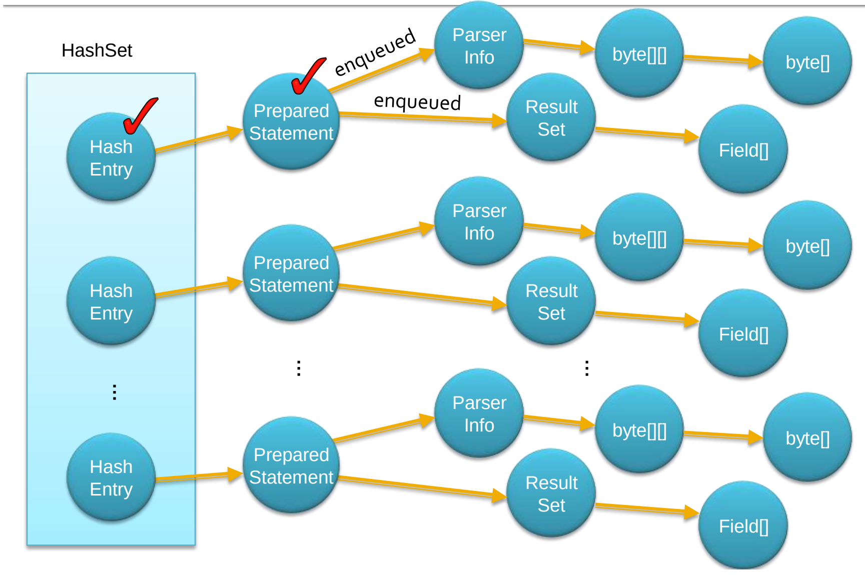
Transitive Closure: Phase One

	MaxS&U
HashEntry → PreparedStatement	16-32
PreparedStatement → ParserInfo	0-1
PreparedStatement → ResultSet	0-1



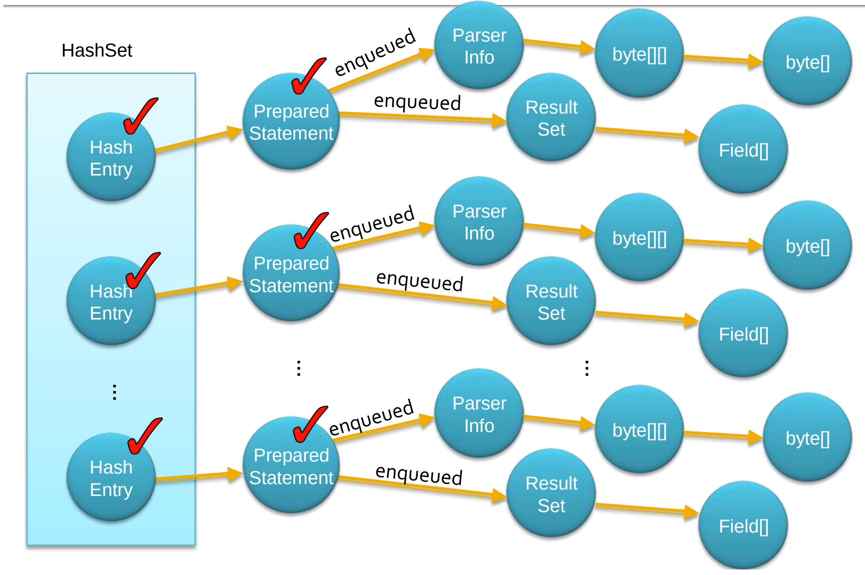
Transitive Closure: Phase One

	MaxS&U
HashEntry → PreparedStatement	16-32
PreparedStatement → ParserInfo	0-1
PreparedStatement → ResultSet	0-1



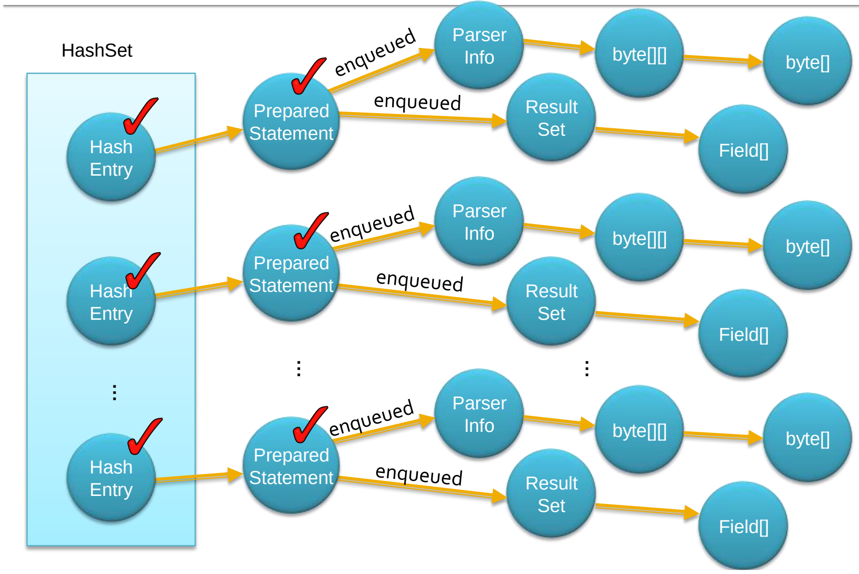
Transitive Closure: Phase One

	MaxS&U
HashEntry → PreparedStatement	16-32
PreparedStatement → ParserInfo	0-1
PreparedStatement → ResultSet	0-1



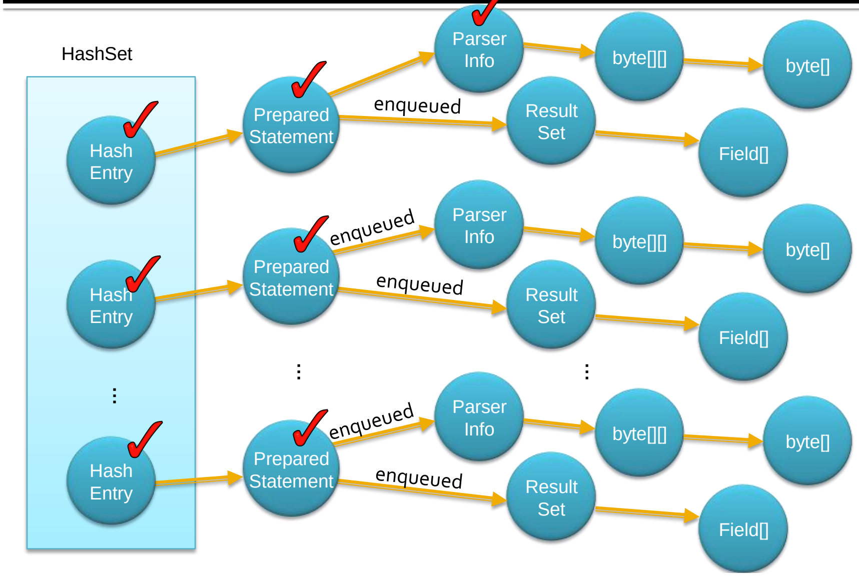
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	0
PreparedStatement → ResultSet	0-1	0



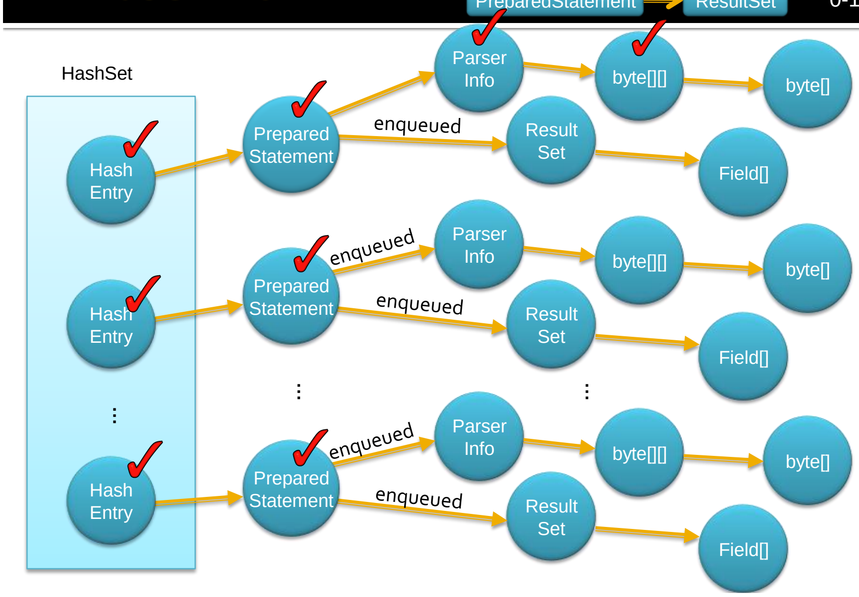
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	20
PreparedStatement → ResultSet	0-1	0



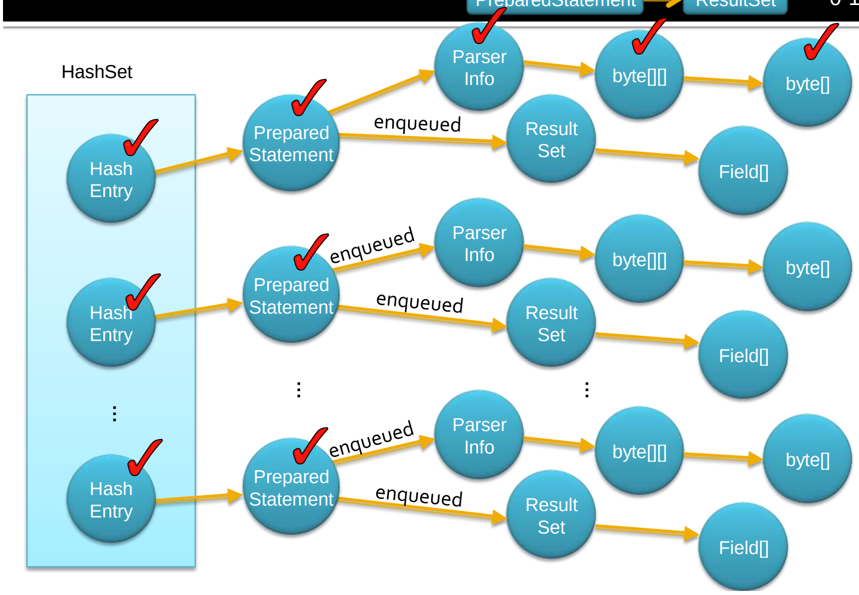
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	60
PreparedStatement → ResultSet	0-1	0



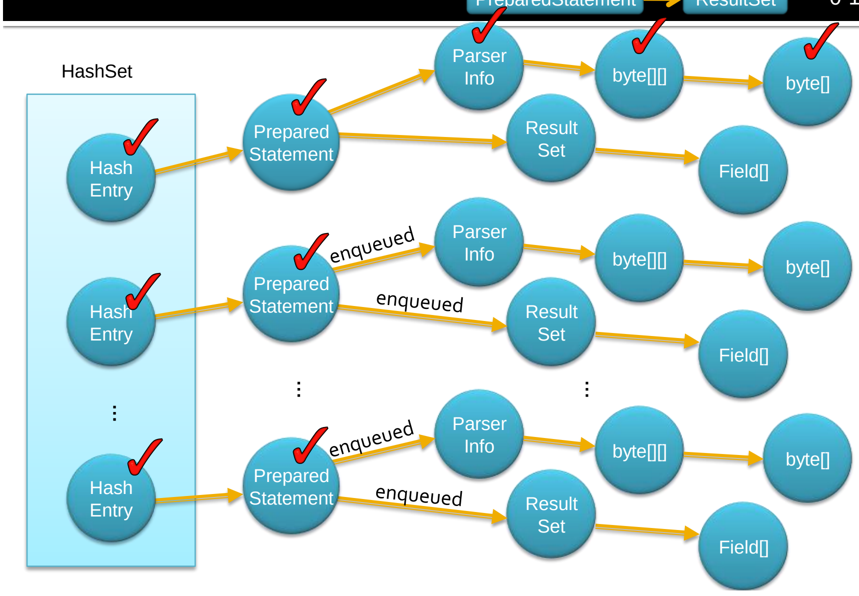
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	572
PreparedStatement → ResultSet	0-1	0



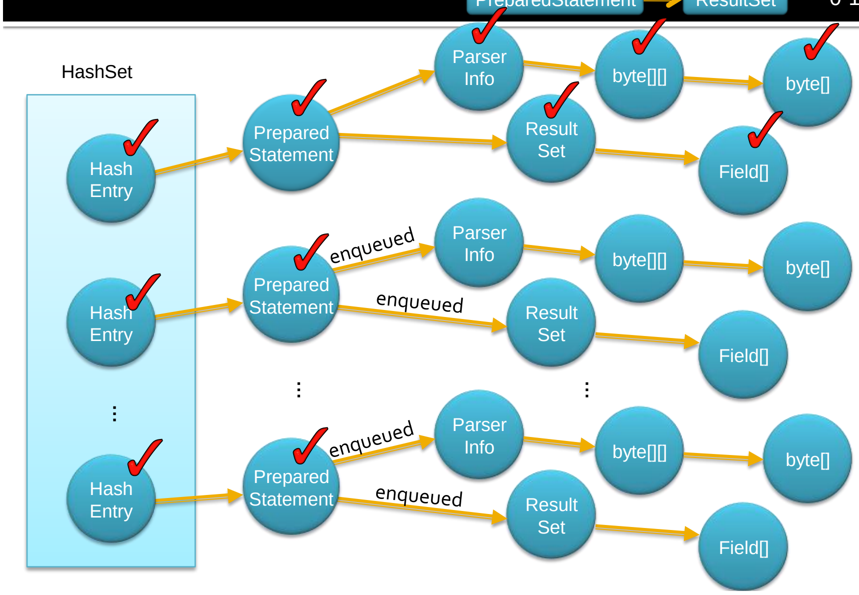
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	572
PreparedStatement → ResultSet	0-1	0



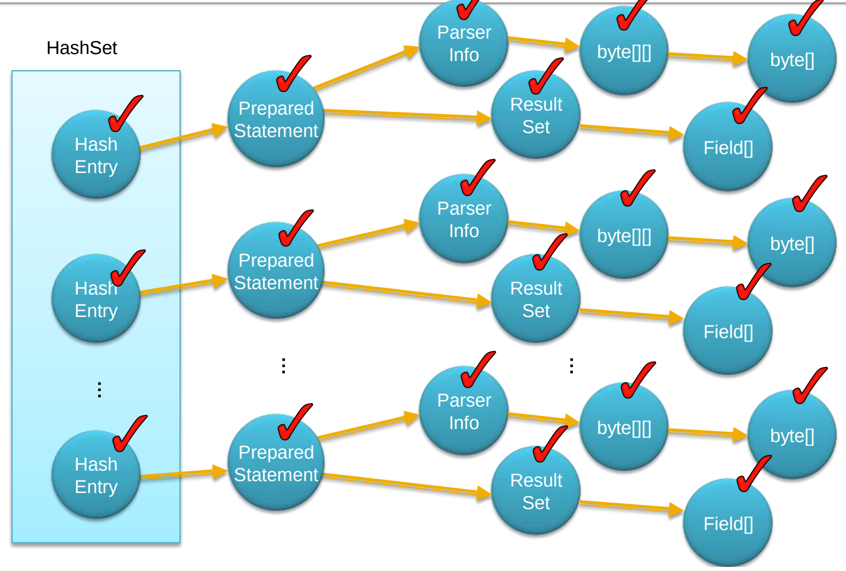
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	572
PreparedStatement → ResultSet	0-1	244



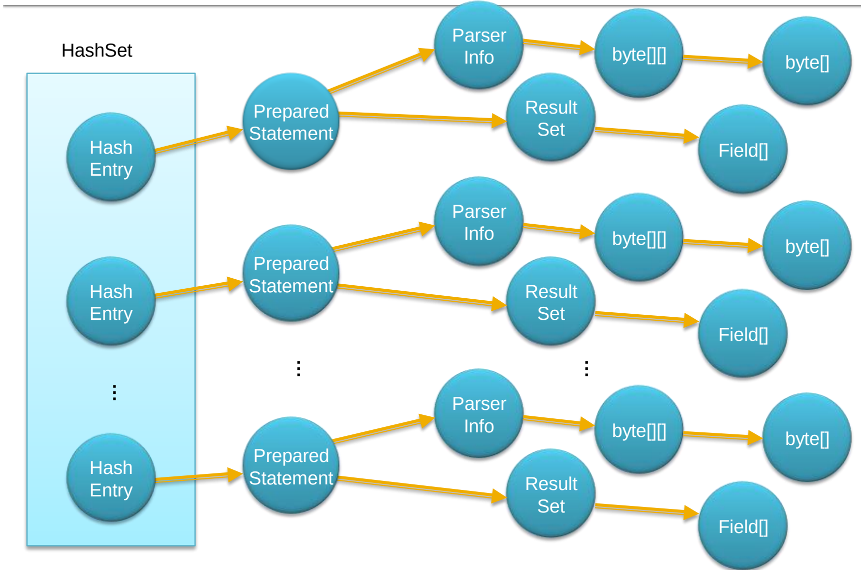
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	420M
PreparedStatement → ResultSet	0-1	180M



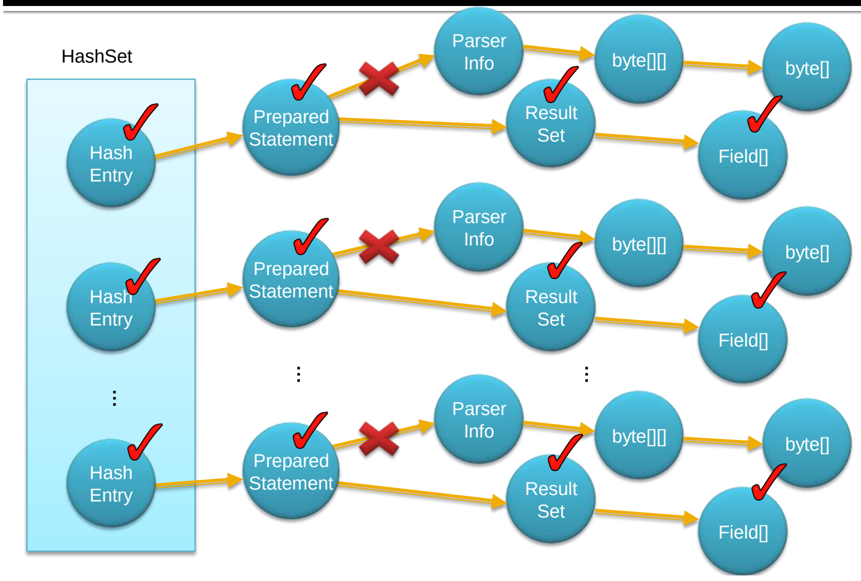
Select Best Reference Type

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	420M
PreparedStatement → ResultSet	0-1	180M



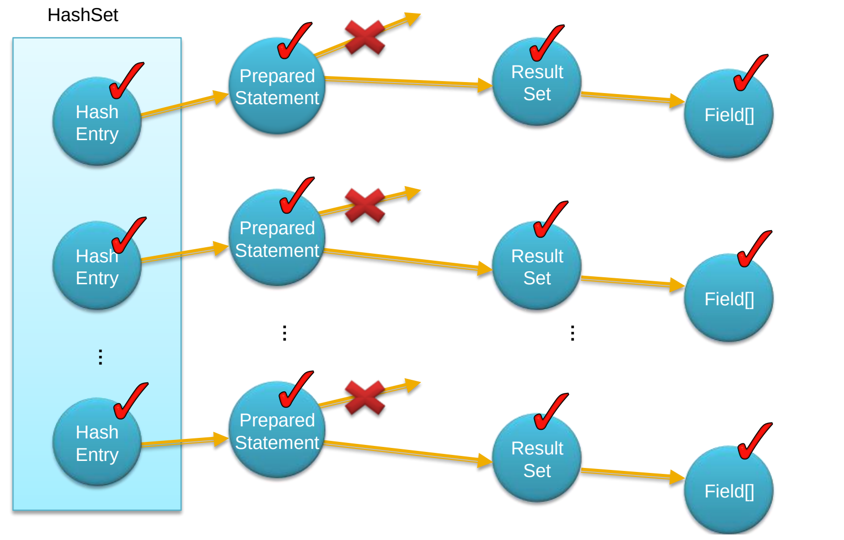
Prune Selected Reference Type

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	420M
PreparedStatement → ResultSet	0-1	180M



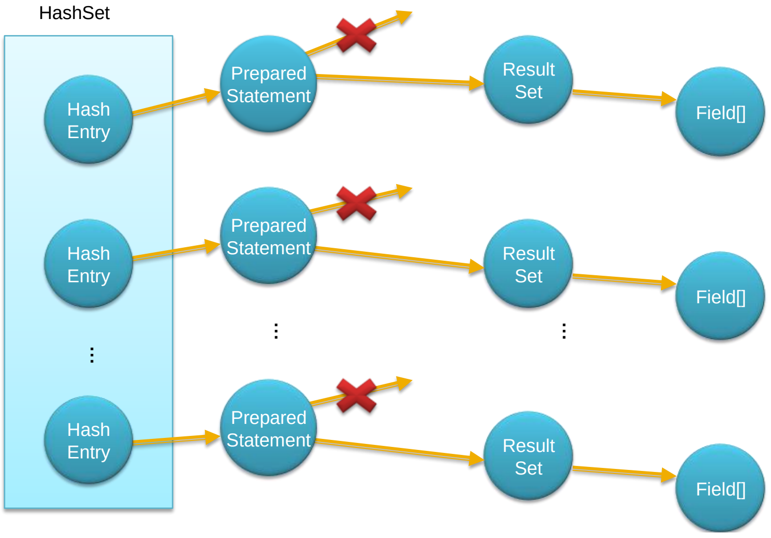
Prune Selected Reference Type

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	420M
PreparedStatement → ResultSet	0-1	180M



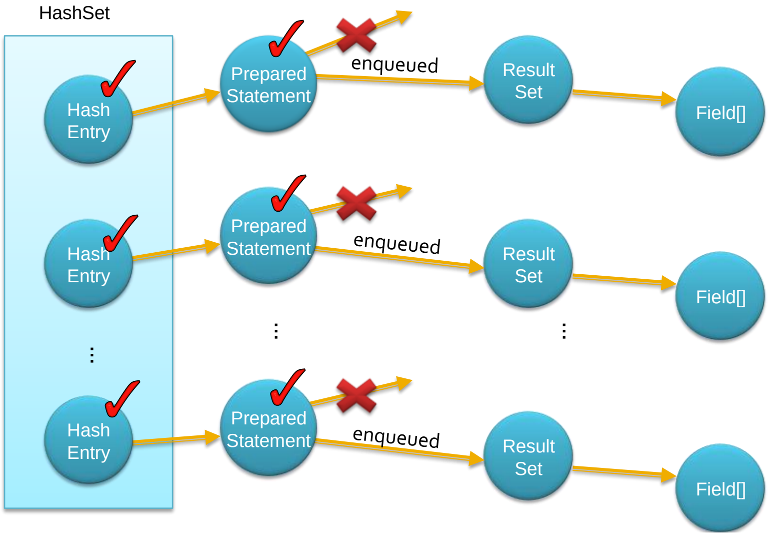
Leak Partially Eliminated

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	0
PreparedStatement → ResultSet	0-1	0



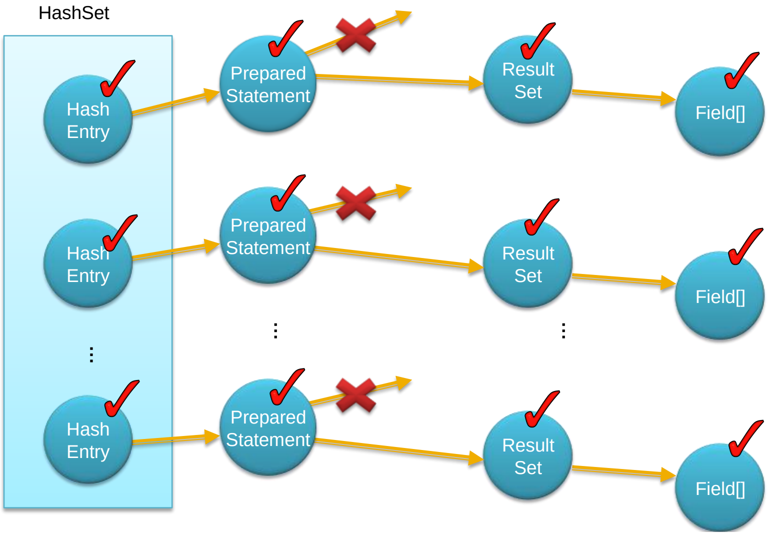
Transitive Closure: Phase One

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	0
PreparedStatement → ResultSet	0-1	0



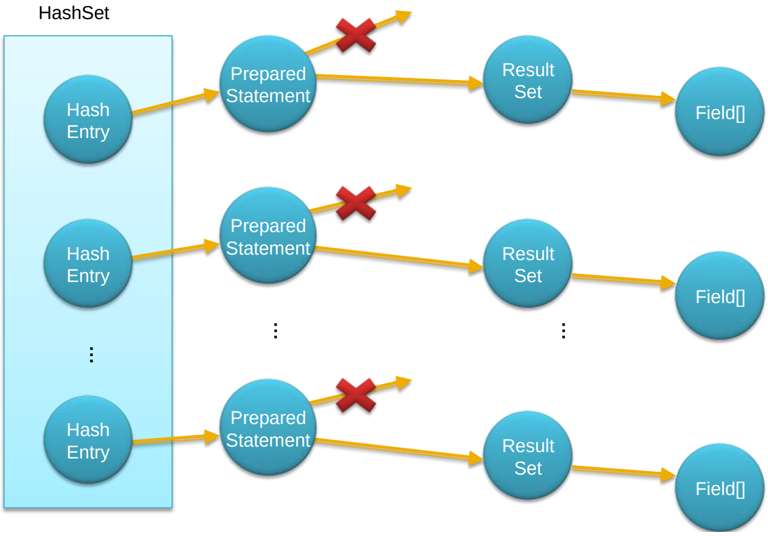
Transitive Closure: Phase Two

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	50M
PreparedStatement → ResultSet	0-1	220M



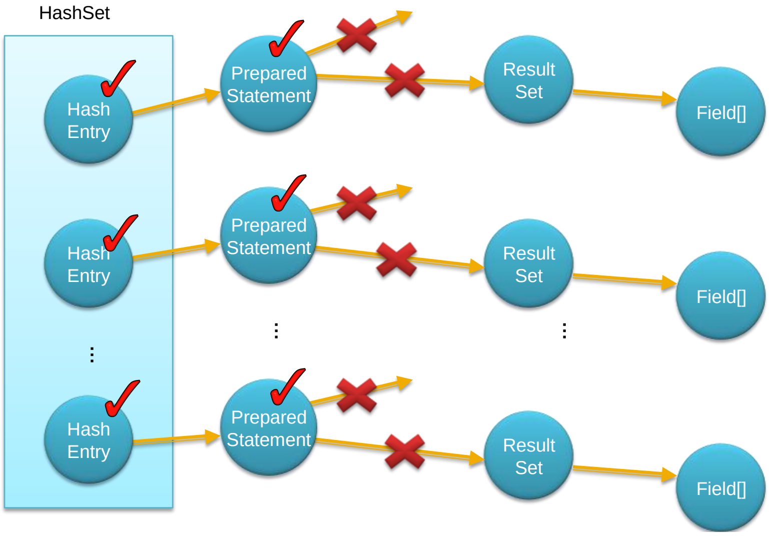
Select Best Reference Type

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	50M
PreparedStatement → ResultSet	0-1	220M



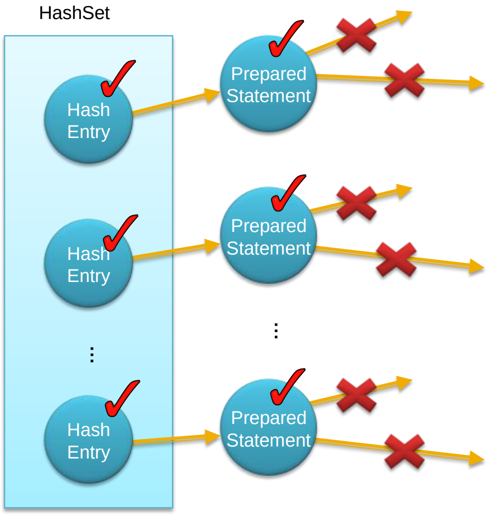
Prune Selected Reference Type

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	50M
PreparedStatement → ResultSet	0-1	220M

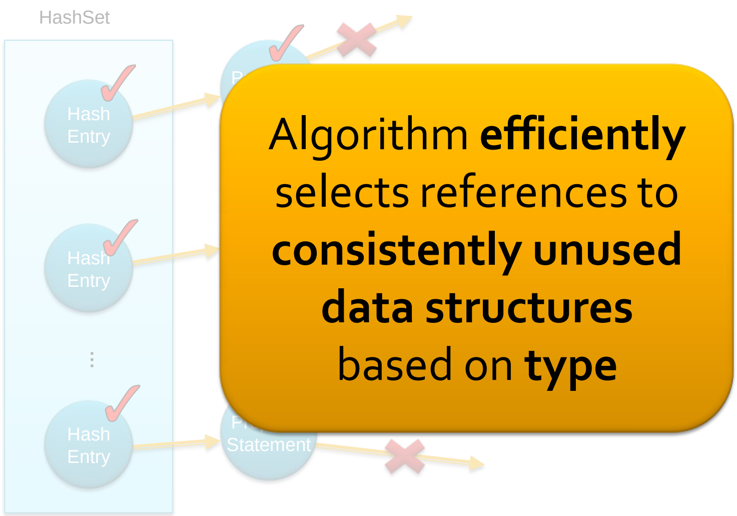


Prune Selected Reference Type

	MaxS&U	Bytes
HashEntry → PreparedStatement	16-32	0
PreparedStatement → ParserInfo	0-1	50M
PreparedStatement → ResultSet	0-1	220M



Prediction Summary



Algorithm **efficiently** selects references to **consistently unused data structures** based on **type**

Outline

- Why tolerate leaks
- Why leaks are so bad
- How leak pruning works
- How leak pruning predicts leaked objects
- Evaluation

Implementation

- Leak pruning added to Jikes RVM 2.9.2

  http://www.jikesrvm.org/Research+Archive

Implementation

- Leak pruning added to Jikes RVM 2.9.2

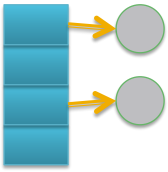
  http://www.jikesrvm.org/Research+Archive

- Performance stress test
  - Non-leaking programs: DaCapo & SPEC benchmarks
  - Low overhead
    - 3% (Core 2) or 5% (Pentium 4)
    - Primarily read barriers

Tolerating Leaks

Leak	Leak pruning's effect
Eclipse "Diff"	Tolerates until 24-hr limit (>200X longer)
ListLeak	Tolerates until 24-hr limit (>25,000X longer)
SwapLeak	Tolerates until 24-hr limit (>2,200X longer)
Eclipse "Copy-Paste"	Most dead but some live (81X longer)
MySQL	Most dead but some live (35X longer)
JbbMod	All dead but pruning misses some (21X longer)
SPECjbb2000	Heap growth is mostly live (4.7X longer)
Mckoi Database	Thread leak: extra support needed (1.6X longer)
DualLeak	Heap growth is live (No help)

Tolerating Leaks

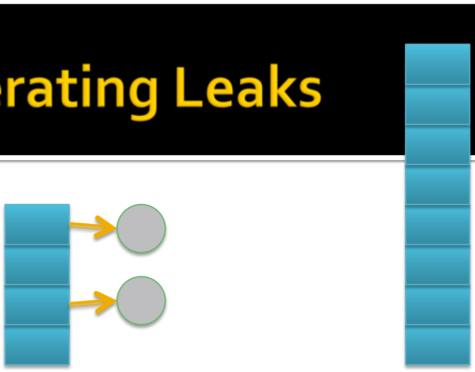


Eclipse "Copy-Paste"	Most dead but some live (81X longer)
MySQL	Most dead but some live (35X longer)

SPECjbb2000	Heap growth is mostly live (4.7X longer)

DualLeak	Heap growth is live (No help)

Tolerating Leaks

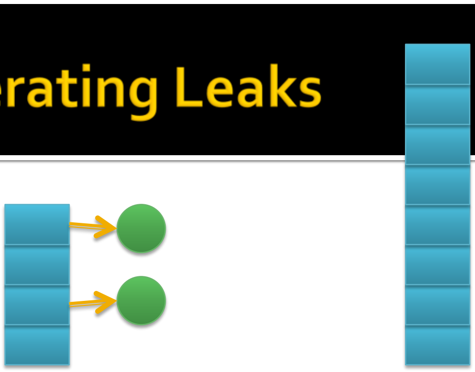


Eclipse “Copy-Paste”	Most dead but some live (81X longer)
MySQL	Most dead but some live (35X longer)

SPECjbb2000	Heap growth is mostly live (4.7X longer)

DualLeak	Heap growth is live (No help)

Tolerating Leaks

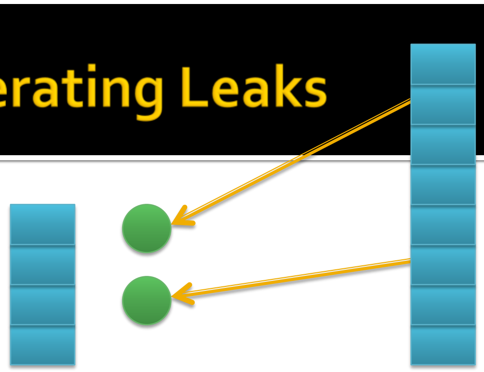


Eclipse “Copy-Paste”	Most dead but some live (81X longer)
MySQL	Most dead but some live (35X longer)

SPECjbb2000	Heap growth is mostly live (4.7X longer)

DualLeak	Heap growth is live (No help)

Tolerating Leaks

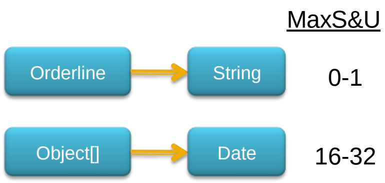


Eclipse “Copy-Paste”	Most dead but some live (81X longer)
MySQL	Most dead but some live (35X longer)

SPECjbb2000	Heap growth is mostly live (4.7X longer)

DualLeak	Heap growth is live (No help)

Tolerating Leaks



MaxS&U

Orderline → String 0-1

Object[] → Date 16-32

JbbMod	All dead but pruning misses some (21X longer)

Tolerating Leaks

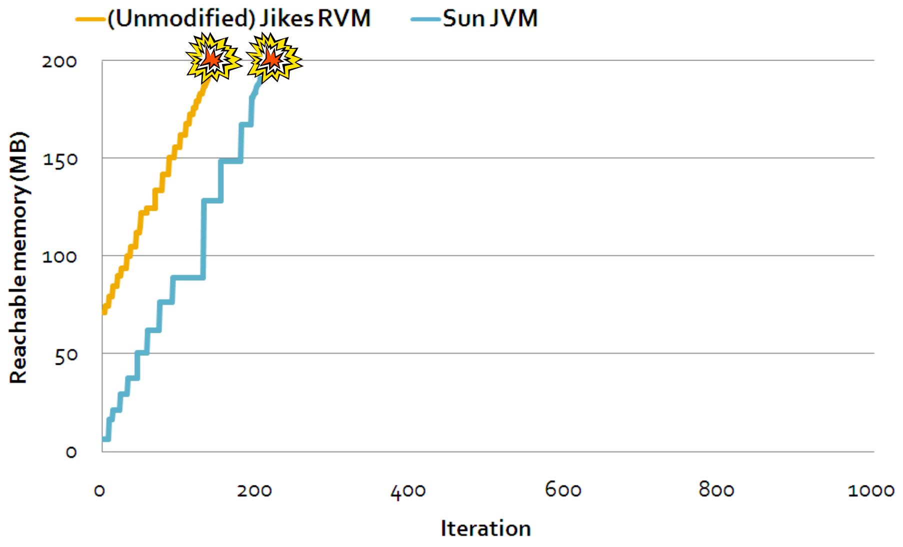
Leak	Leak pruning’s effect
Eclipse “Diff”	Tolerates until 24-hr limit (>200X longer)
ListLeak	Tolerates until 24-hr limit (>25,000X longer)
SwapLeak	Tolerates until 24-hr limit (>2,200X longer)
Eclipse “Copy-Paste”	Most dead but some live (81X longer)
MySQL	Most dead but some live (35X longer)
JbbMod	All dead but pruning misses some (21X longer)
SPECjbb2000	Heap growth is mostly live (4.7X longer)
Mckoi Database	Thread leak: extra support needed (1.6X longer)
DualLeak	Heap growth is live (No help)

Tolerating Leaks

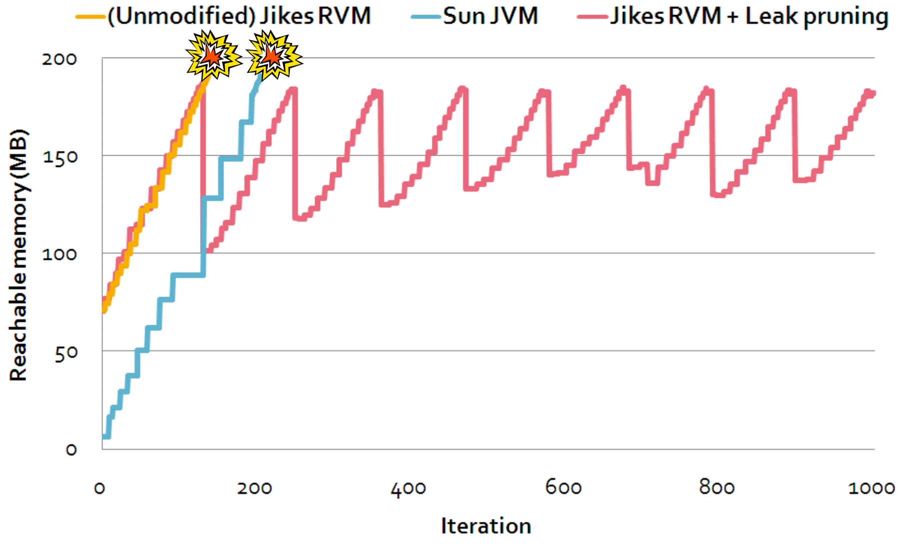
Eclipse “Diff”	Tolerates until 24-hr limit (>200X longer)

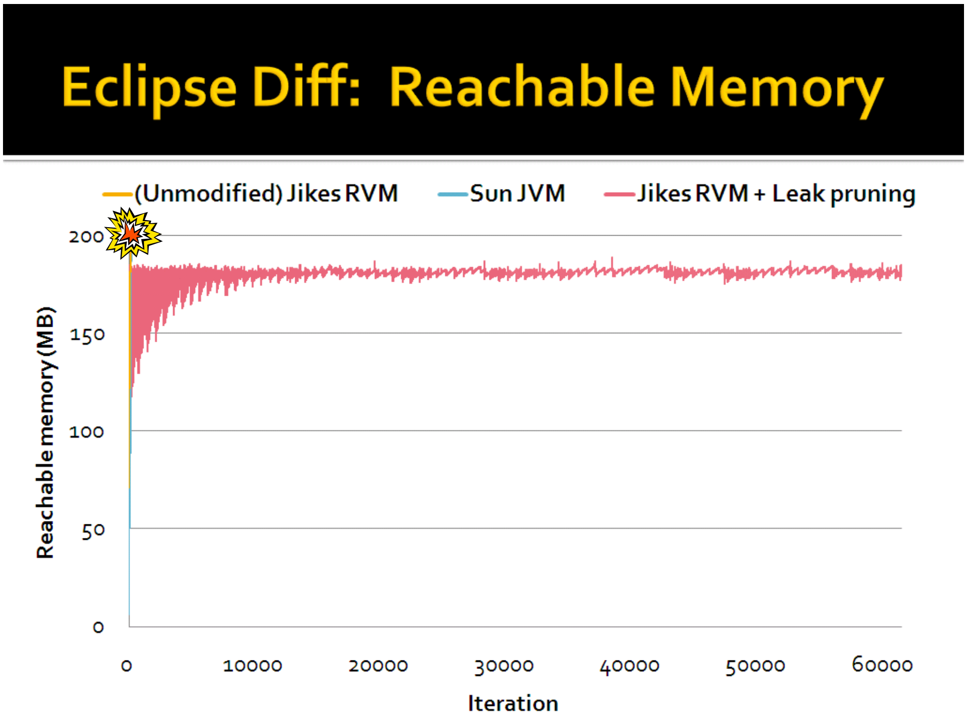
- Reported on Eclipse Bugzilla
- Leak: recursive difference
- Automated with Eclipse Plugin

Eclipse Diff: Reachable Memory

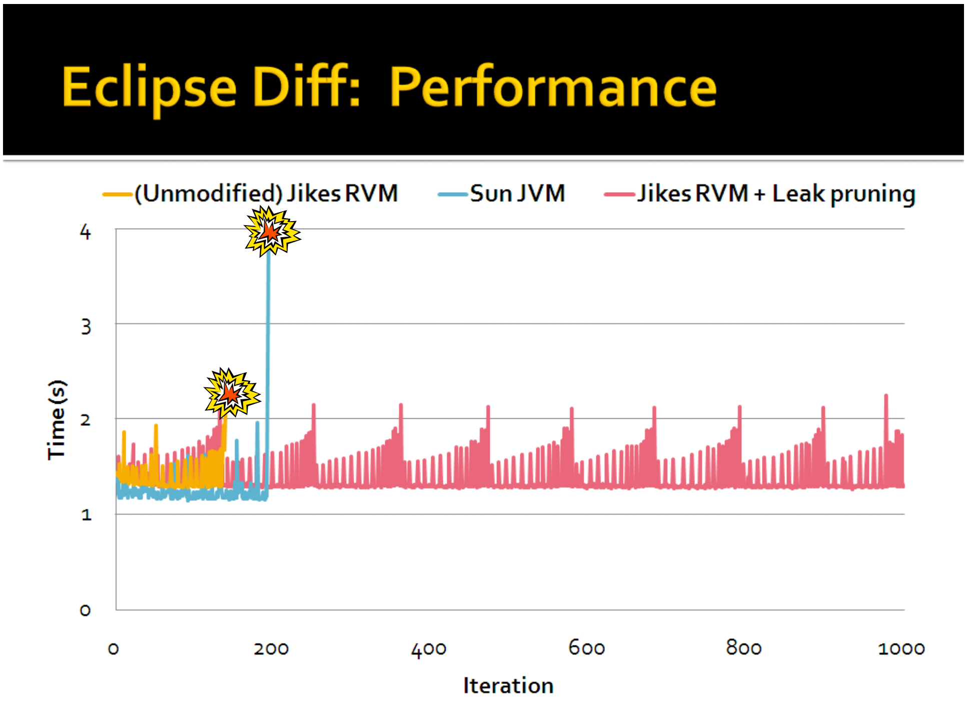


Eclipse Diff: Reachable Memory

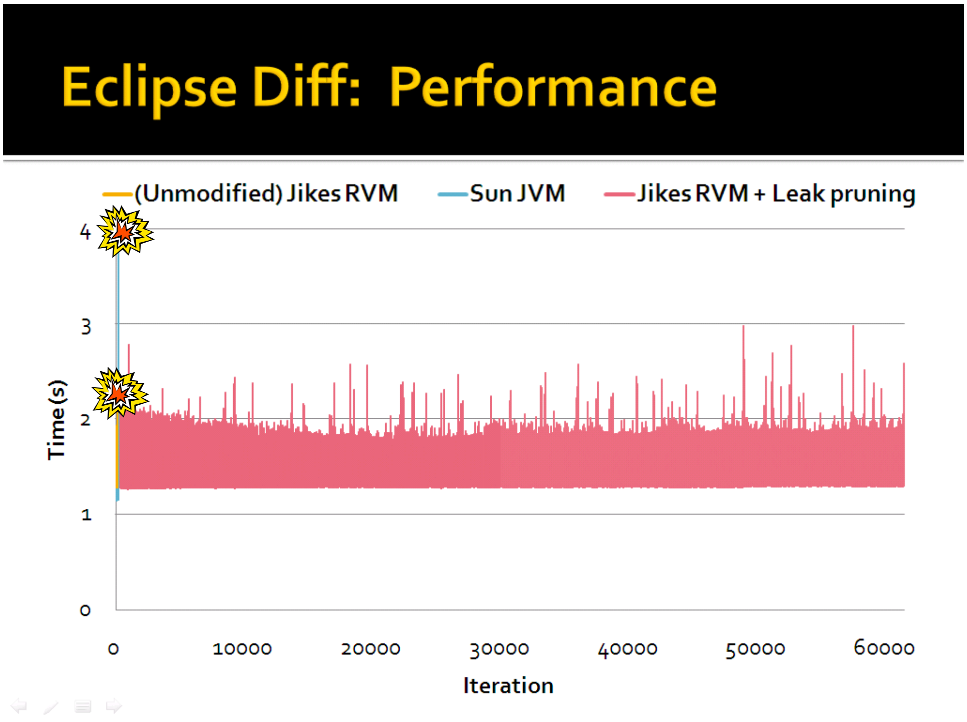




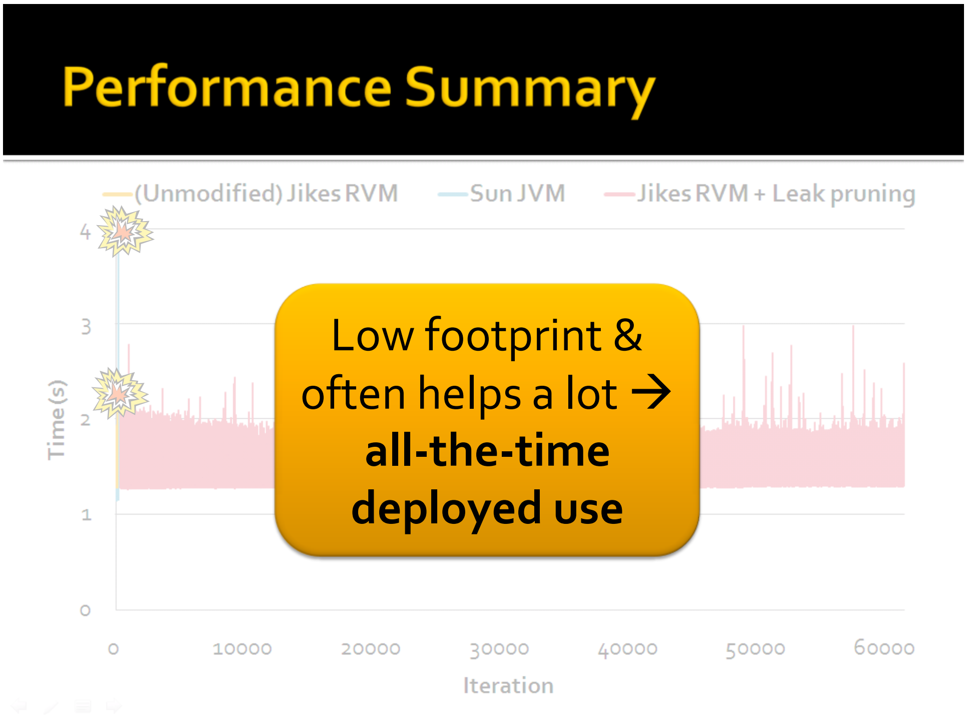
Eclipse Diff: Reachable Memory



Eclipse Diff: Performance



Eclipse Diff: Performance



Performance Summary

Low footprint & often helps a lot → **all-the-time deployed use**

Related Work

- Memory corruption & concurrency bugs
  - Perturb layout & scheduling [**Rx**, Qin et al. '05] [**DieHard**, Berger & Zorn '06] [**Atom-Aid**, Lucia et al. '08] [**Grace**, Berger et al. '08]
- Fatal errors
  - Ignore errors [**Failure-oblivious computing**, Rinard et al. '04] [**Append**, Dobolyi & Weimer '08]

Related Work

- Memory corruption & concurrency bugs
  - Perturb layout & scheduling [**Rx**, Qin et al. '05] [**DieHard**, Berger & Zorn '06] [**Atom-Aid**, Lucia et al. '08] [**Grace**, Berger et al. '08]
- Fatal errors
  - Ignore errors [**Failure-oblivious computing**, Rinard et al. '04] [**Append**, Dobolyi & Weimer '08]

- Memory leaks in unmanaged languages
  - Leak-friendly layout [**Plug**, Novark et al. '08]
  - Bound allocation sites [**Cyclic alloc**, Nguyen & Rinard '07]

Tolerating Memory Leaks in Managed Languages

- Finding leaks before deployment is hard
  - Deployed systems need immediate help
- Leak pruning: GC based on liveness
  - High precision & low overhead
  - Bounded resources
  - Preserves semantics

Tolerating Memory Leaks in Managed Languages

- Finding leaks before deployment is hard
  - Deployed systems need immediate help
- Leak pruning: GC based on liveness
  - High precision & low overhead
  - Bounded resources
  - Preserves semantics

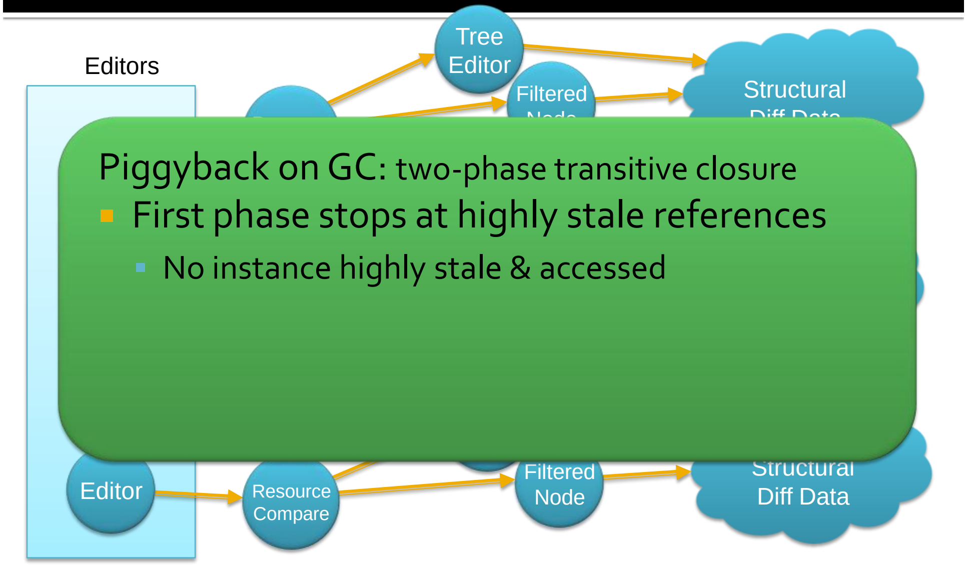
Thank you!

Backup

Eclipse "Diff" Leak

- Reported on Eclipse Bugzilla
- Recursive difference leaks memory

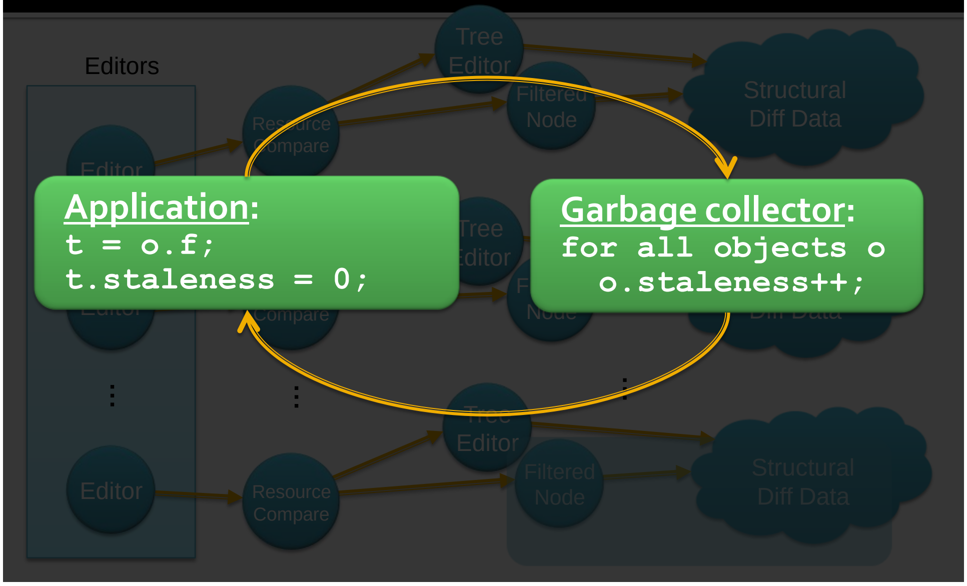
Eclipse "Diff" Leak



Piggyback on GC: two-phase transitive closure

- First phase stops at highly stale references
  - No instance highly stale & accessed

Fine-Grained Usage Tracking

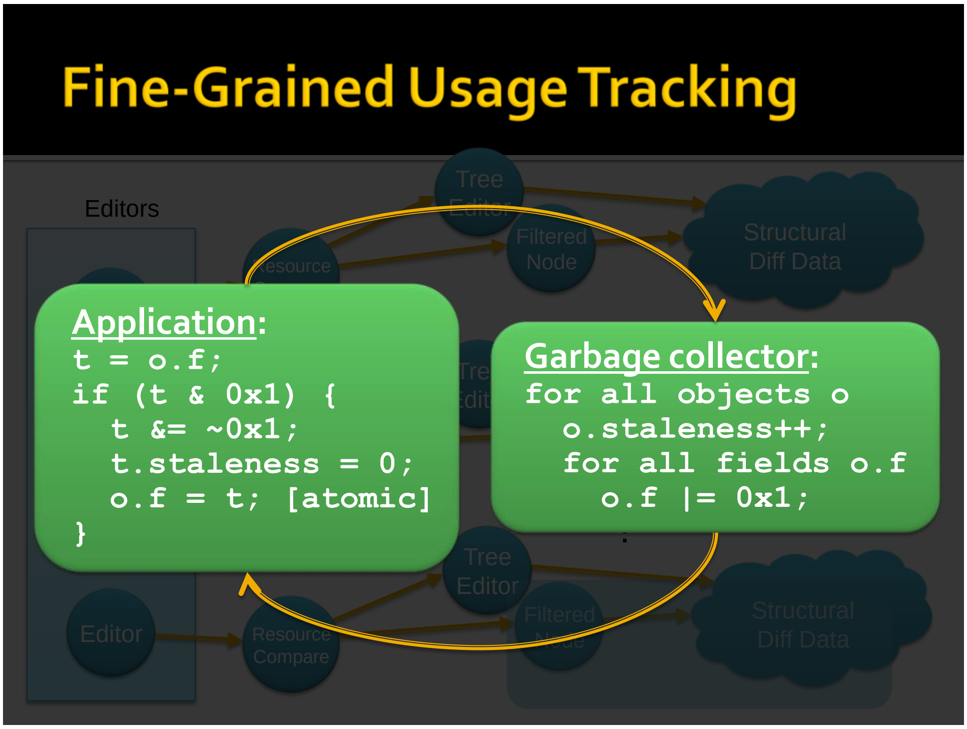


Application:

```
t = o.f;
t.staleness = 0;
```

Garbage collector:

```
for all objects o
  o.staleness++;
```



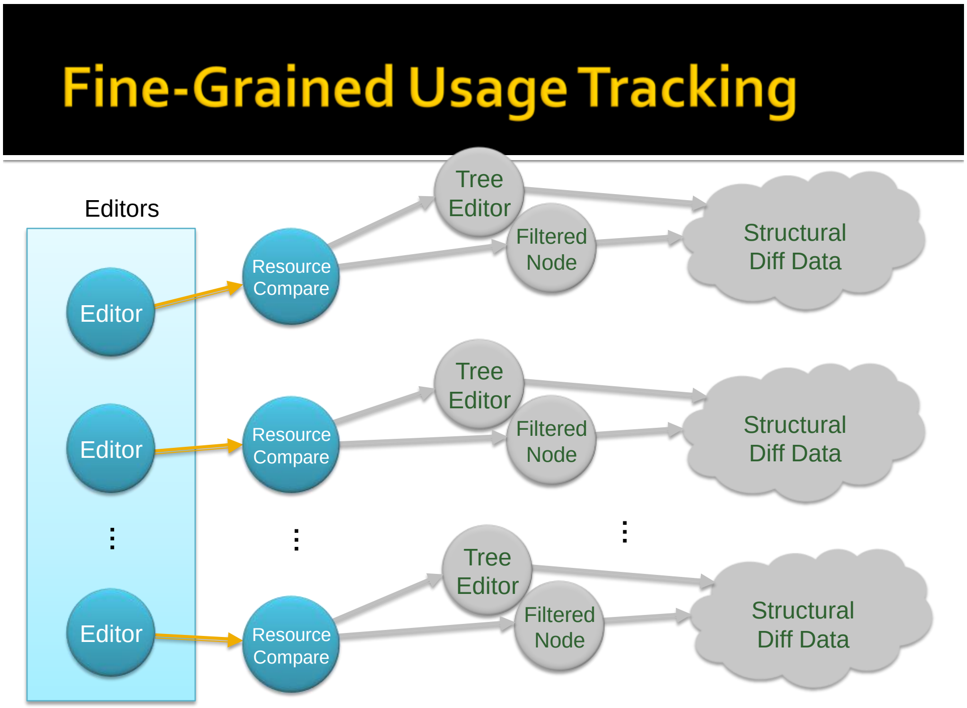
Fine-Grained Usage Tracking

Application:

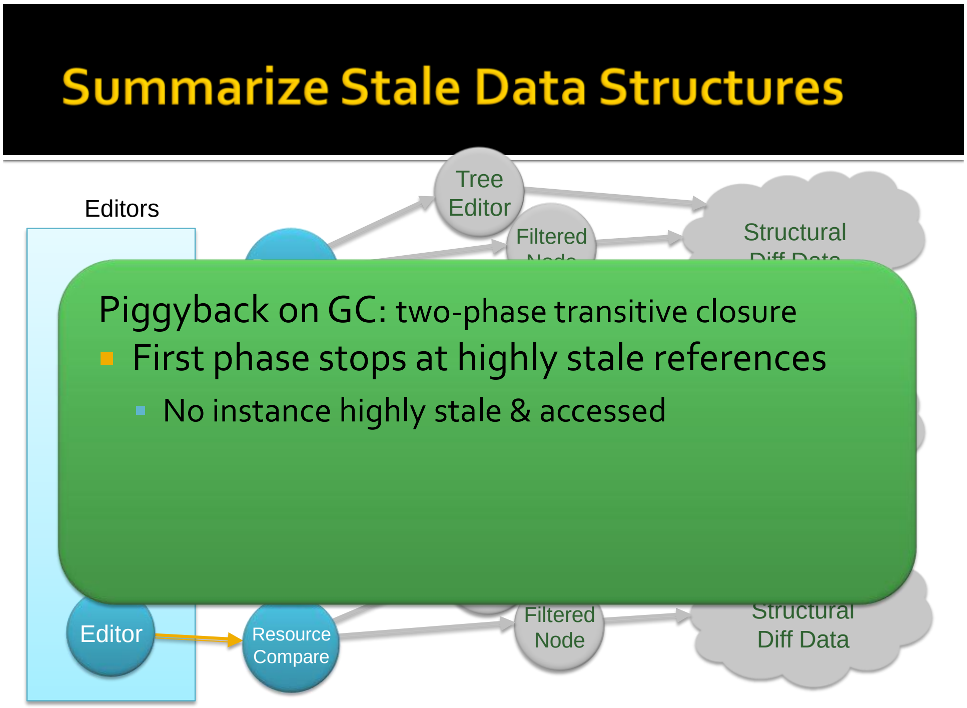
```
t = o.f;
if (t & 0x1) {
  t &= ~0x1;
  t.staleness = 0;
  o.f = t; [atomic]
}
```

Garbage collector:

```
for all objects o
  o.staleness++;
  for all fields o.f
    o.f |= 0x1;
```



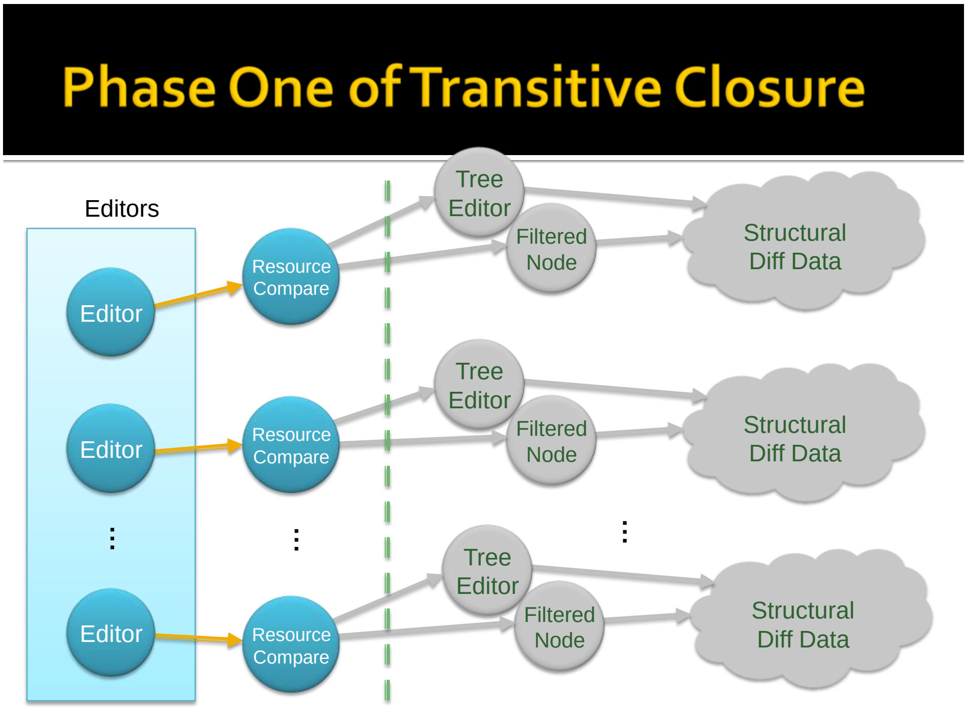
Fine-Grained Usage Tracking



Summarize Stale Data Structures

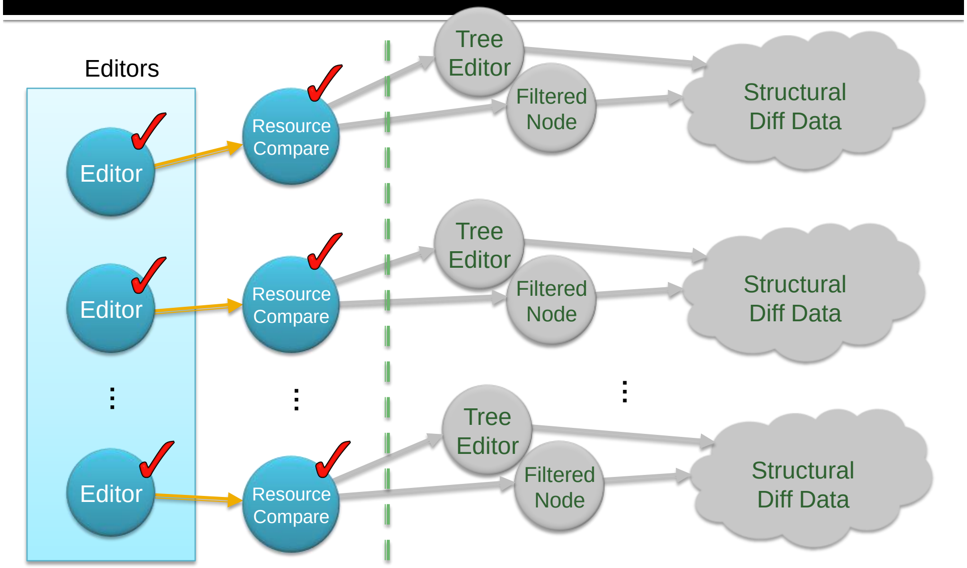
Piggyback on GC: two-phase transitive closure

- First phase stops at highly stale references
  - No instance highly stale & accessed

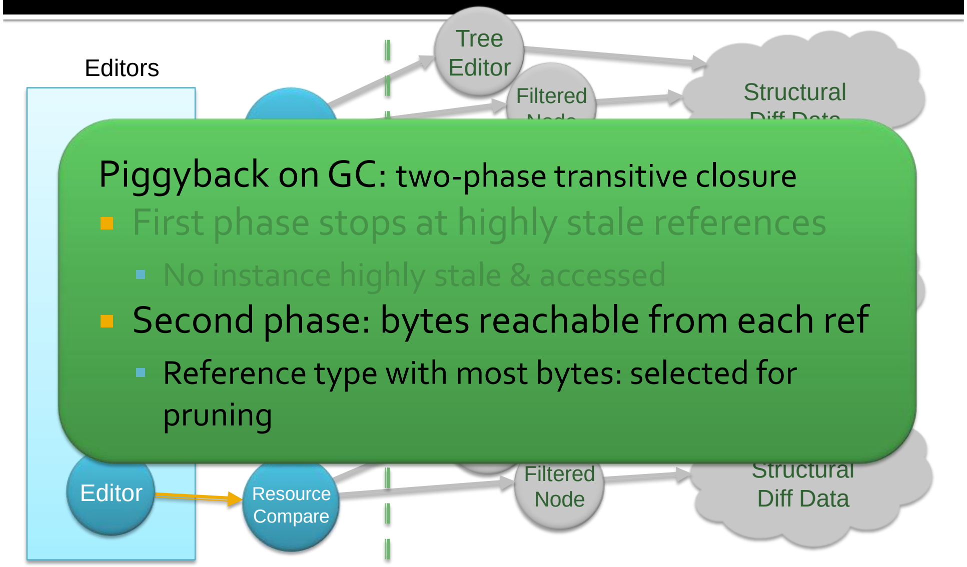


Phase One of Transitive Closure

Phase One of Transitive Closure



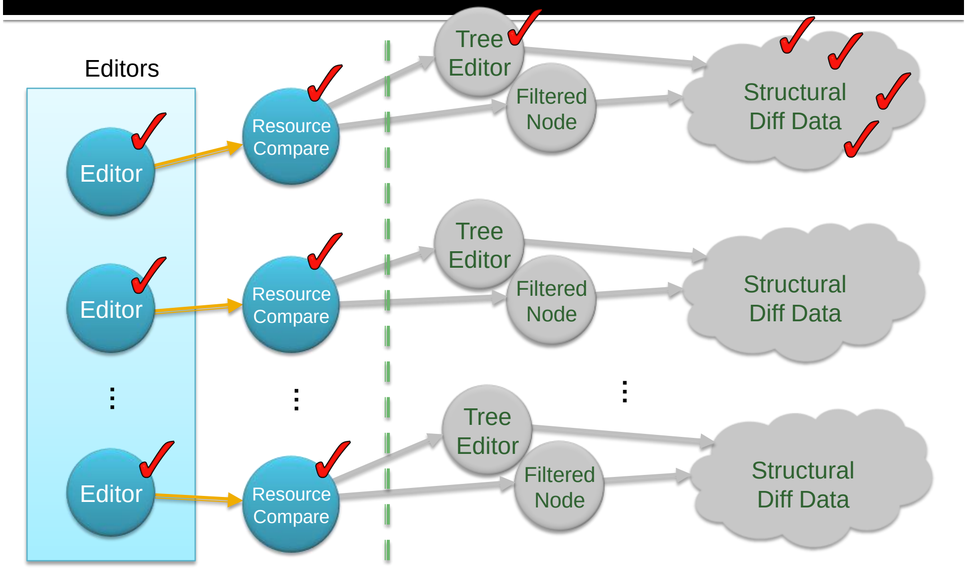
Phase Two of Transitive Closure



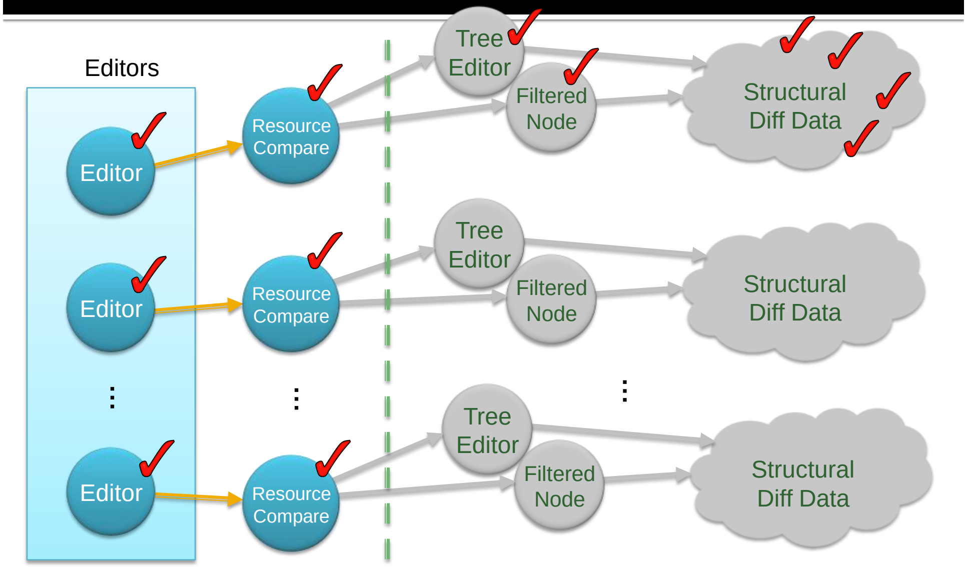
Piggyback on GC: two-phase transitive closure

- First phase stops at highly stale references
  - No instance highly stale & accessed
- Second phase: bytes reachable from each ref
  - Reference type with most bytes: selected for pruning

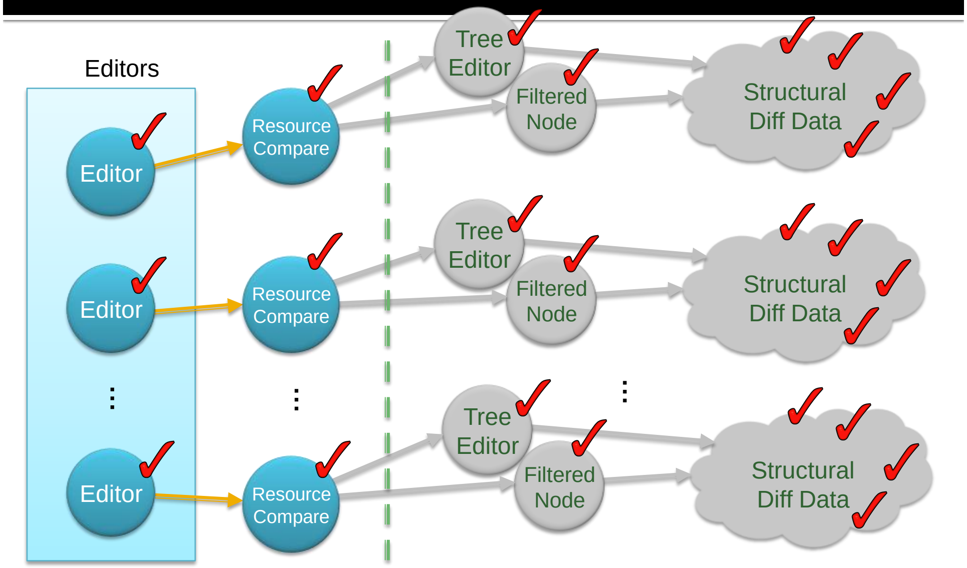
Phase Two of Transitive Closure



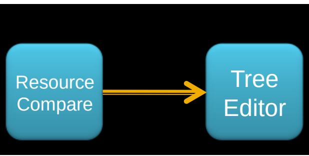
Phase Two of Transitive Closure

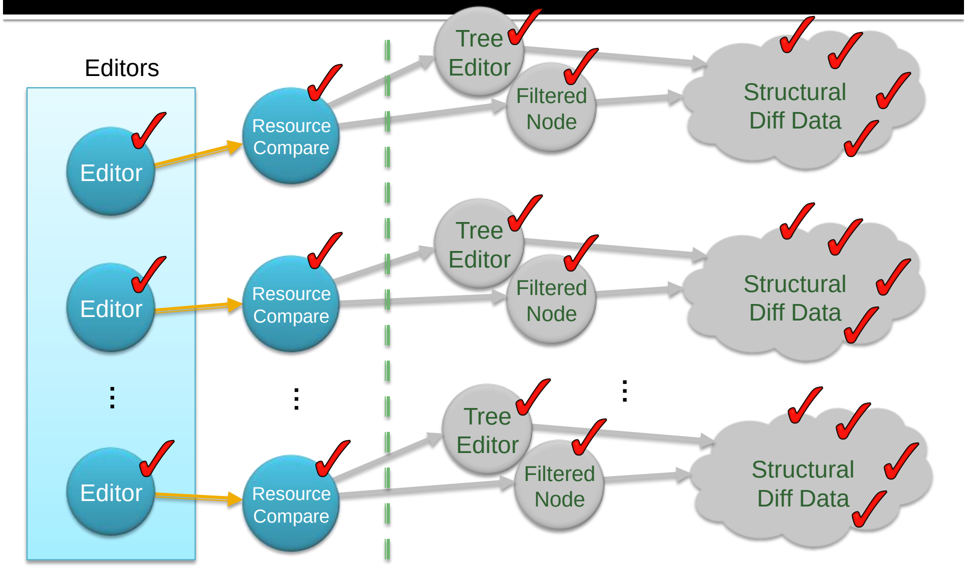


Phase Two of Transitive Closure

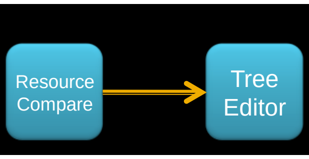


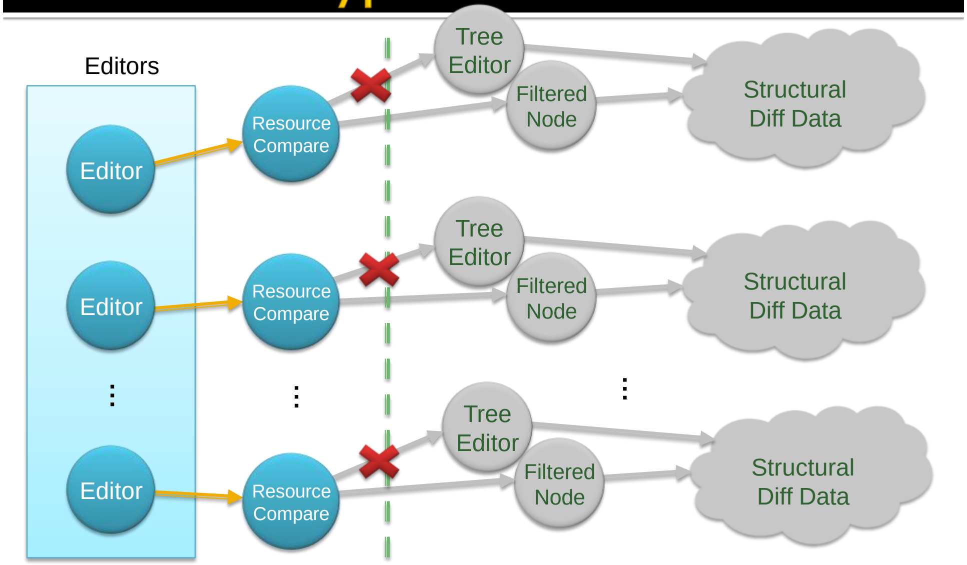
Select Reference Type



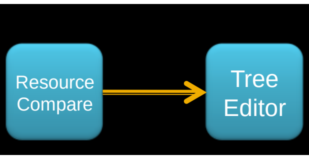


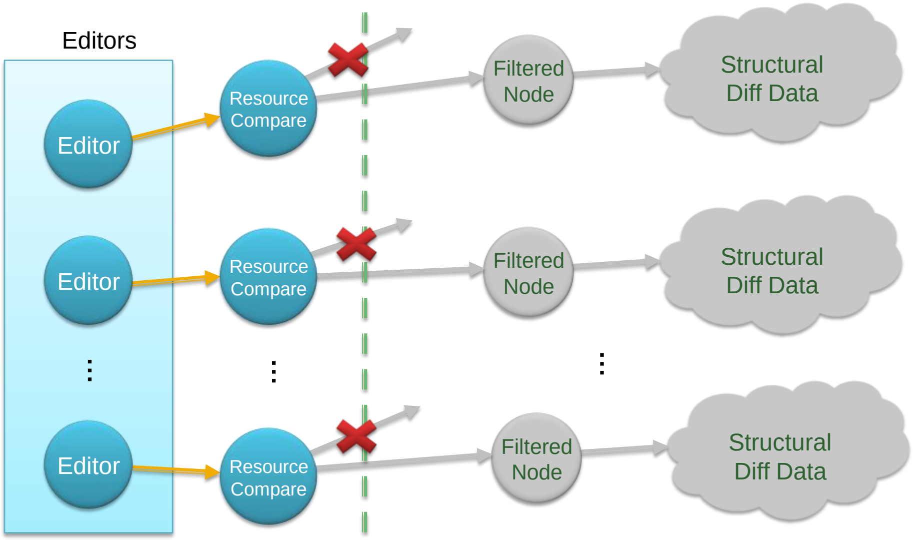
Prune Selected Reference Type

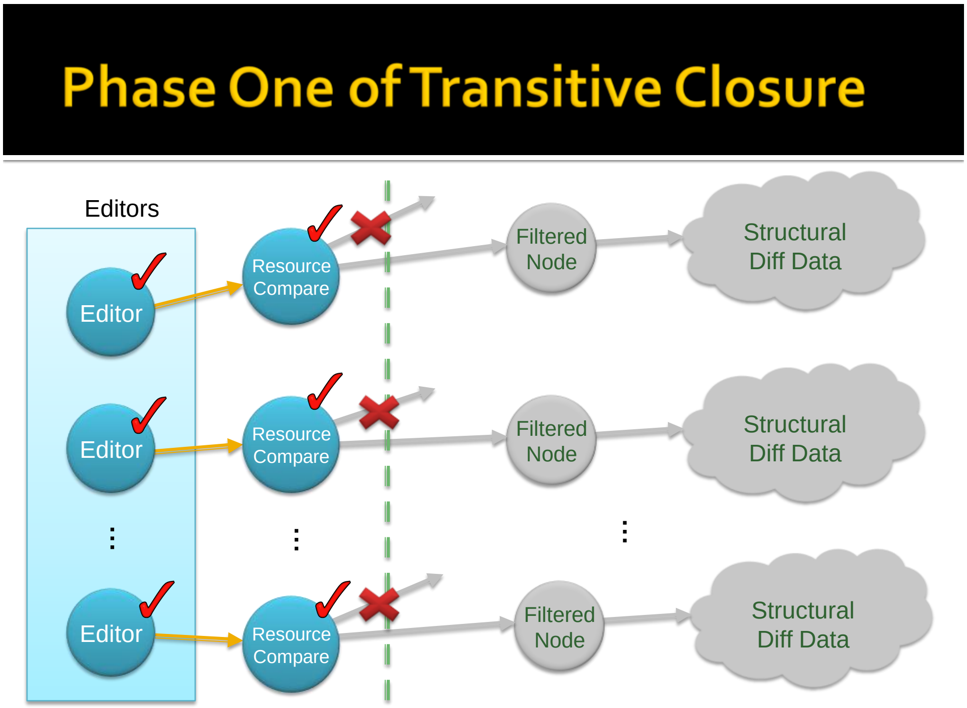




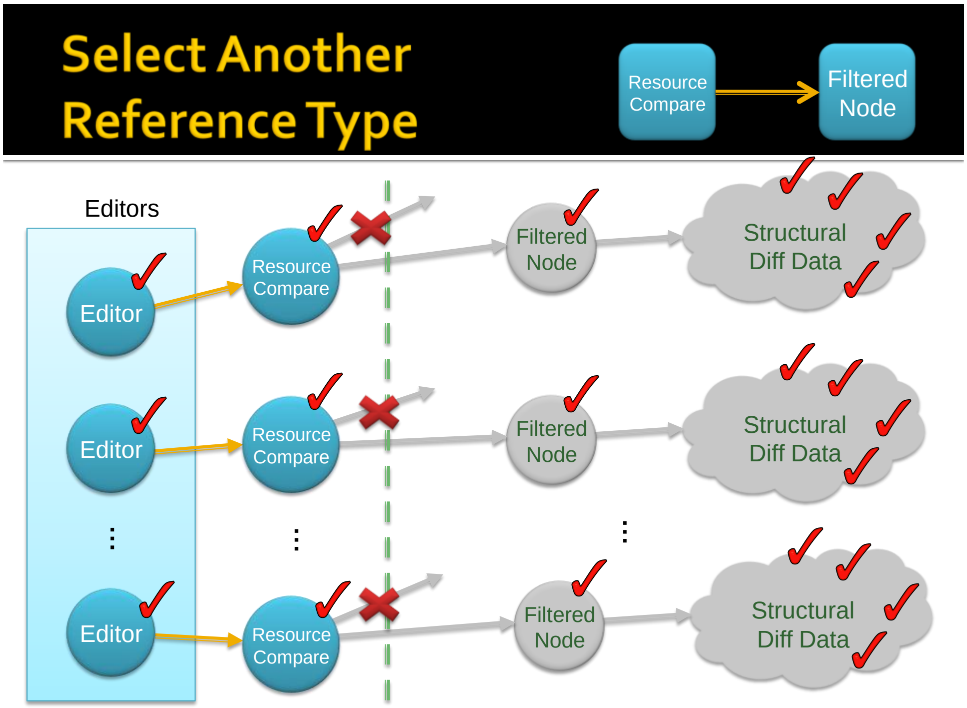
Prune Selected Reference Type





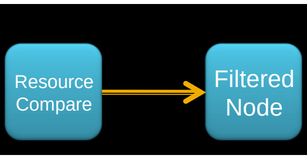


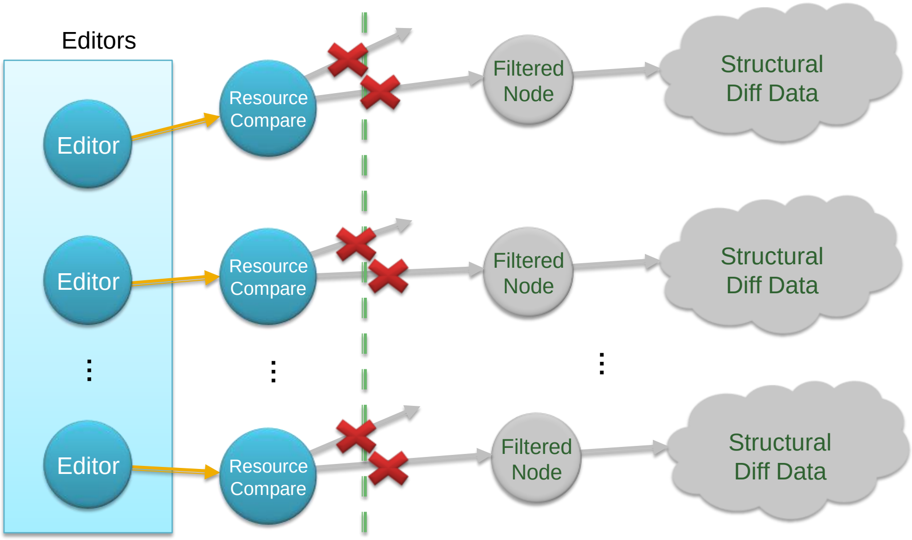
Phase One of Transitive Closure



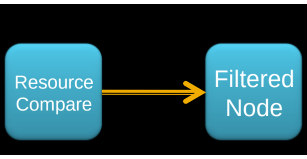
Select Another Reference Type

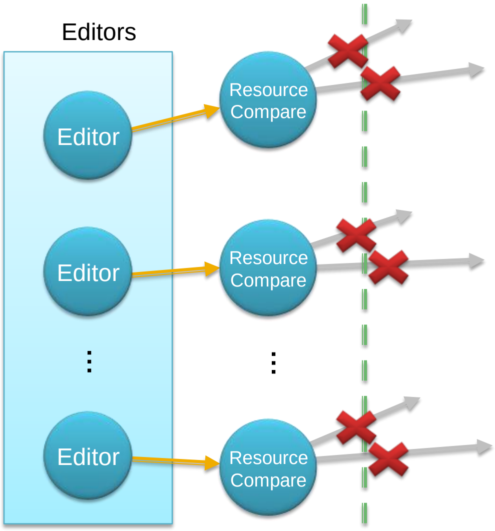
Prune Reference Type





Prune Reference Type


Resource Compare
Filtered Node


Editors
Editor
Editor
⋮
Editor
Resource Compare
Resource Compare
⋮
Resource Compare