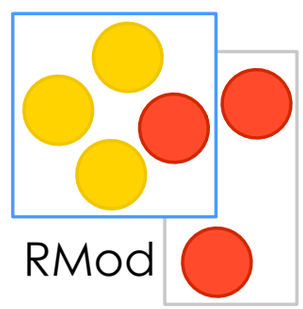
Application-Level Virtual Memory for Object-Oriented Systems

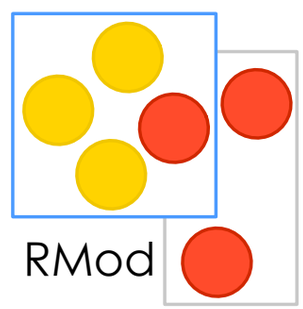
Mariano Martinez Peck



Université
Lille1
Sciences et Technologies

Application-Level Virtual Memory for Object-Oriented Systems

Mariano Martinez Peck



Université
Lille1
Sciences et Technologies

Outline

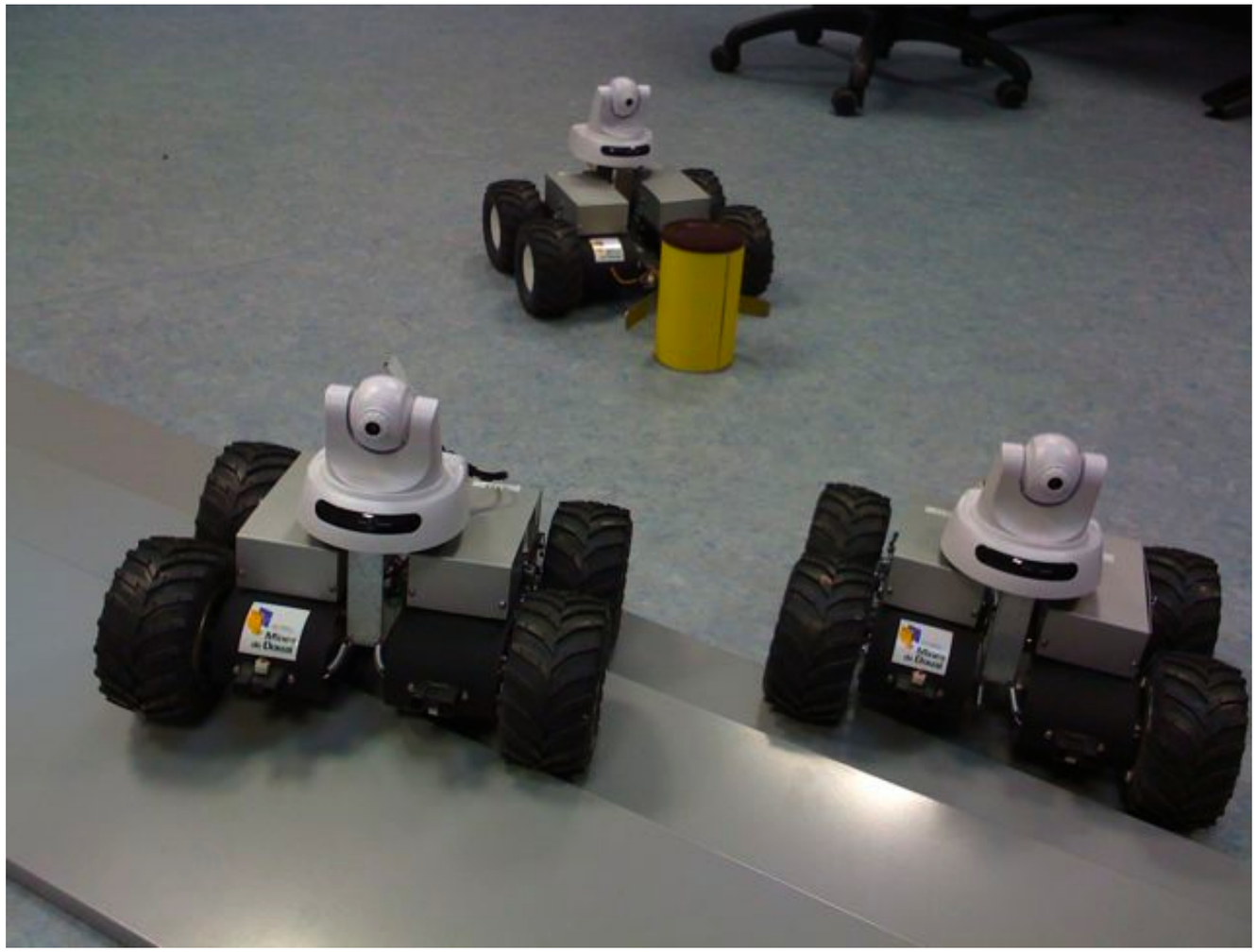
1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Context

Context



Context


AT&T 3G
7:42 PM
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
NYTimes
Notes
Clock
Calculator
Facebook
Settings
App Store
Compass
Phone
Mail
Safari
iPod

Context


AT&T 3G
7:42 PM
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
NYTimes
Notes
Clock
Calculator
Facebook
Settings
App Store
Compass
Phone
Mail
Safari
iPod

Context


AT&T 3G
7:42 PM
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
NYTimes
Notes
Clock
Calculator
Facebook
Settings
App Store
Compass
Phone
Mail
Safari
iPod

Context


AT&T 3G
7:42 PM
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
NYTimes
Notes
Clock
Calculator
Facebook
Settings
App Store
Compass
Phone
Mail
Safari
iPod

We need to optimize memory management

Context

- Object-oriented programming. Why?
  - Most modern and widespread.
  - Memory is usually automatically managed (GC).
- Dynamic languages. Why?
  - Powerful.
  - More and more used.

In this context of OOP, is GC actually enough to optimize memory management?

% used objects ?
% unused objects ?

% used memory ?
% unused memory ?

% used objects
% unused objects

% used memory
% unused memory

(web app + CMS)

(standalone + lots of tools + large)
(174 pack. 1364 class. 121010 LOC)

(mobile app)

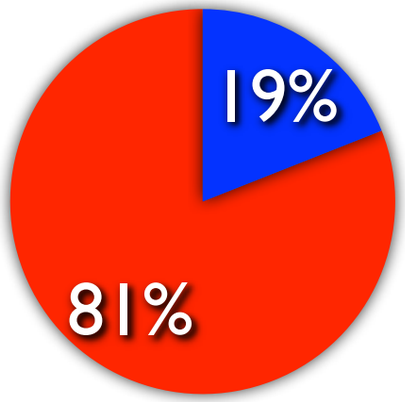
(infrastructure)

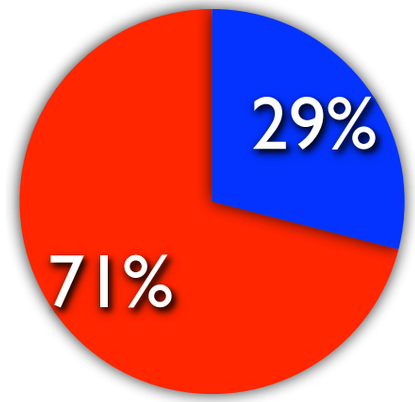
% used objects
% unused objects

% used memory
% unused memory

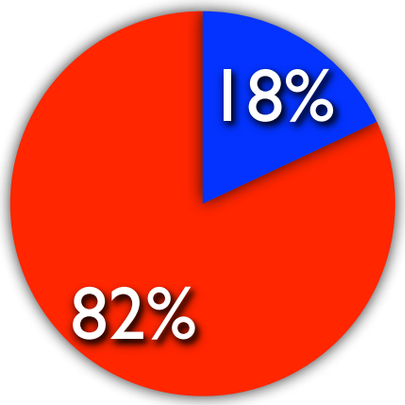
DBXTALK

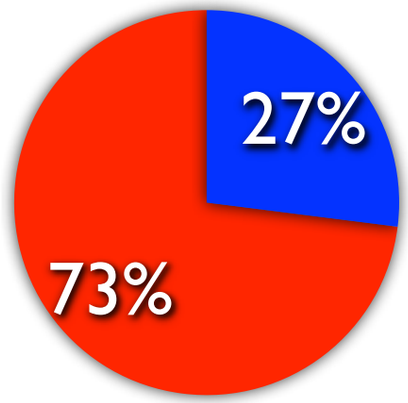
(web app + CMS)

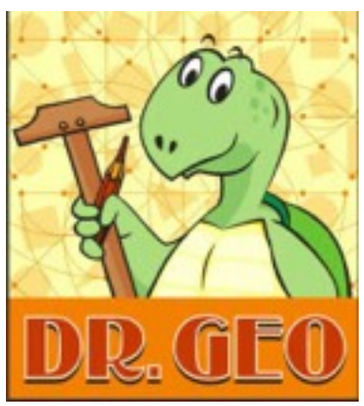




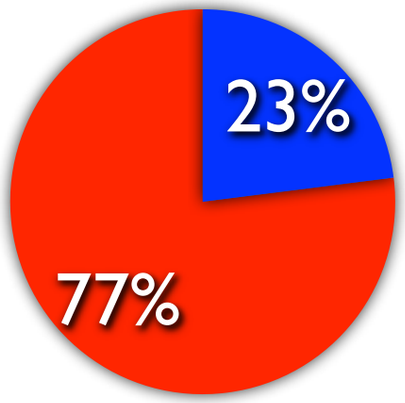
(standalone + lots of tools + large)
(174 pack. 1364 class. 121010 LOC)

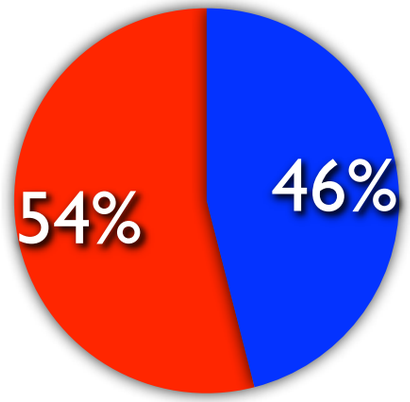






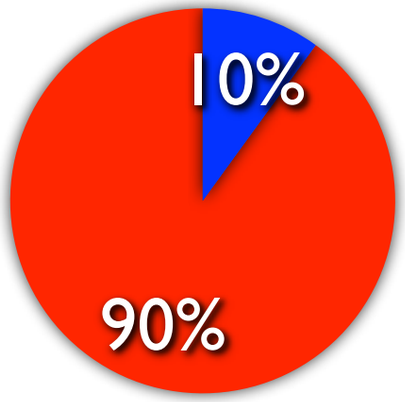
(mobile app)

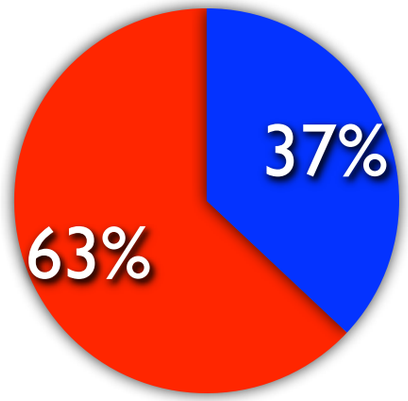




Pharo

(infrastructure)



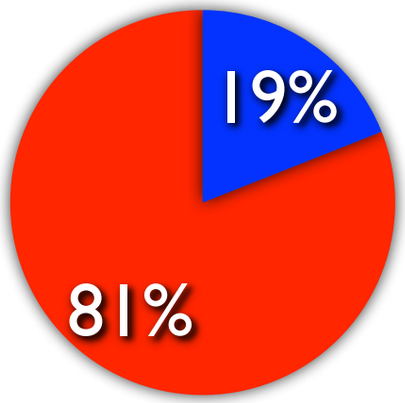


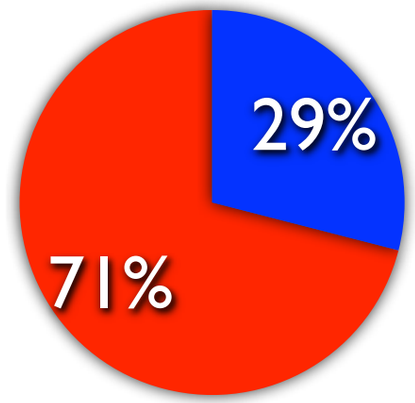
% used objects
% unused objects

% used memory
% unused memory

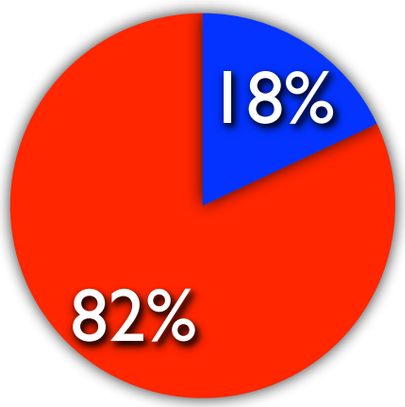
DBXTALK

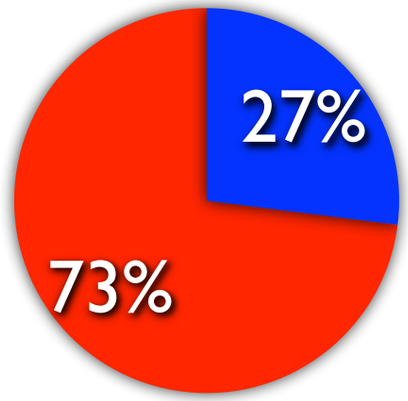
(web app + CMS)



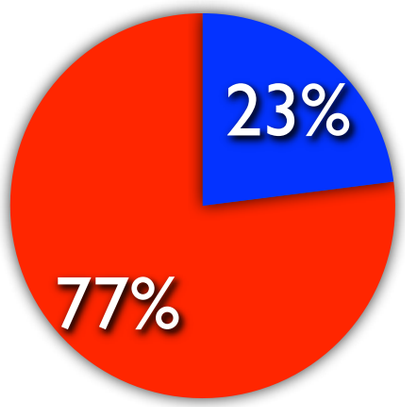


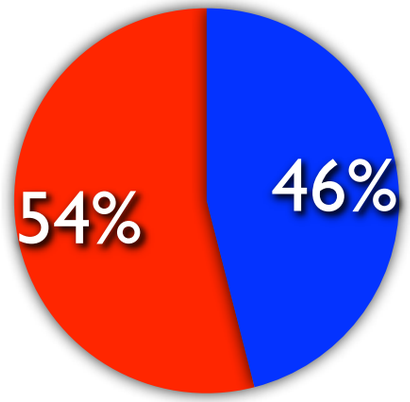
(standalone + lots of tools + large)
(174 pack. 1364 class. 121010 LOC)





(mobile app)





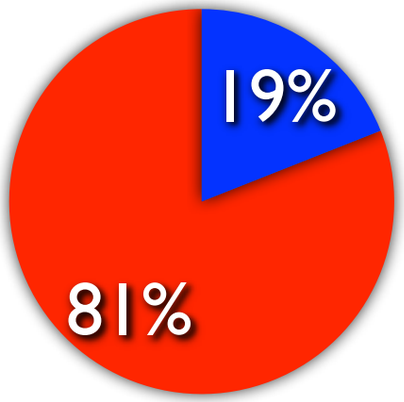
In average, %80 of the
objects are unused.

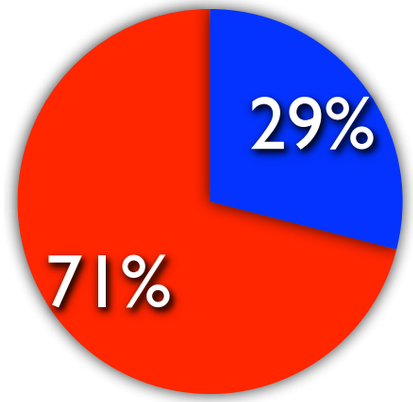
% used objects
% unused objects

% used memory
% unused memory

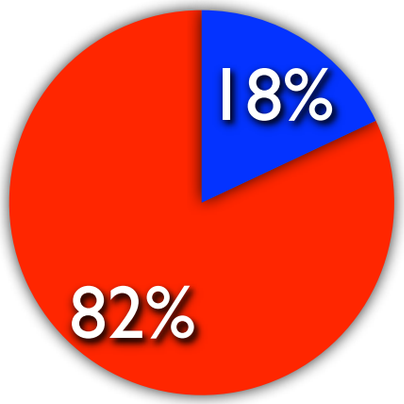
DBXTALK

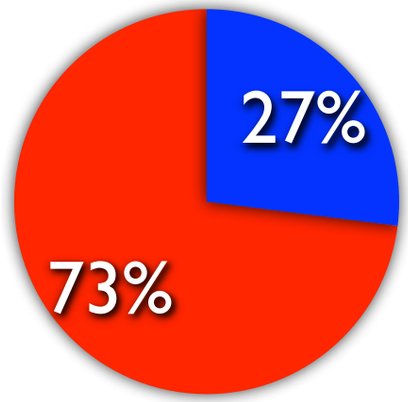
(web app + CMS)



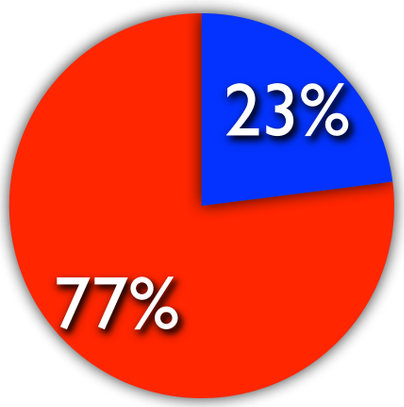


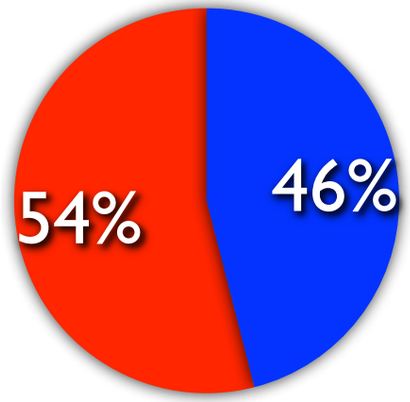
(standalone + lots of tools + large)
(174 pack. 1364 class. 121010 LOC)





(mobile app)





In average, %80 of the **objects** are unused.

In average, 77% of the **memory** is unused.

Unused objects

- Referenced.
- GC cannot collect them!
- Waste primary memory.

Unused objects

- Referenced.
- GC cannot collect them!
- Waste primary memory.

No, the GC is not enough

OS' Virtual Memory is not the answer

- It only swaps pages.
- Developers cannot easily influence it.

Thesis statement

A virtual memory for dynamic OOP languages should be:

* Application-aware
* Efficient

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

A model for efficient application-level virtual memory:

- Designed for dynamic OOP
- Based on the swapping of object graphs

Primary memory

Swap out unused object graphs

Swap in needed object graphs

Secondary memory

file

database

Primary memory

Swap out unused object graphs

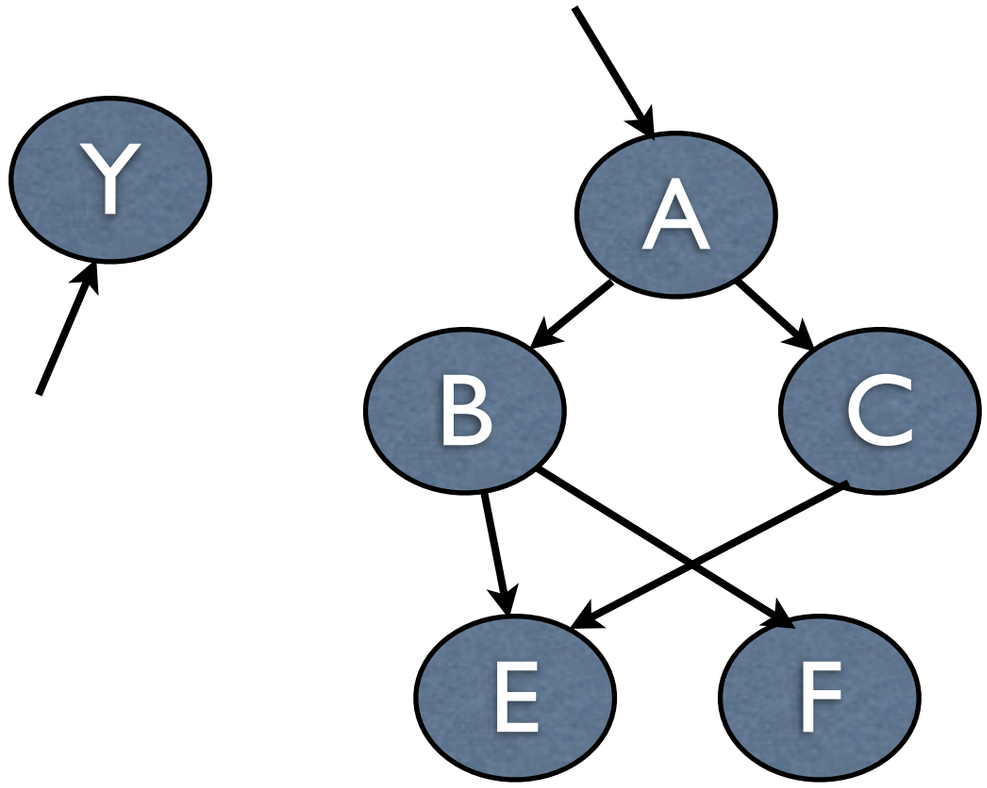
Swap in needed object graphs

Secondary memory

file

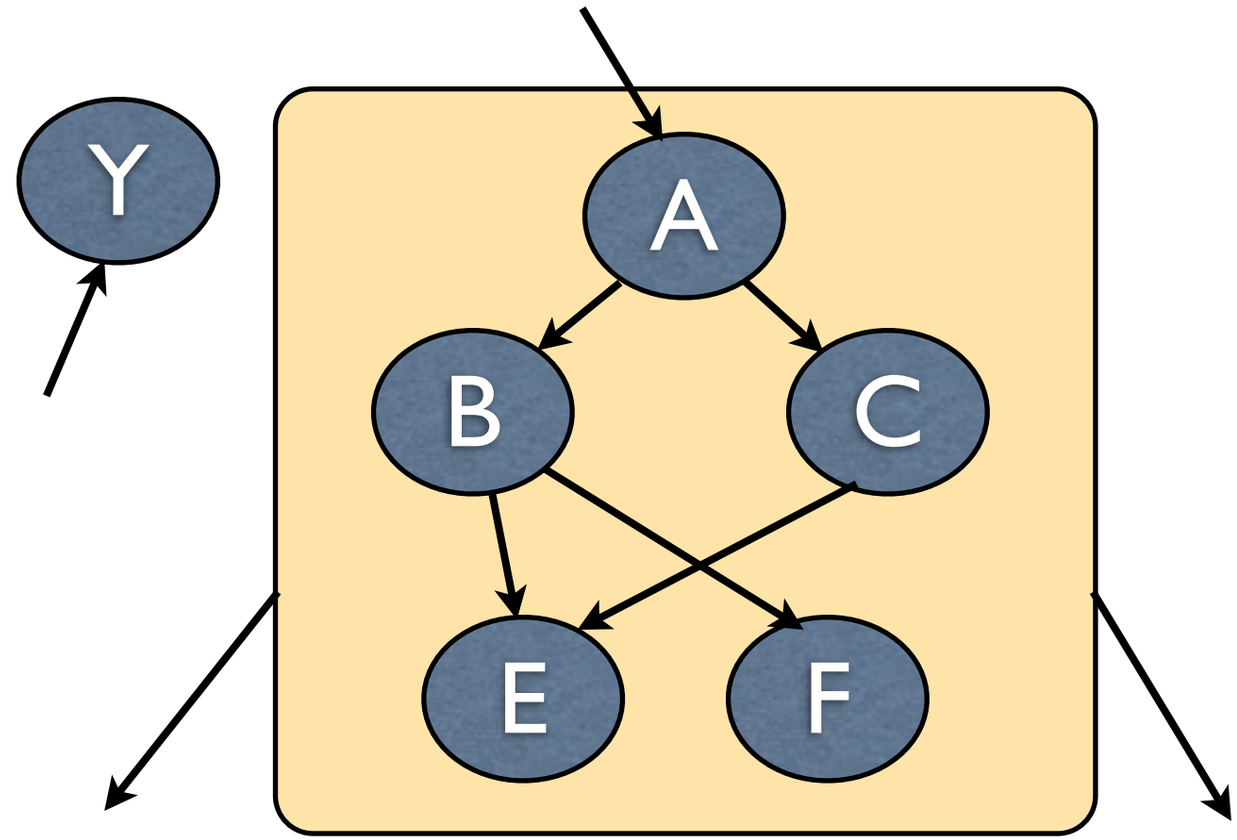
database

Primary memory



Secondary memory

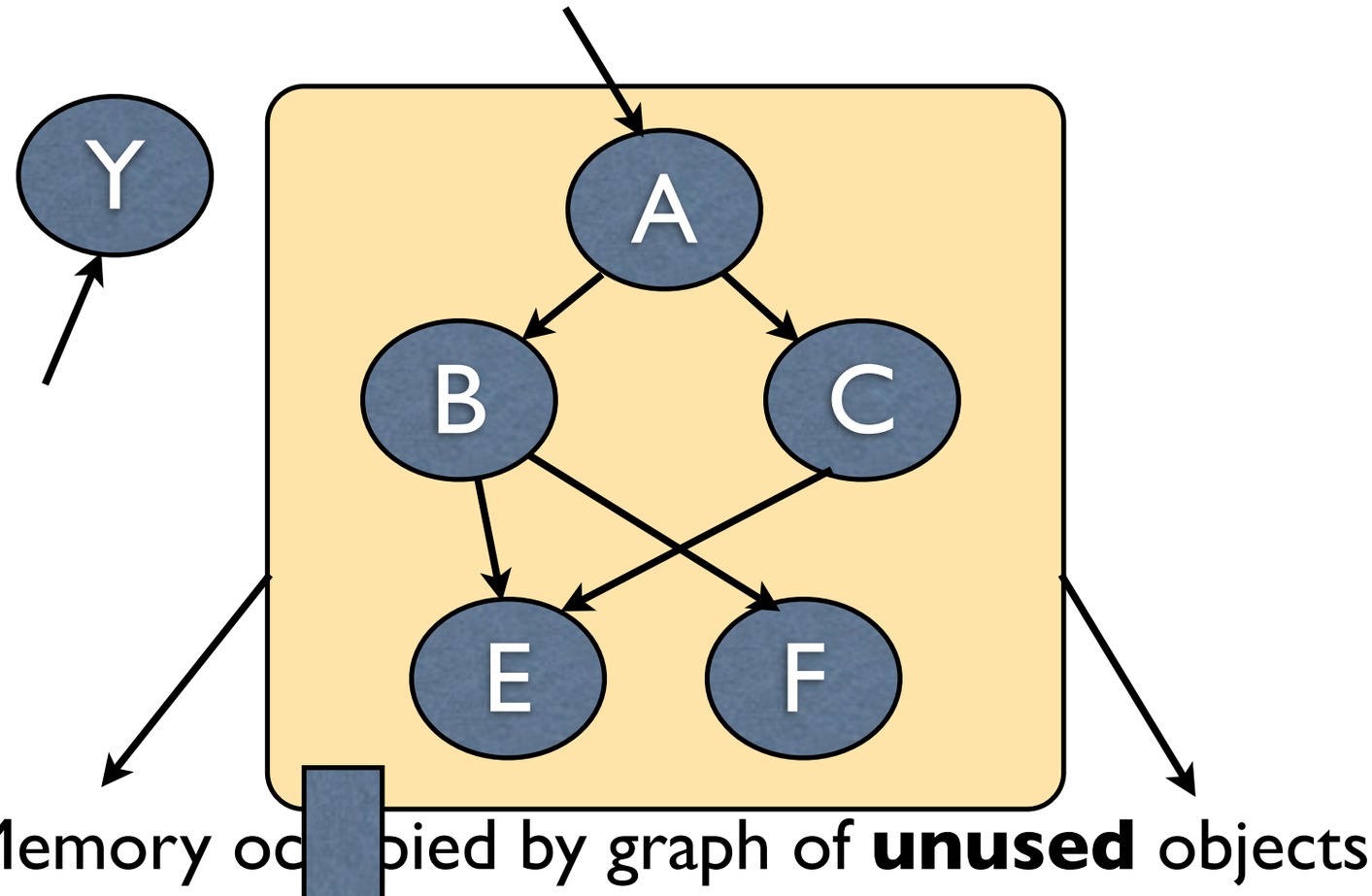
Primary memory



Memory occupied by graph of **unused** objects

Secondary memory

Primary memory

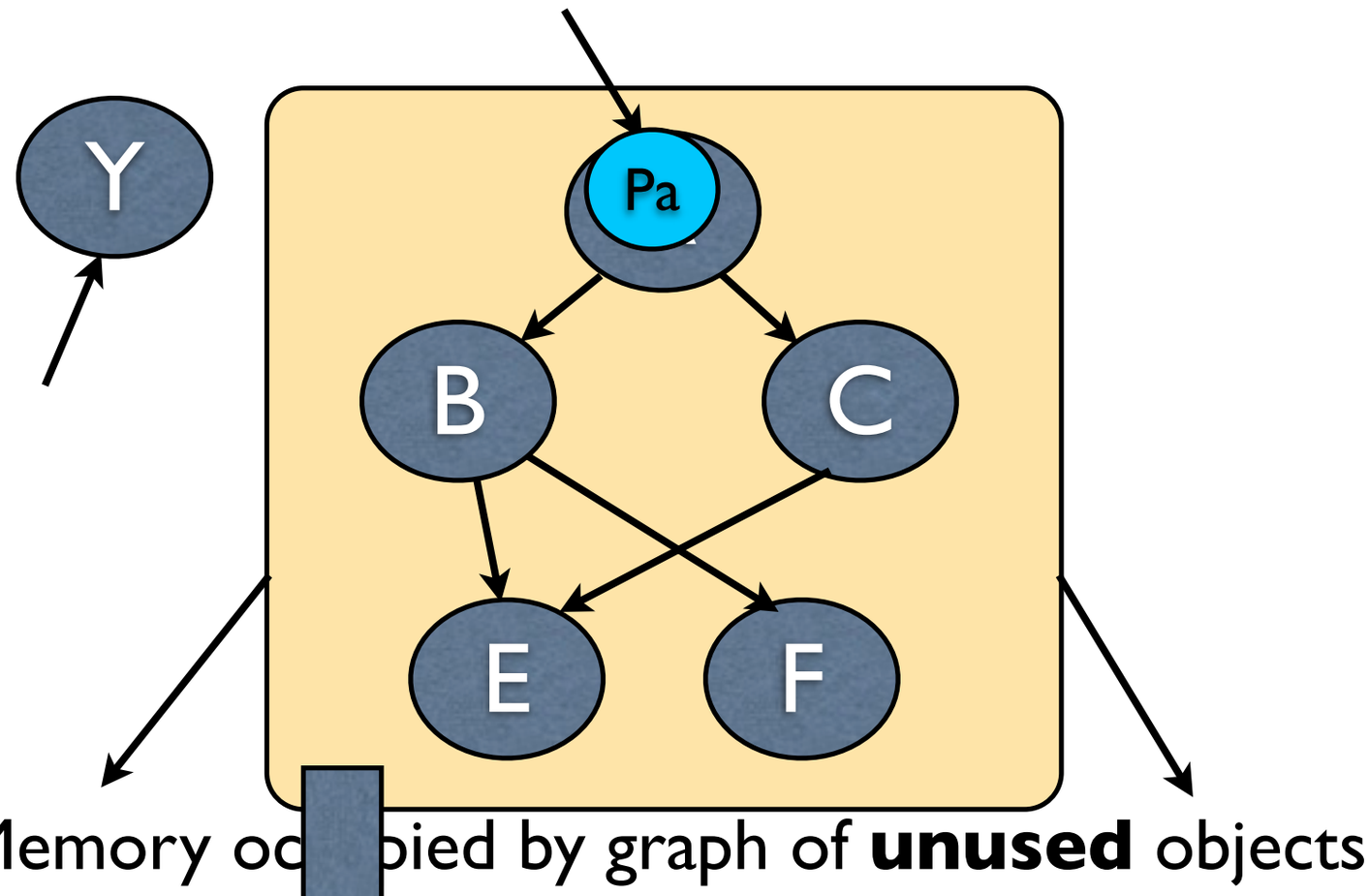


Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory

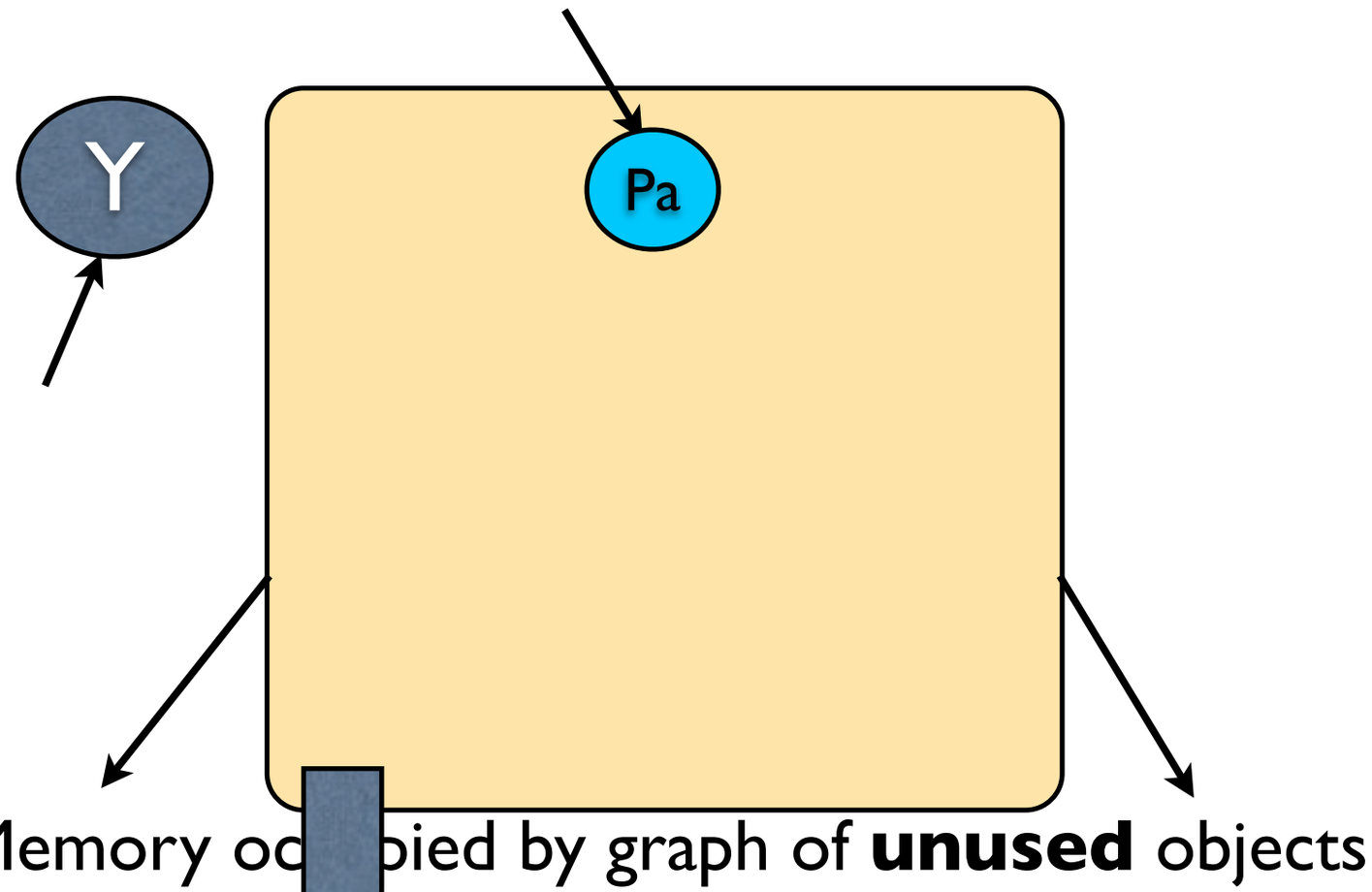


Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory

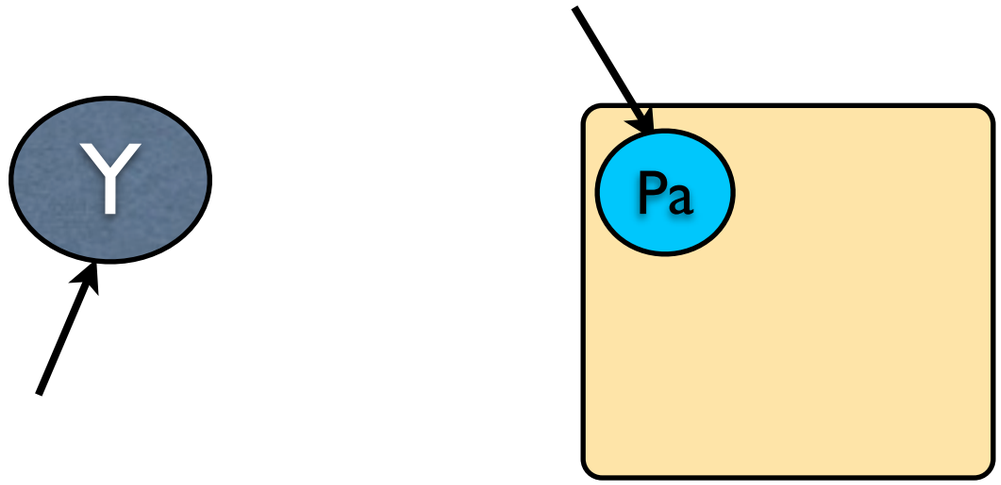


Secondary memory

Swap out

bytes (A,B,C,E,F)

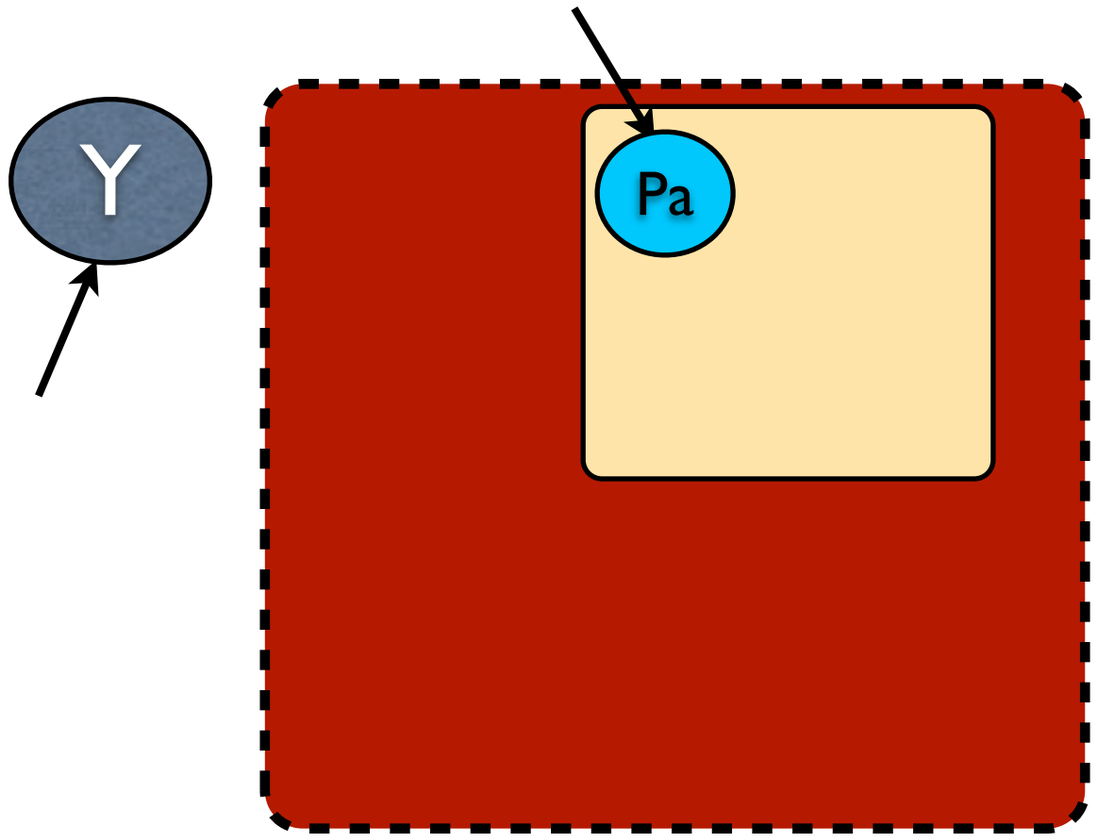
Primary memory



Secondary memory

bytes (A,B,C,E,F)

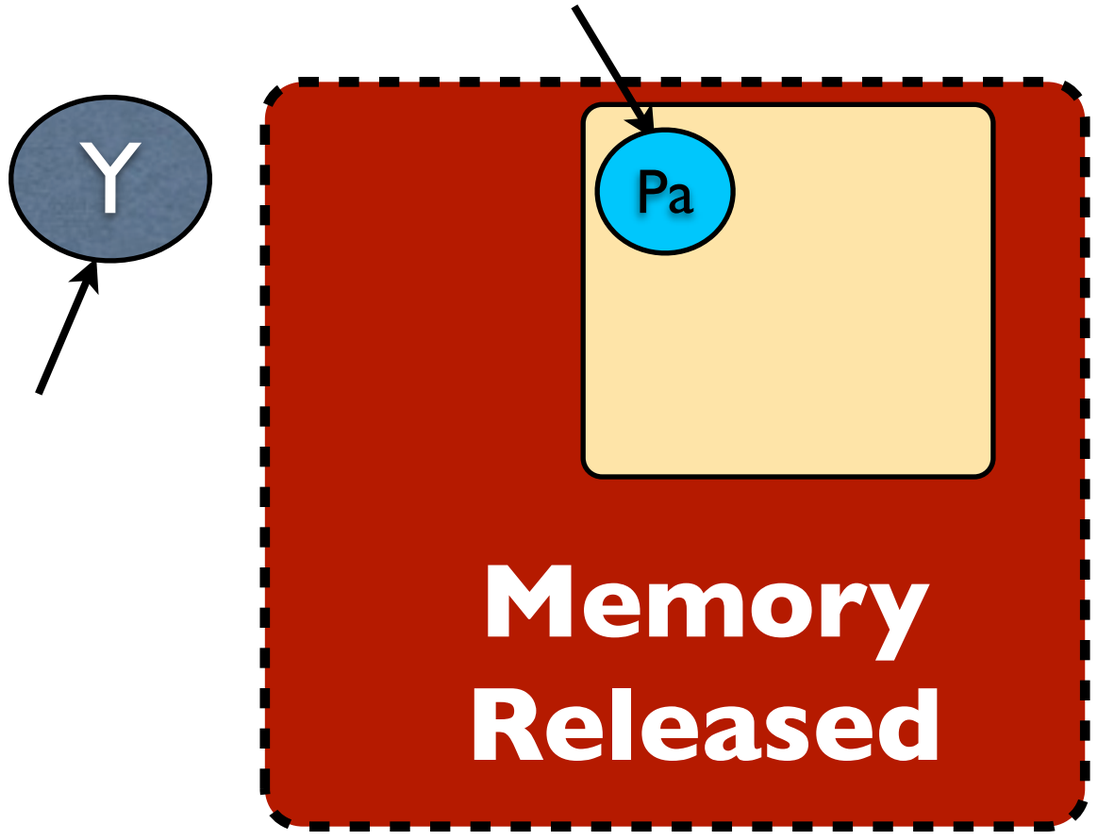
Primary memory



Secondary memory

bytes (A,B,C,E,F)

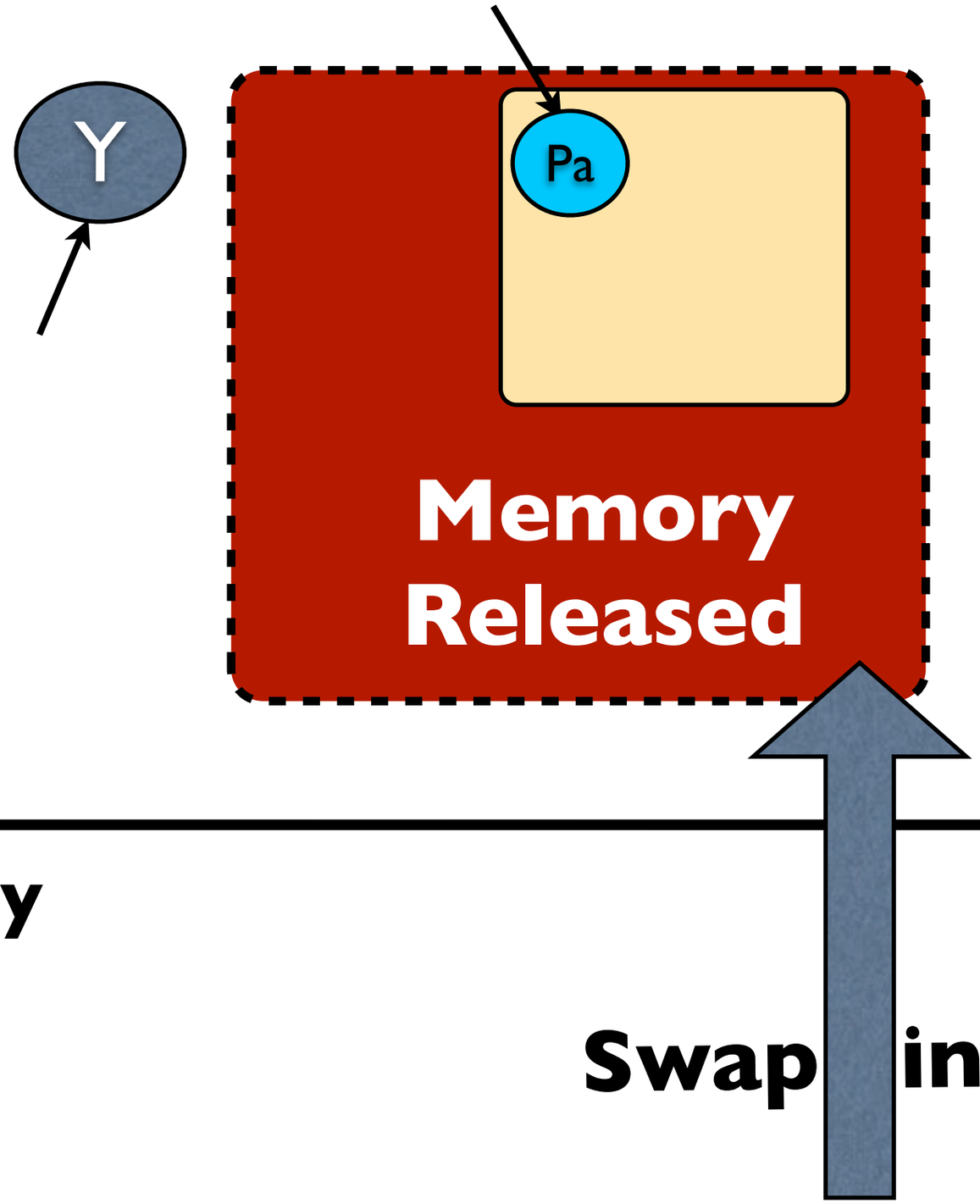
Primary memory



Secondary memory

bytes (A,B,C,E,F)

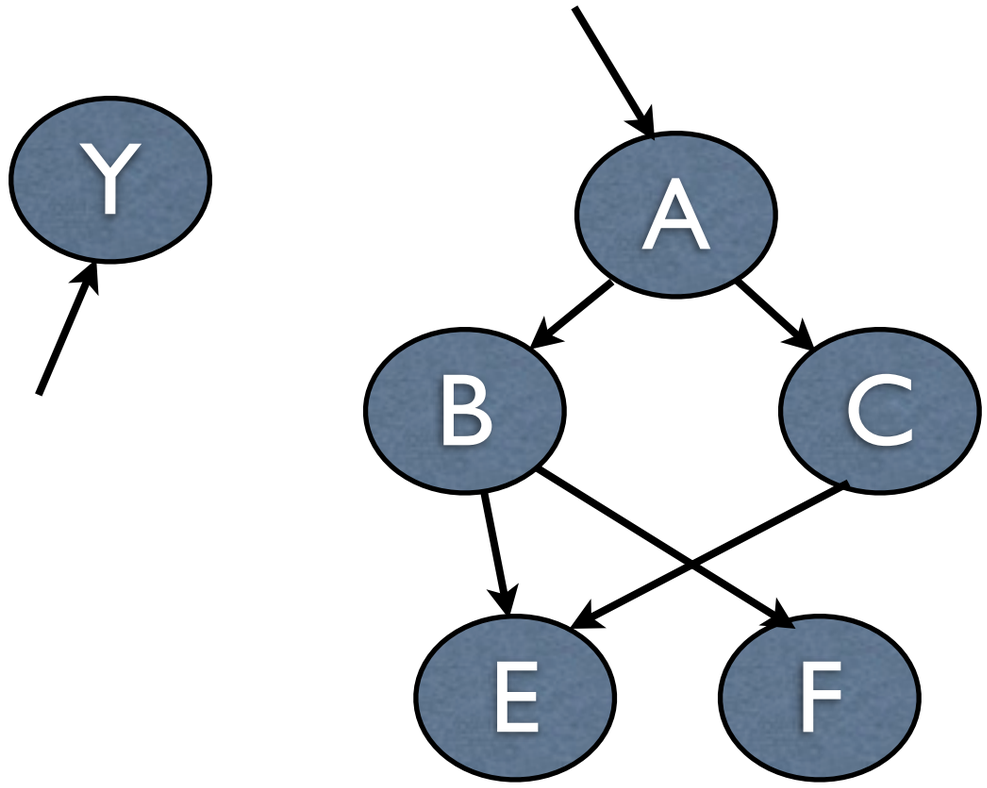
Primary memory



Secondary memory

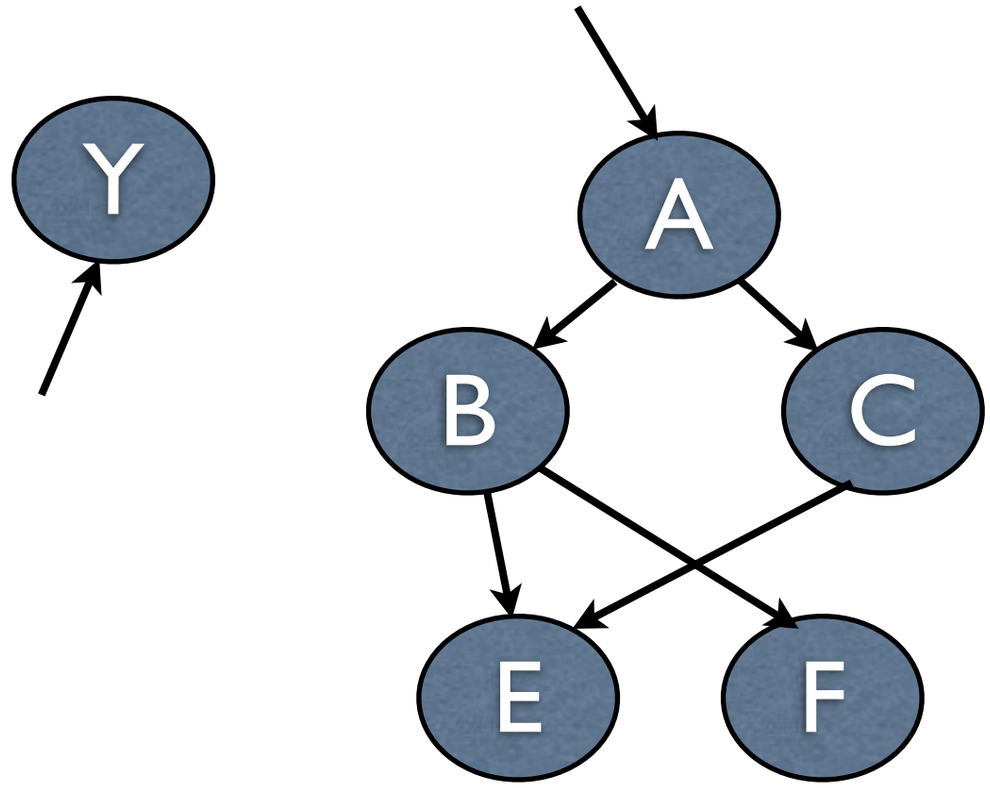
bytes (A,B,C,E,F)

Primary memory



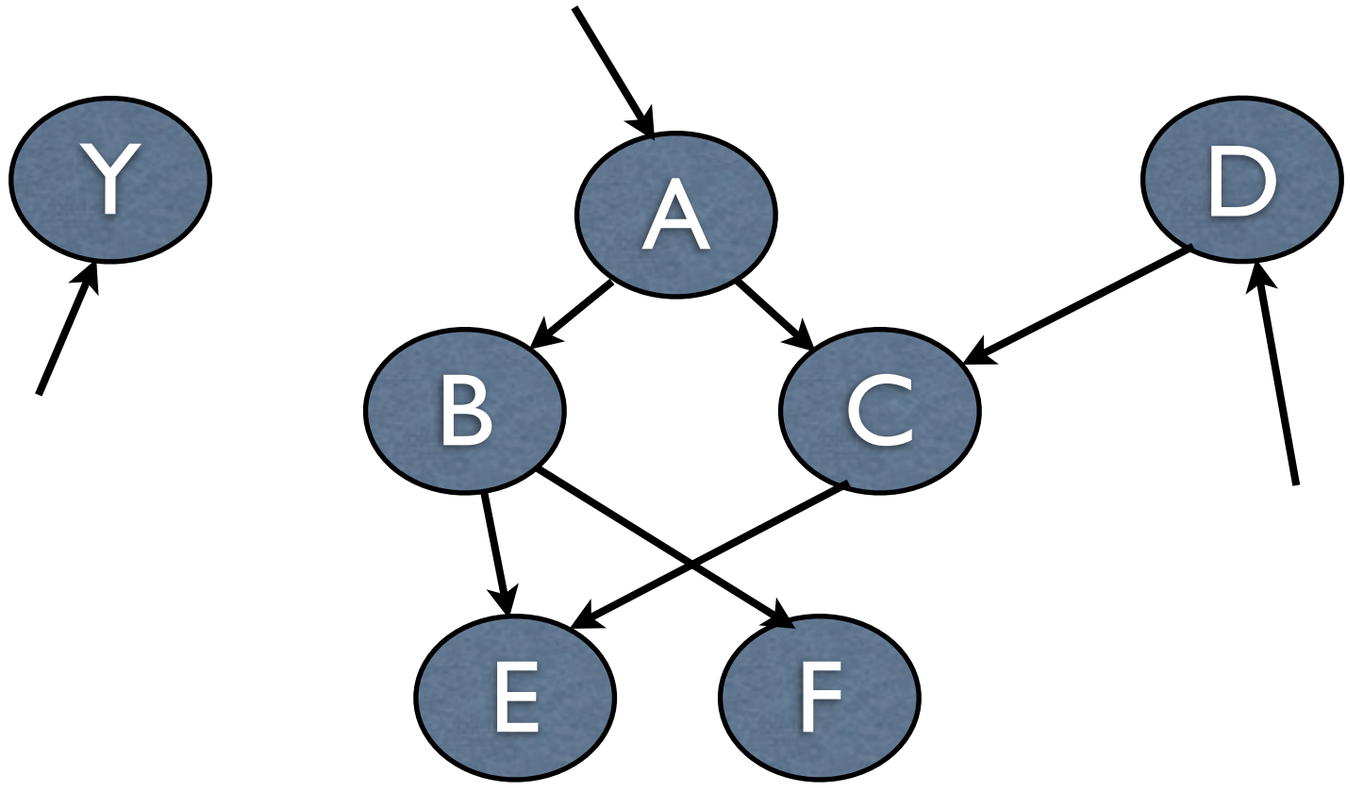
Secondary memory

Primary memory



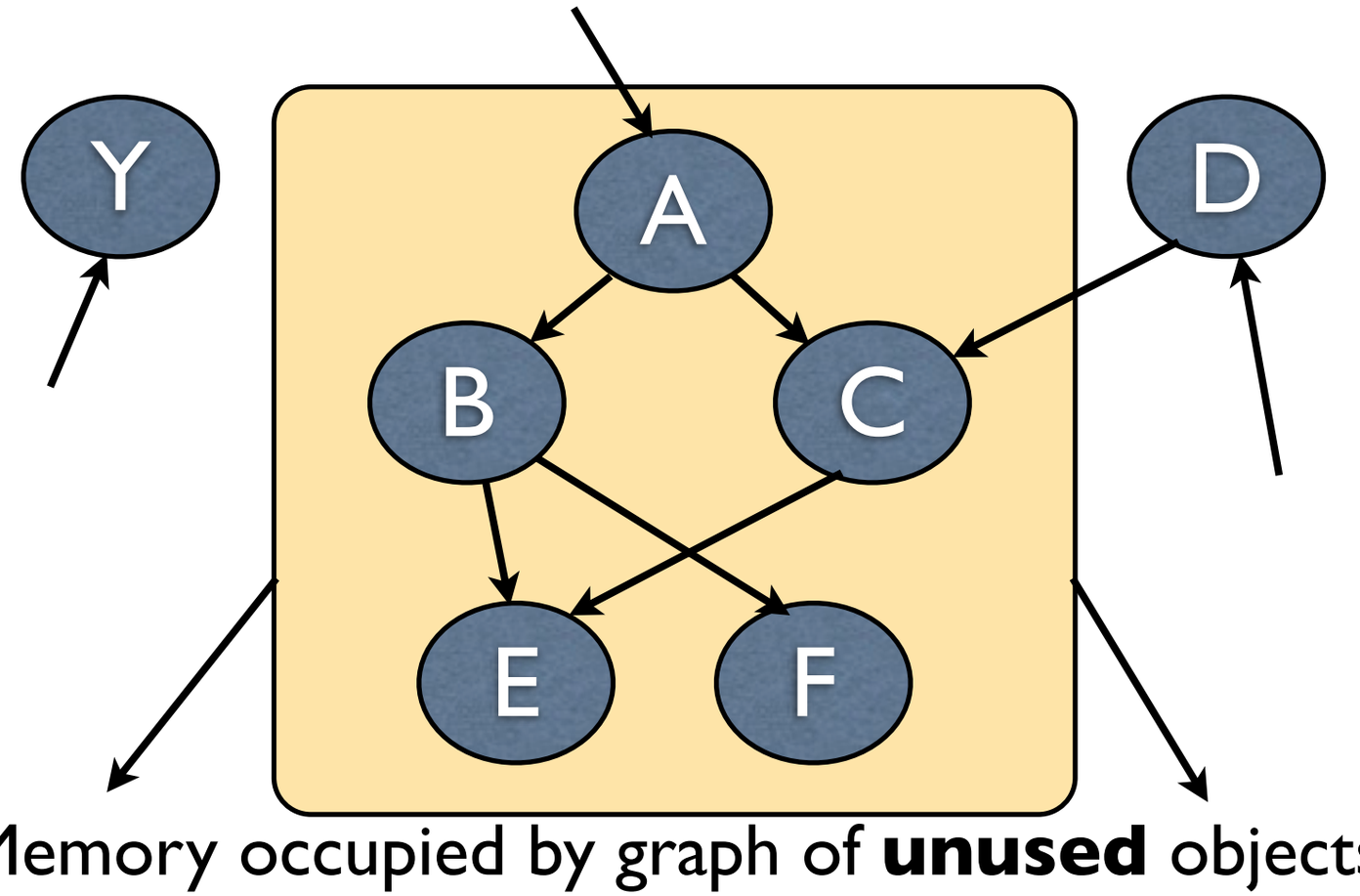
Secondary memory

Primary memory



Secondary memory

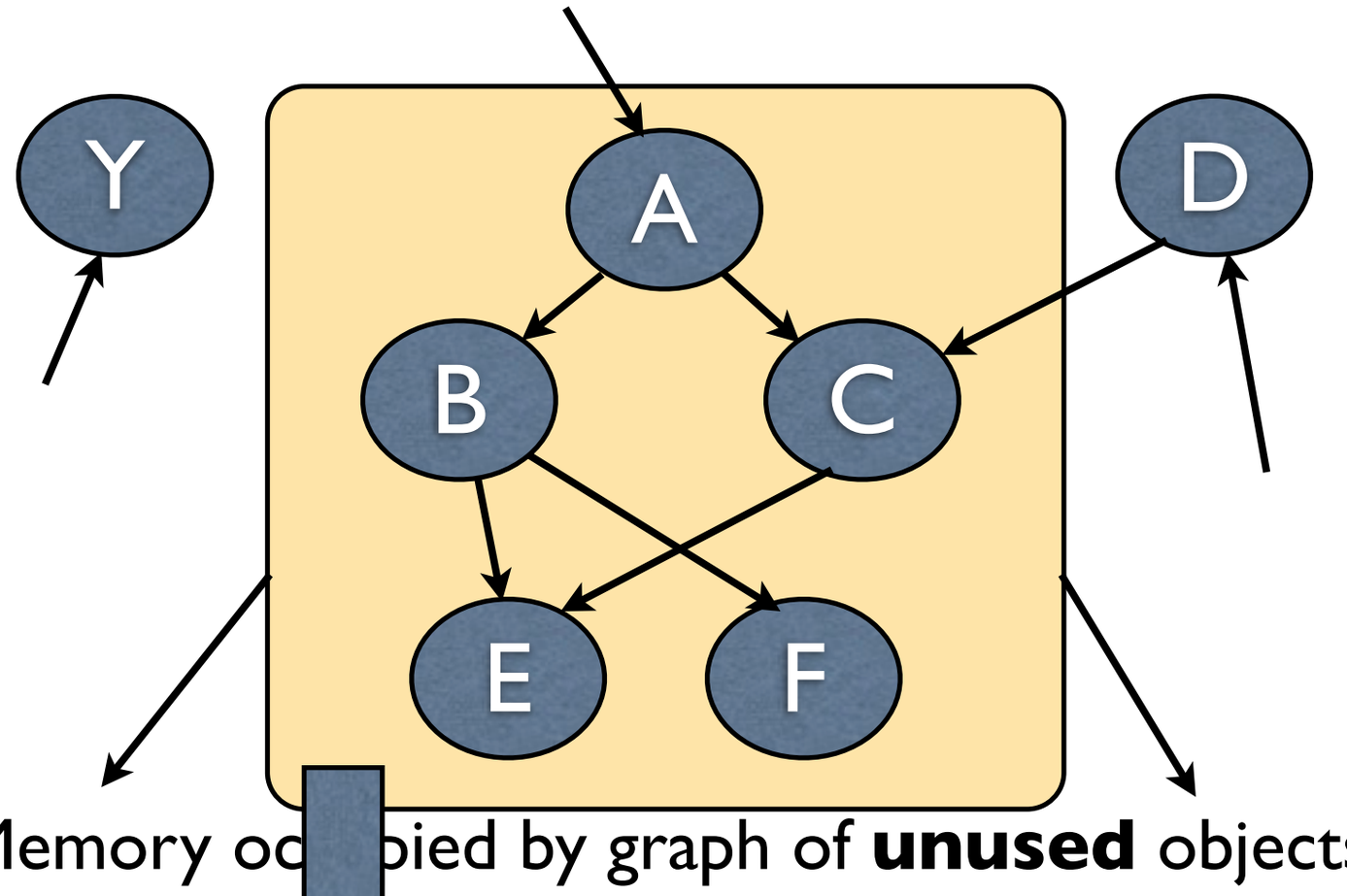
Primary memory



Memory occupied by graph of **unused** objects

Secondary memory

Primary memory



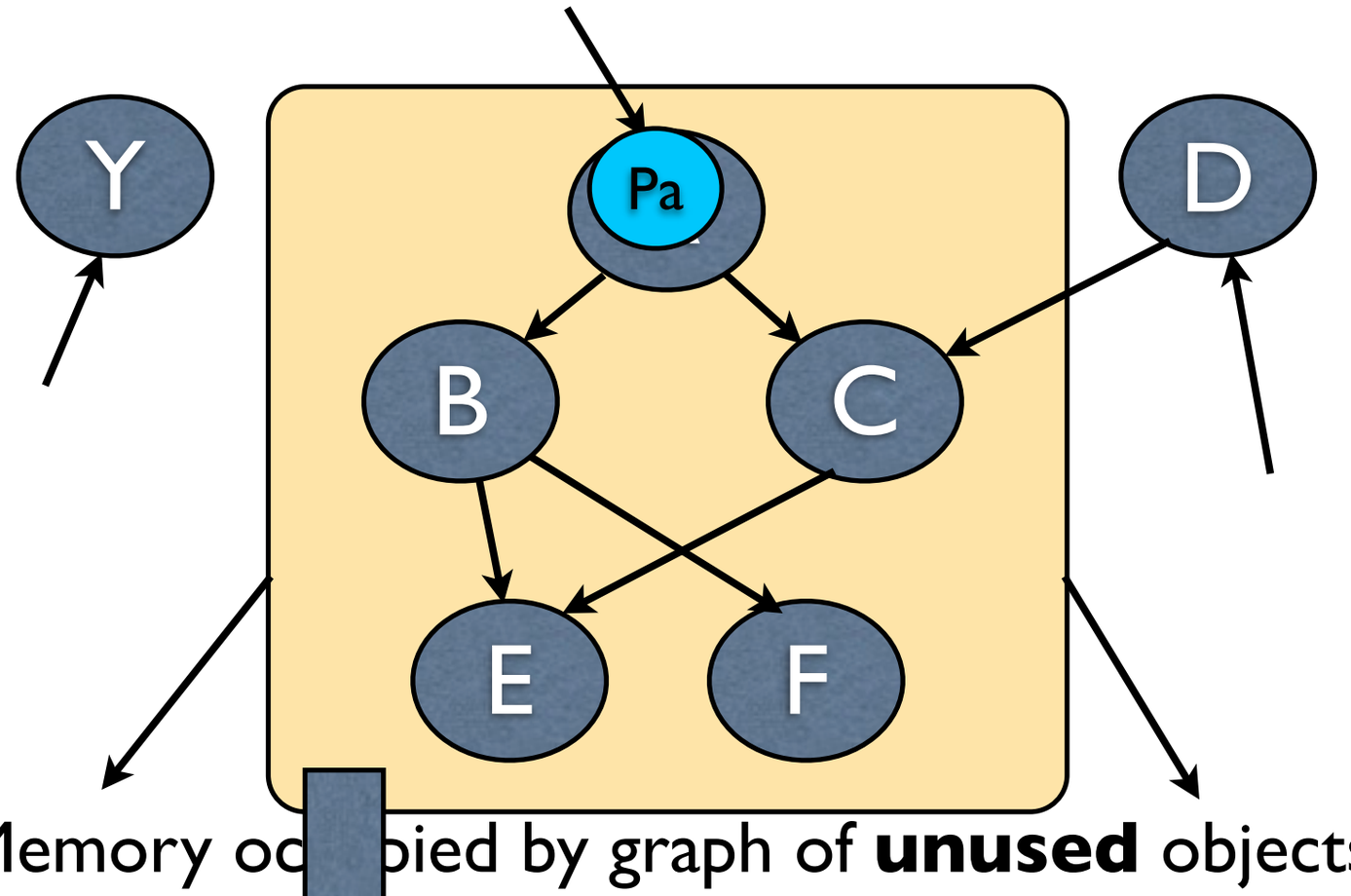
Memory occupied by graph of **unused** objects

Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory

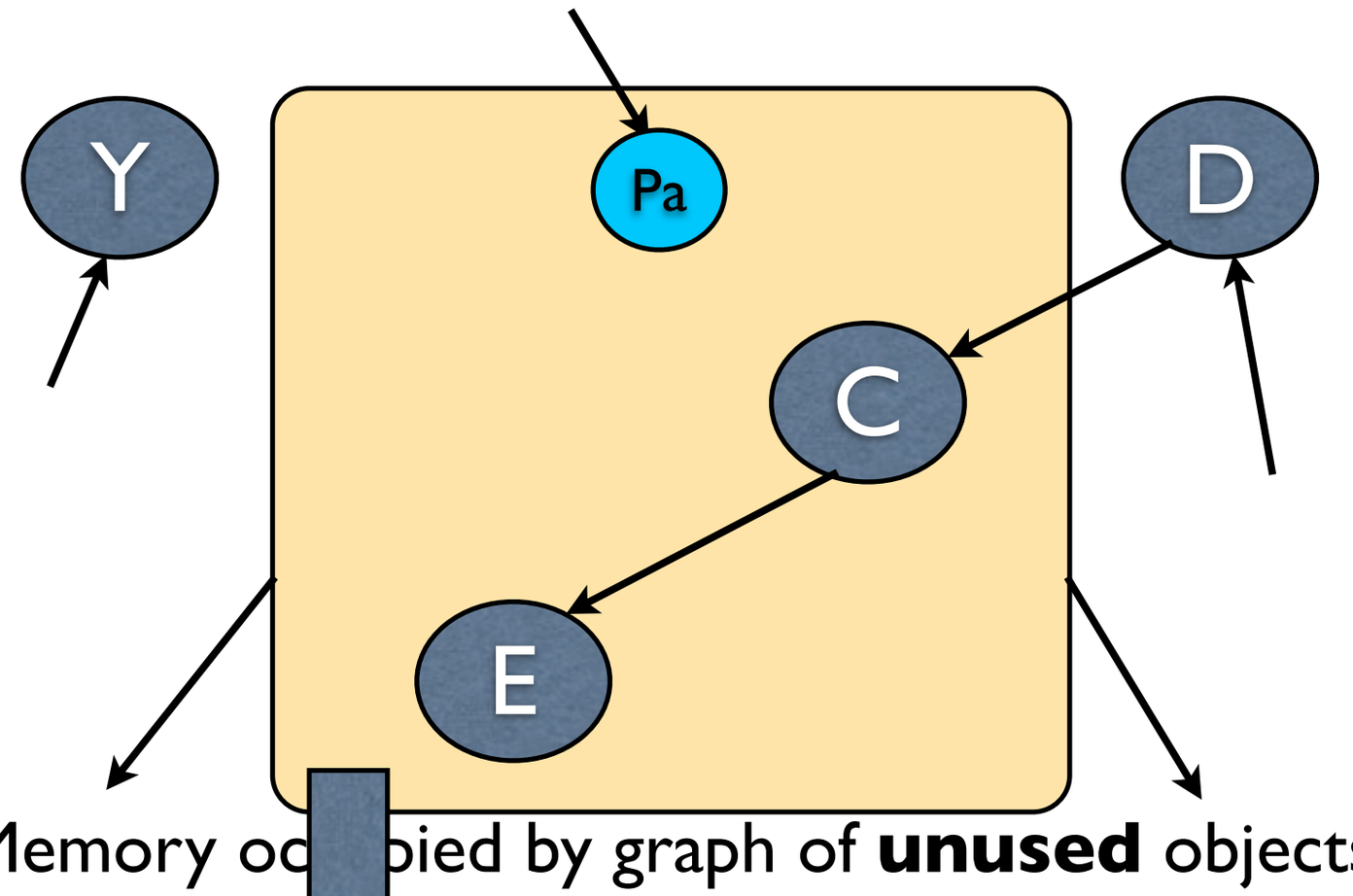


Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory

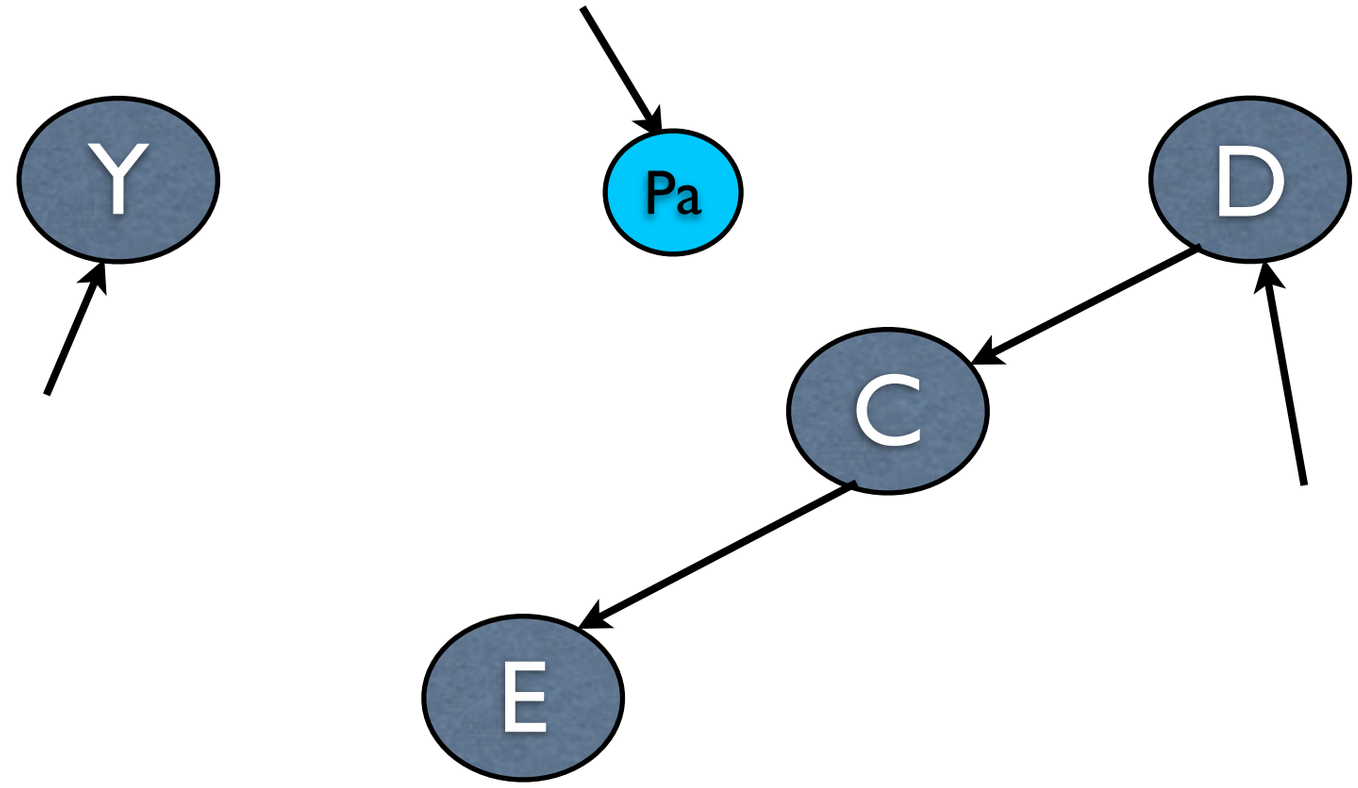


Secondary memory

Swap out

bytes (A,B,C,E,F)

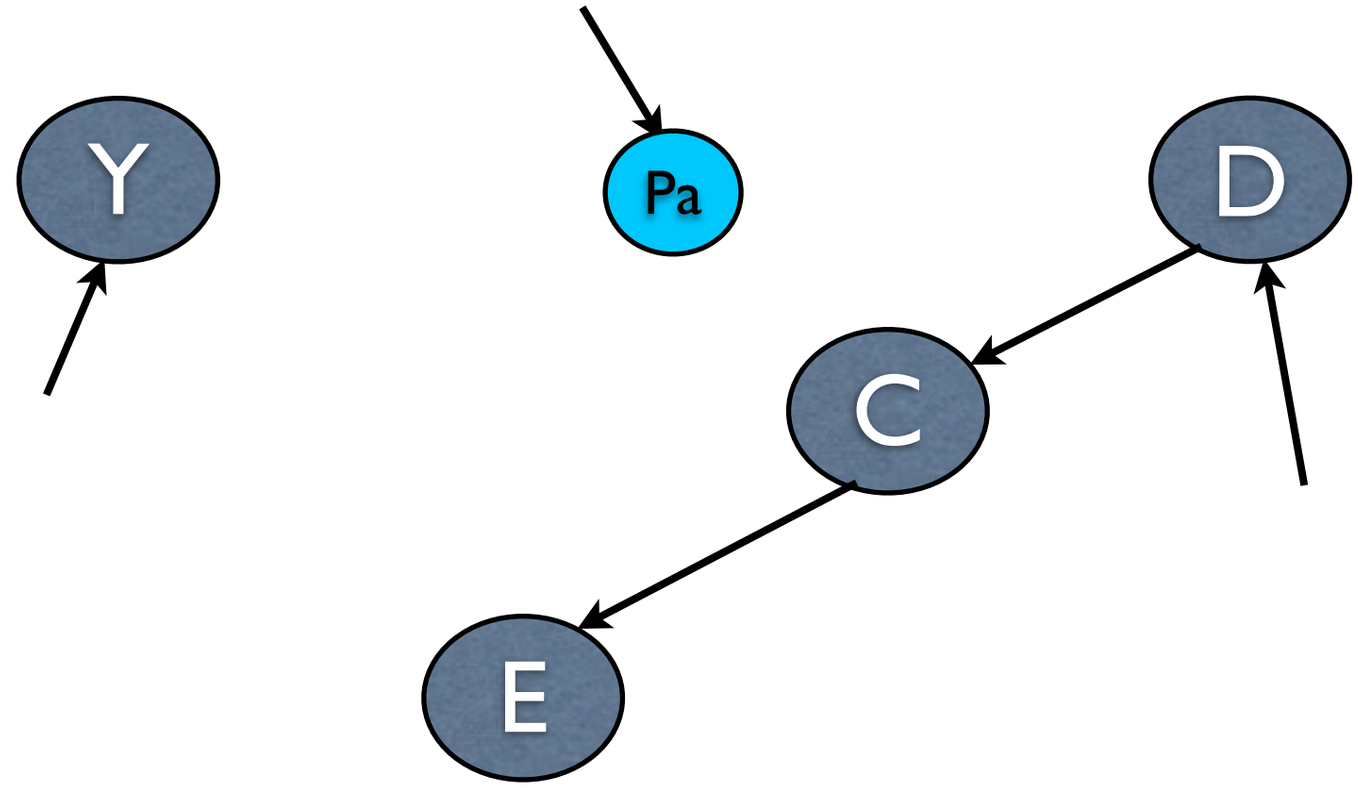
Primary memory



Secondary memory

bytes (A,B,C,E,F)

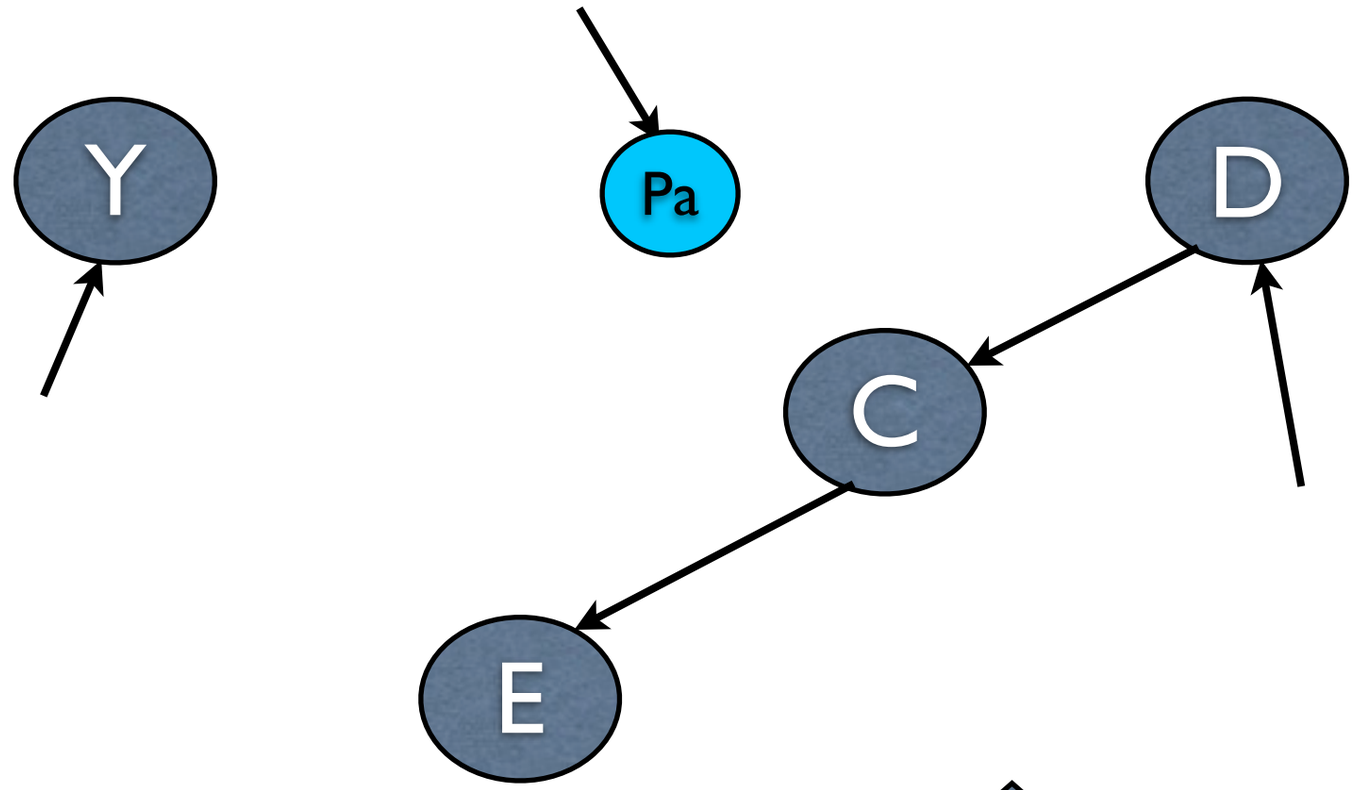
Primary memory



Secondary memory

bytes (A,B,C,E,F)

Primary memory

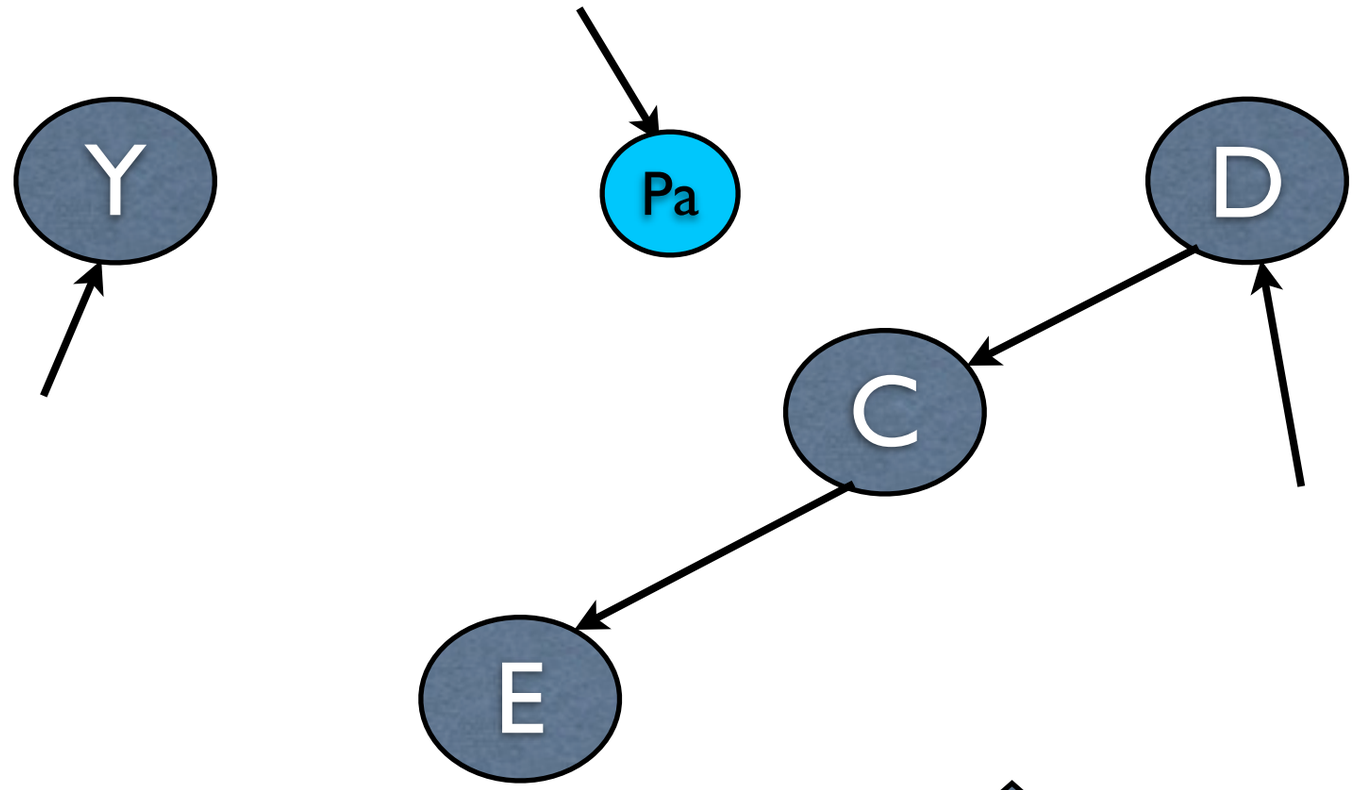


Secondary memory

Swap in

bytes (A,B,C,E,F)

Primary memory

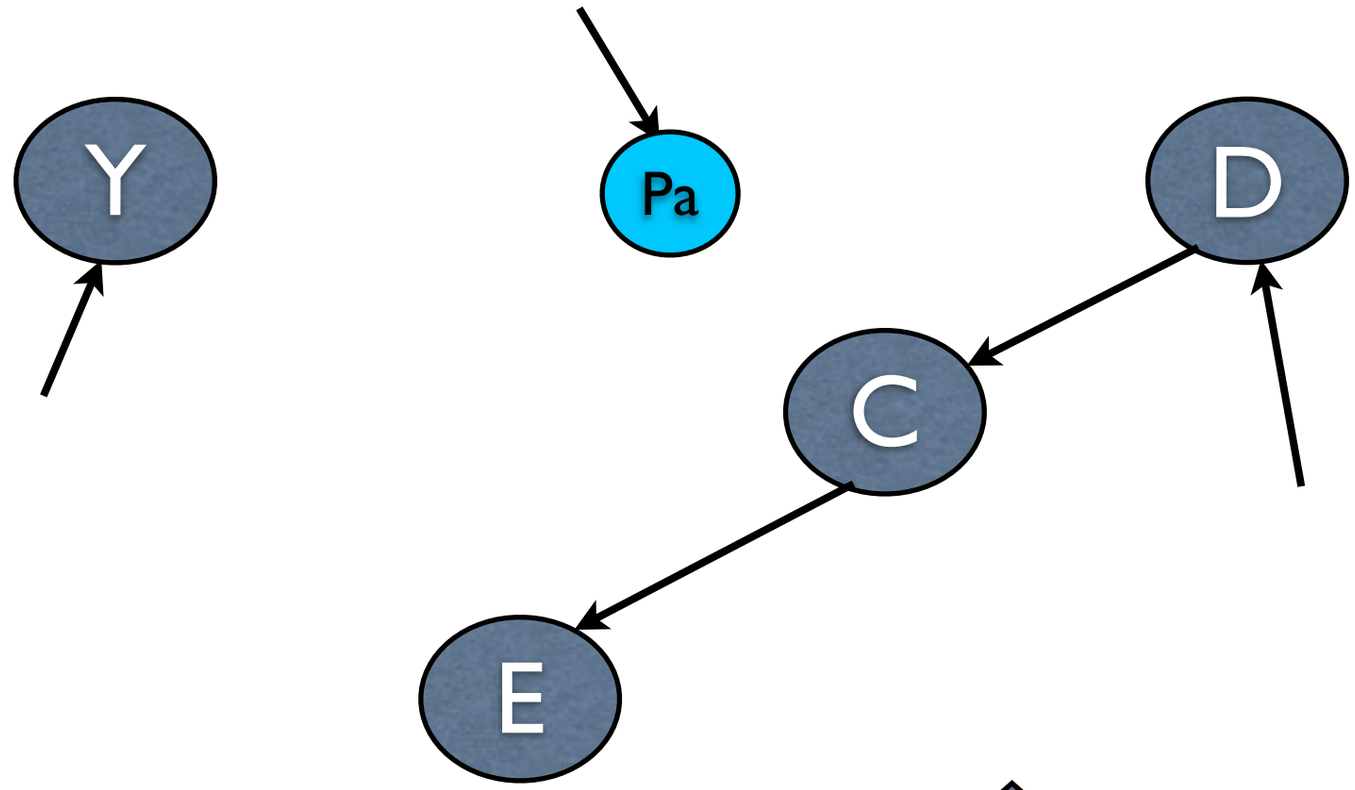


Swap in

Secondary memory

bytes (A,B,C,E,F)

Primary memory

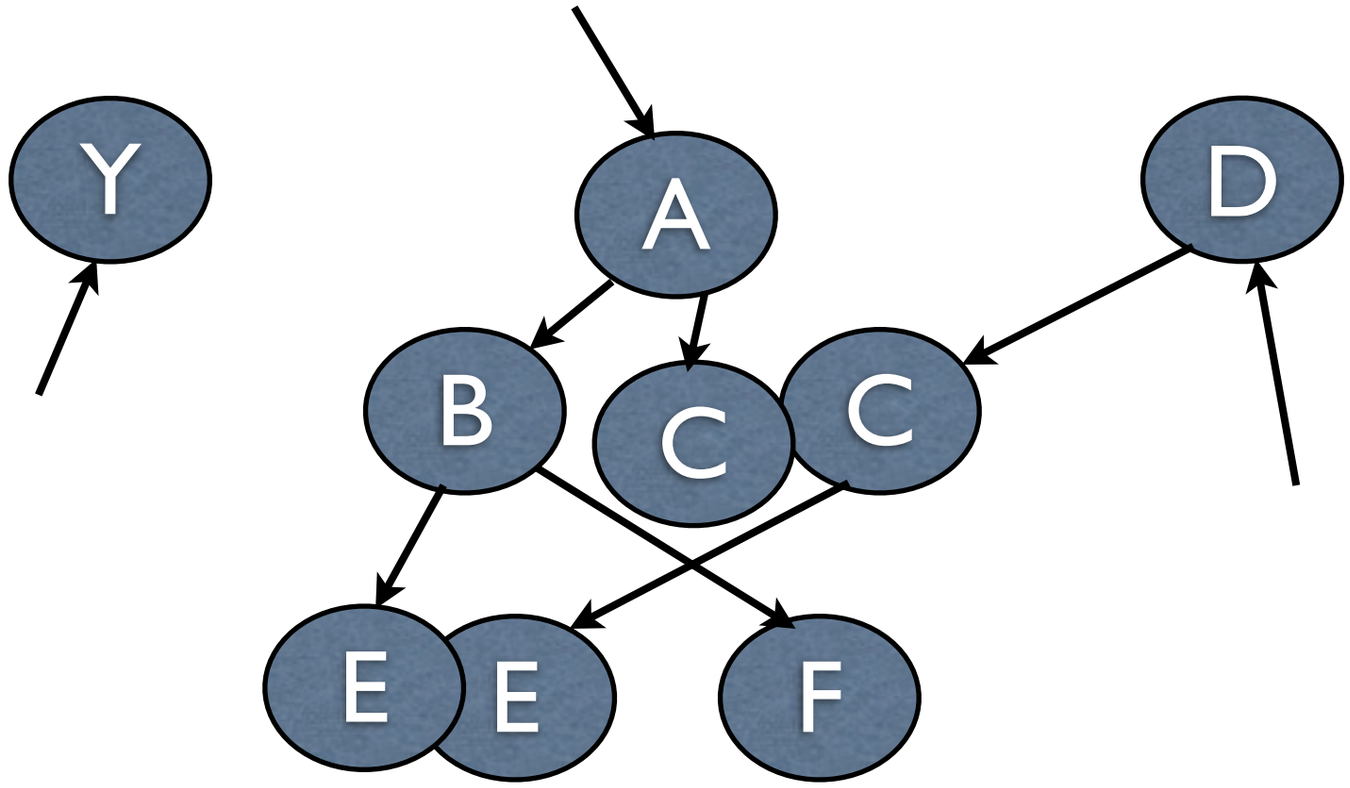


Secondary memory

Swap in

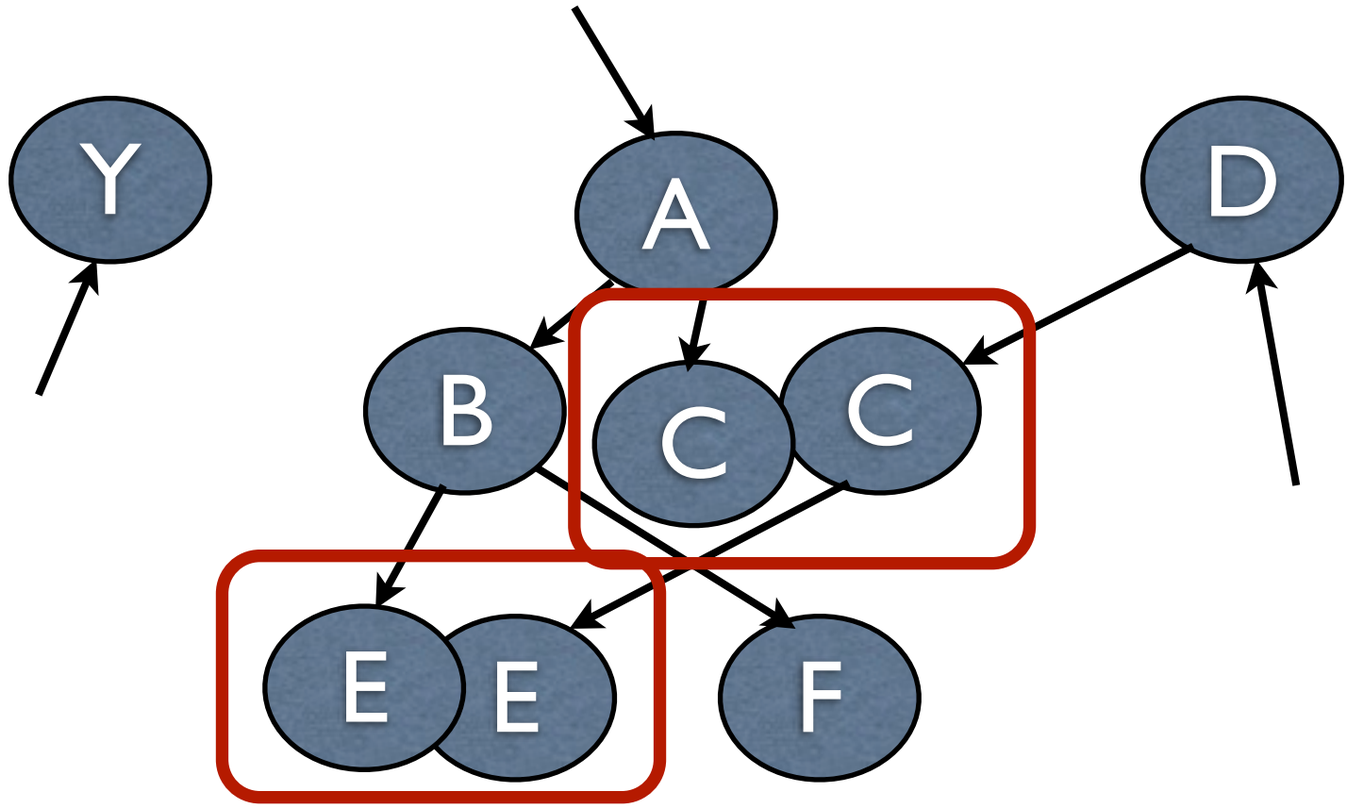
bytes (A,B,C,E,F)

Primary memory



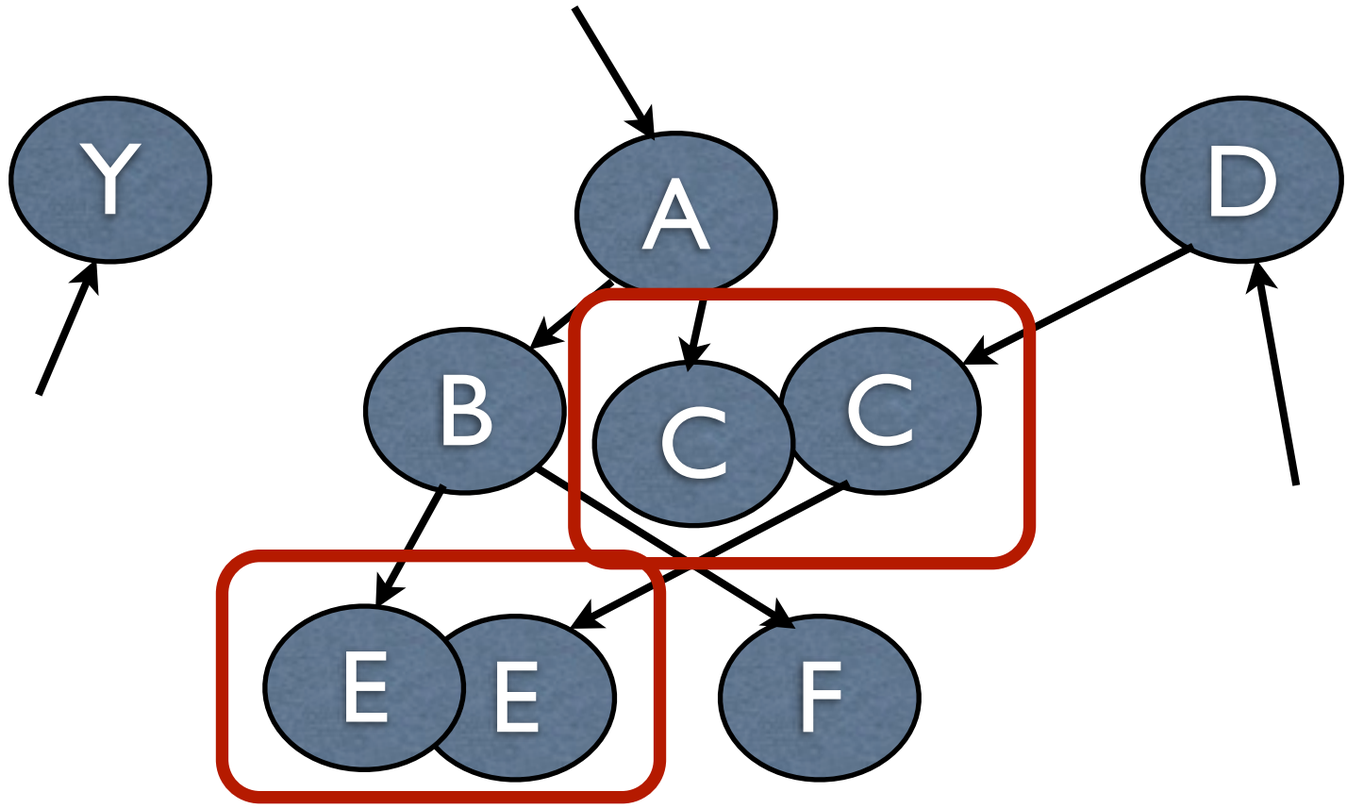
Secondary memory

Primary memory



Secondary memory

Primary memory



Secondary memory

INCORRECT!!!

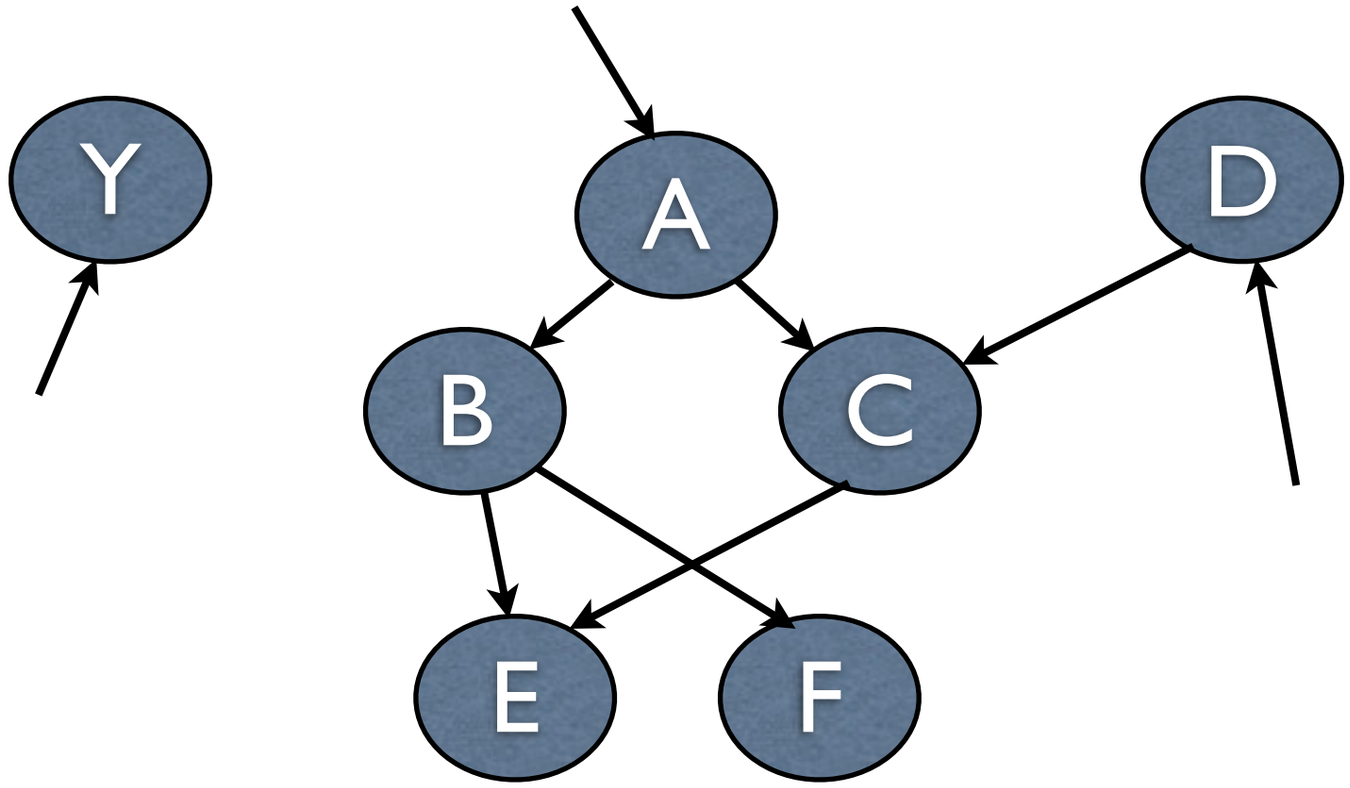
Objectives

- Correctness: *SwapIn(SwapOut(X))==X*
- Maximize memory released.
- Minimize runtime overhead.

Marea subsystems

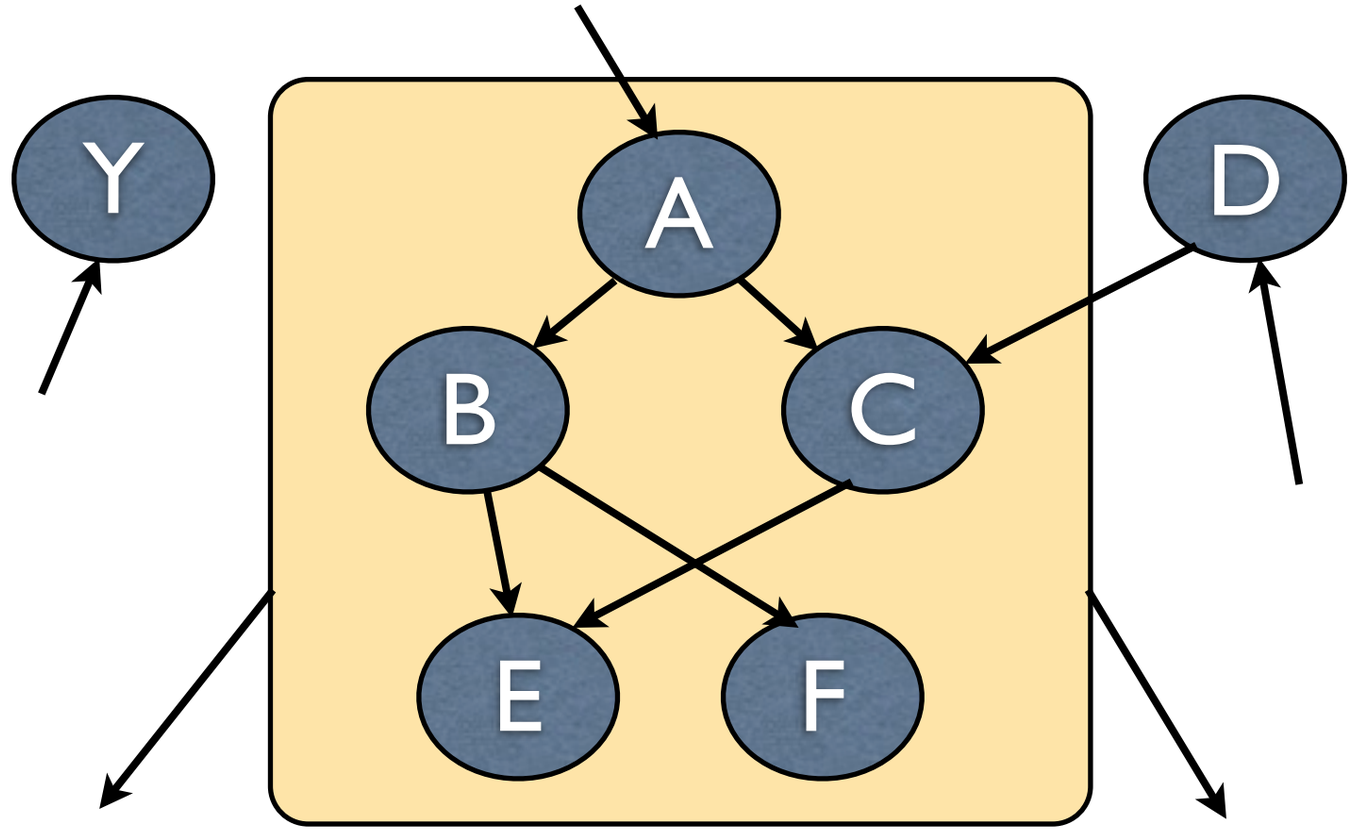
Object Graph Swapper	Proxy Toolbox
Object Graph Serializer	Object Graph Storage

Primary memory



Secondary memory

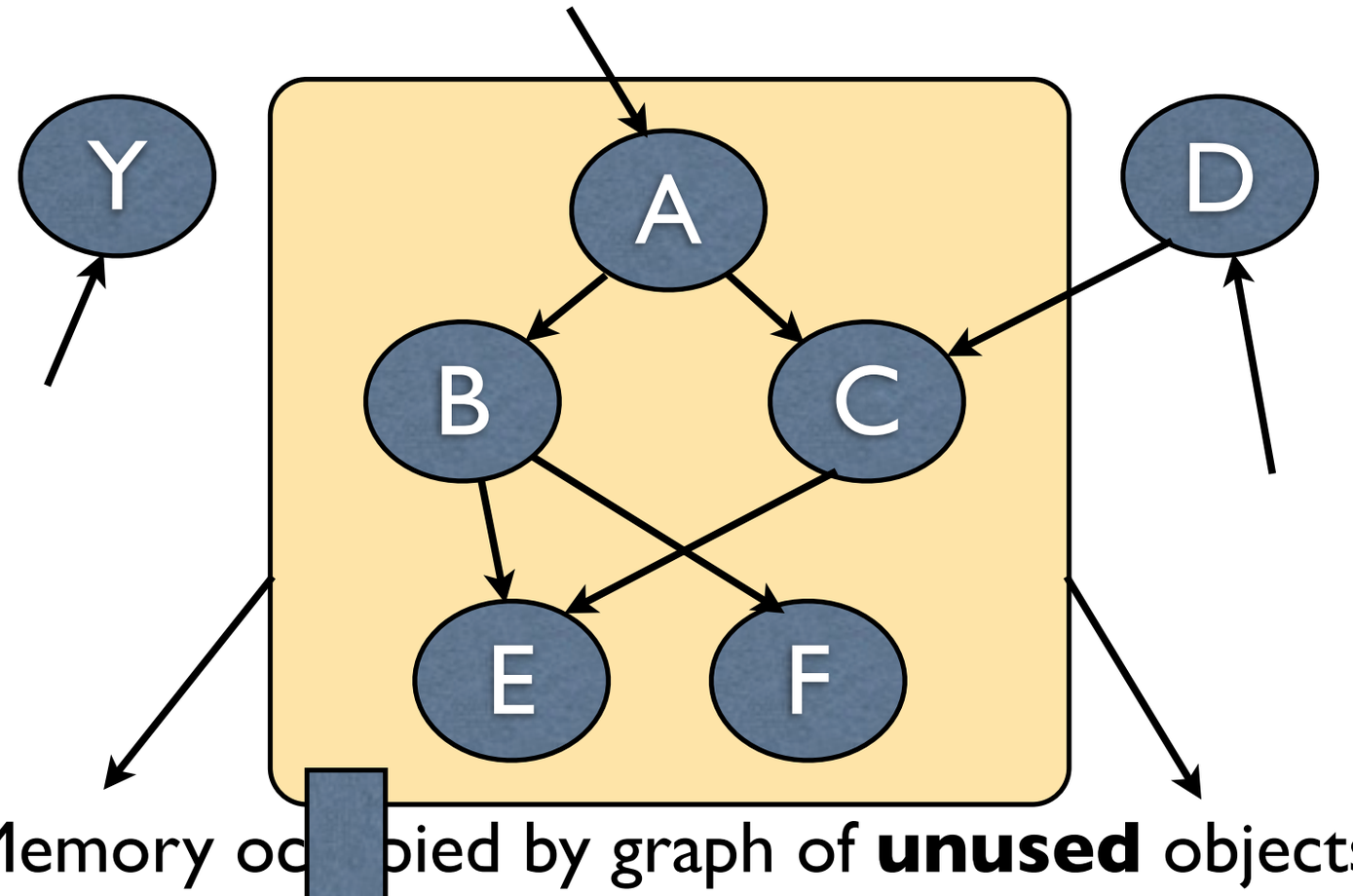
Primary memory



Memory occupied by graph of **unused** objects

Secondary memory

Primary memory



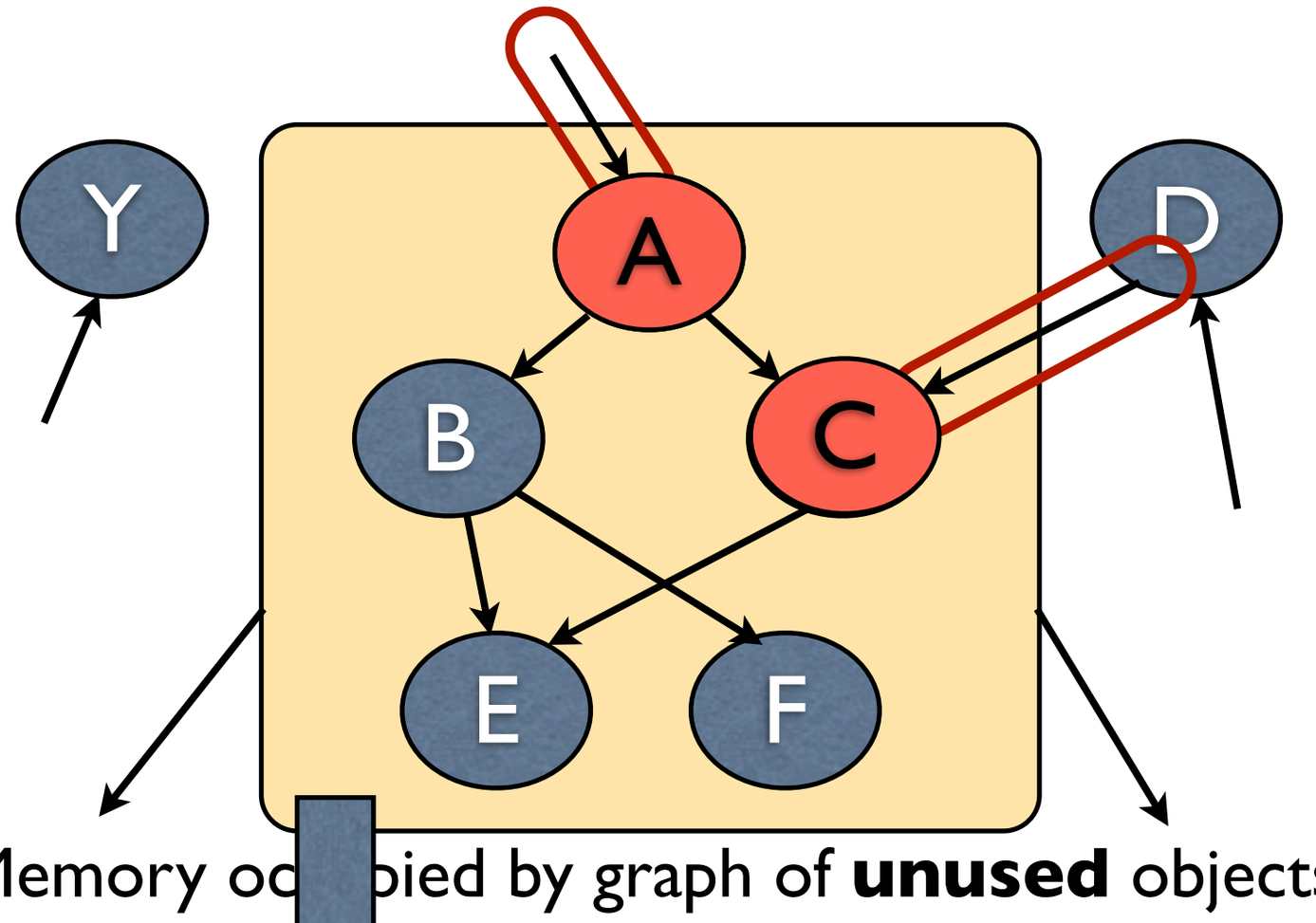
Memory occupied by graph of **unused** objects

Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory



facade objects

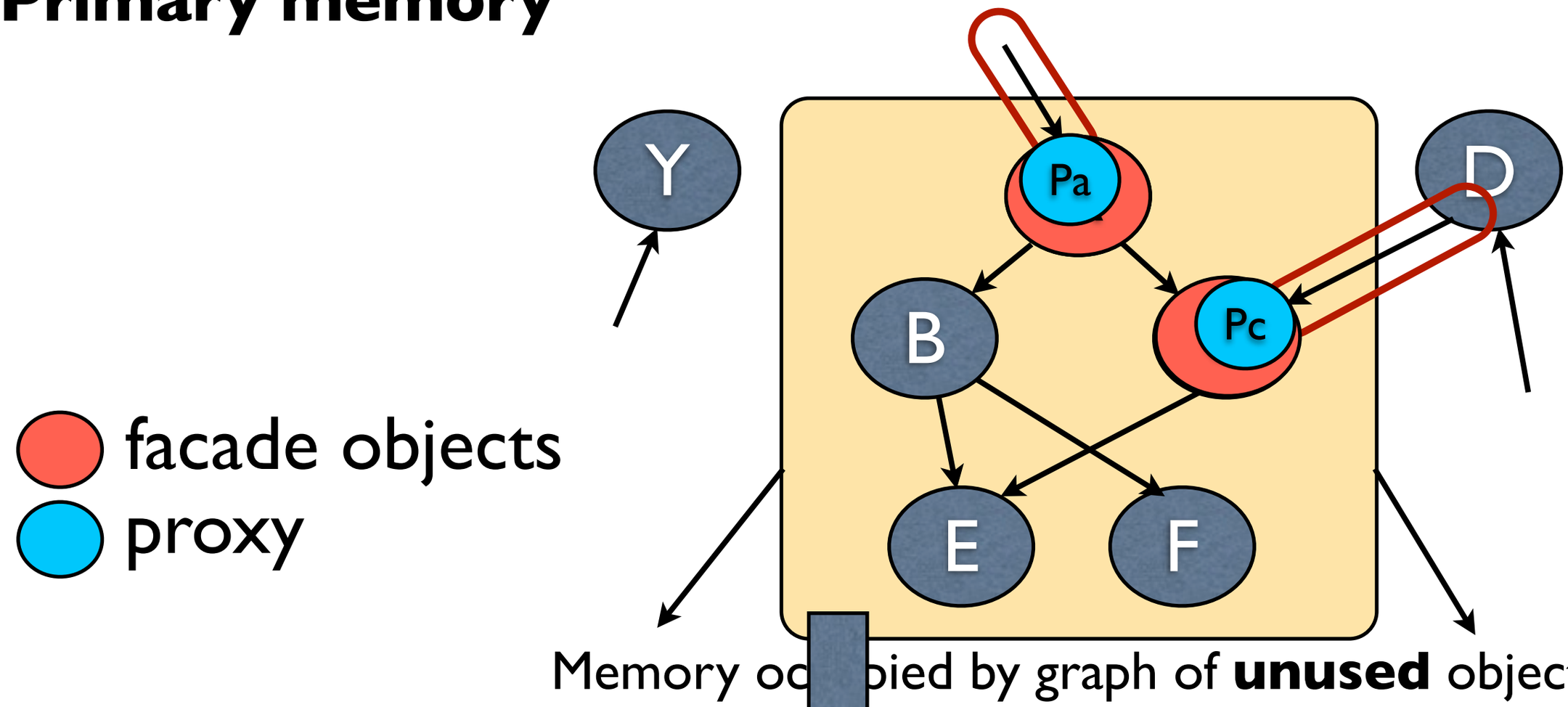
Memory occupied by graph of **unused** objects

Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory

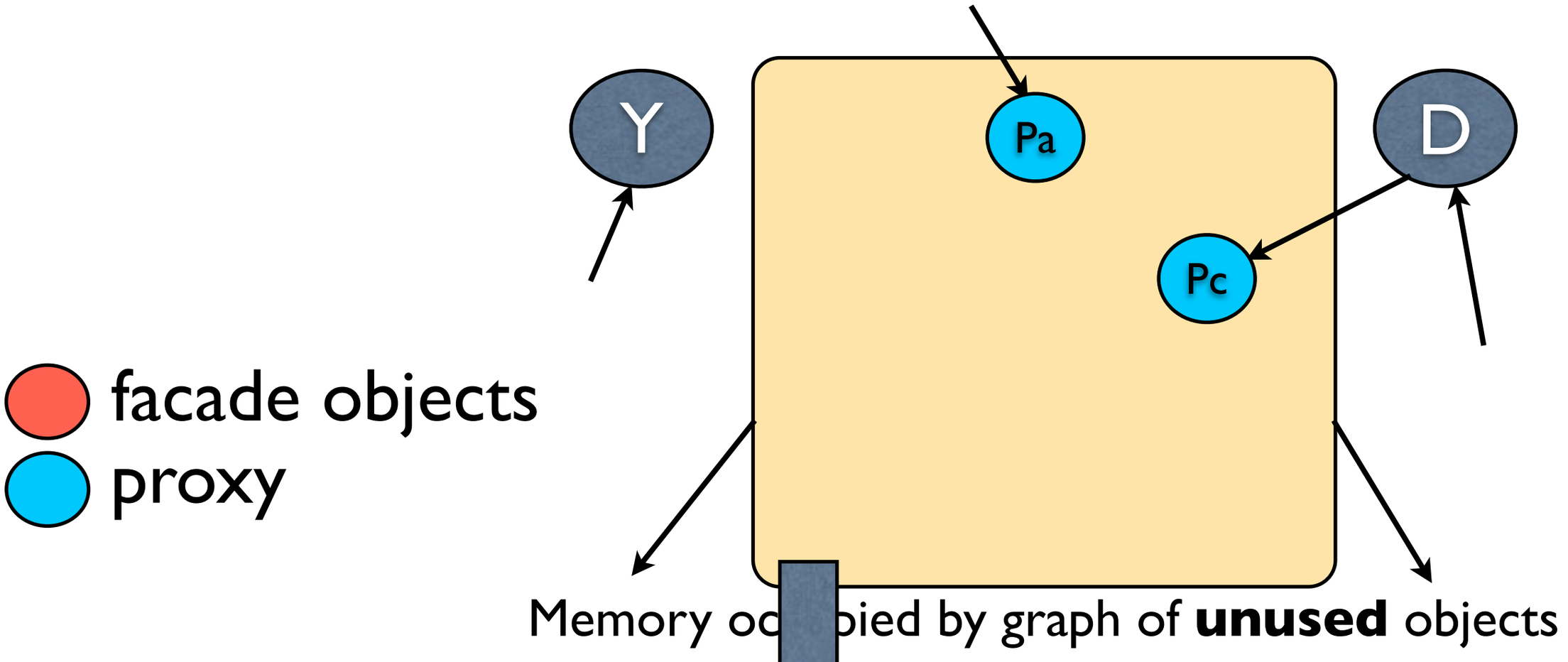


Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory

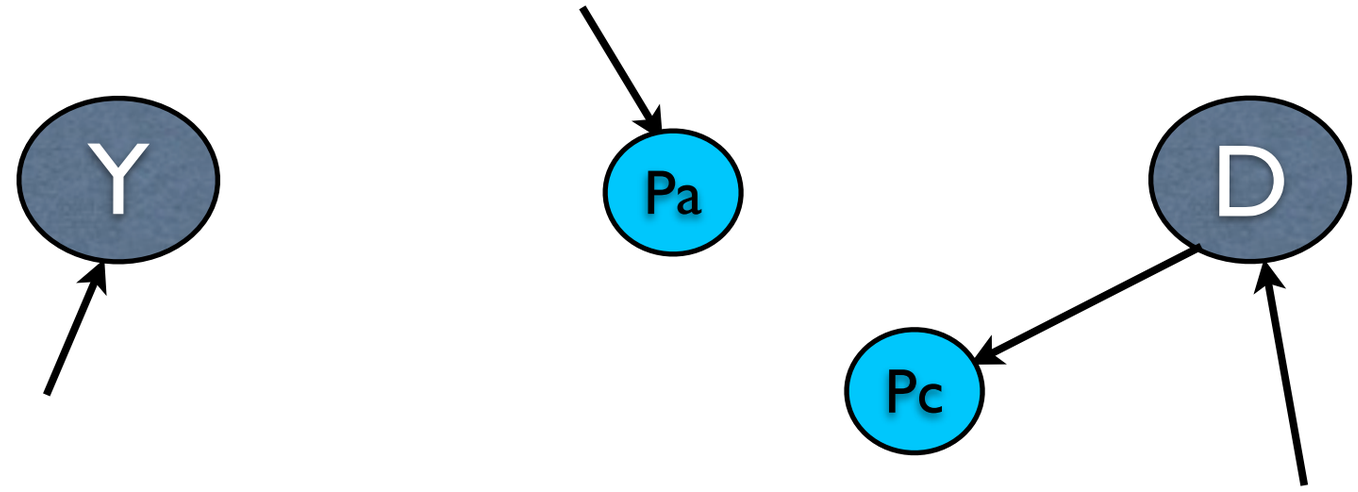


Secondary memory

Swap out

bytes (A,B,C,E,F)

Primary memory



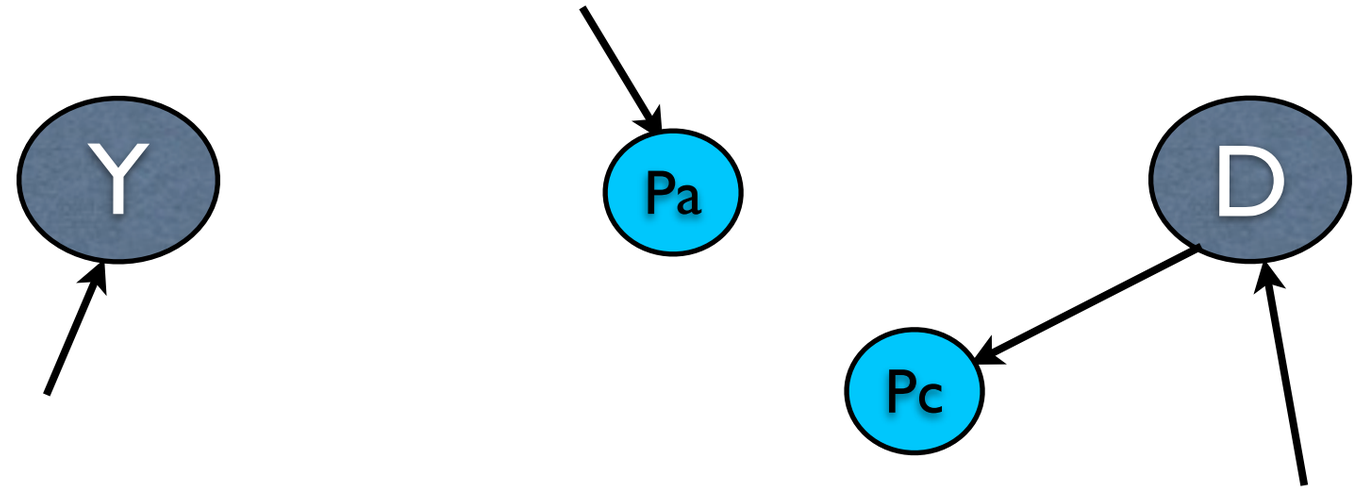
facade objects

proxy

Secondary memory

bytes (A,B,C,E,F)

Primary memory



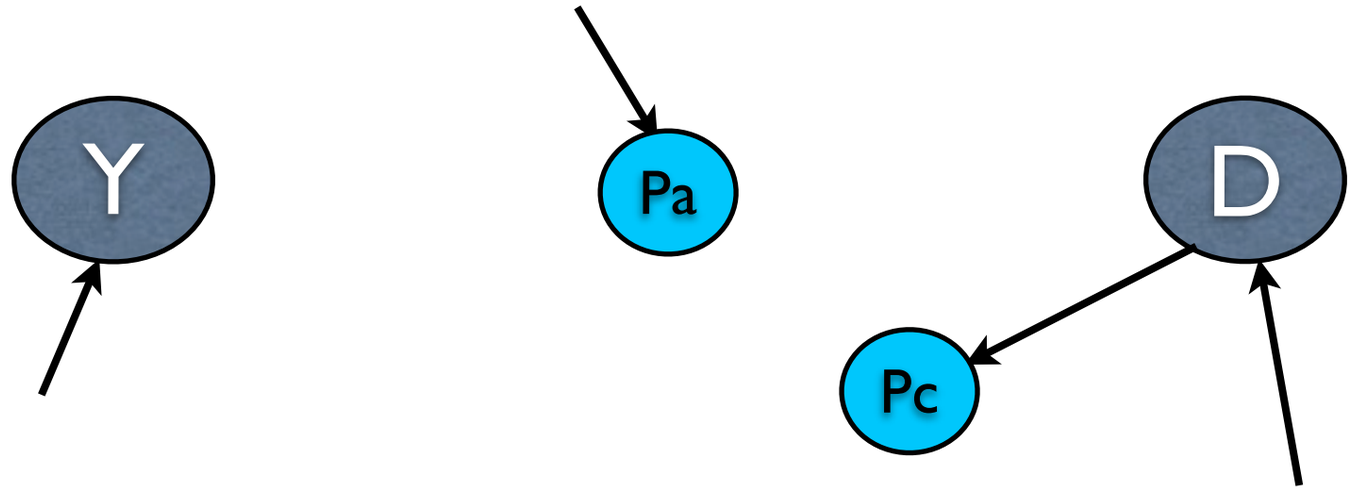
facade objects

proxy

Secondary memory

bytes (A,B,C,E,F)

Primary memory



facade objects

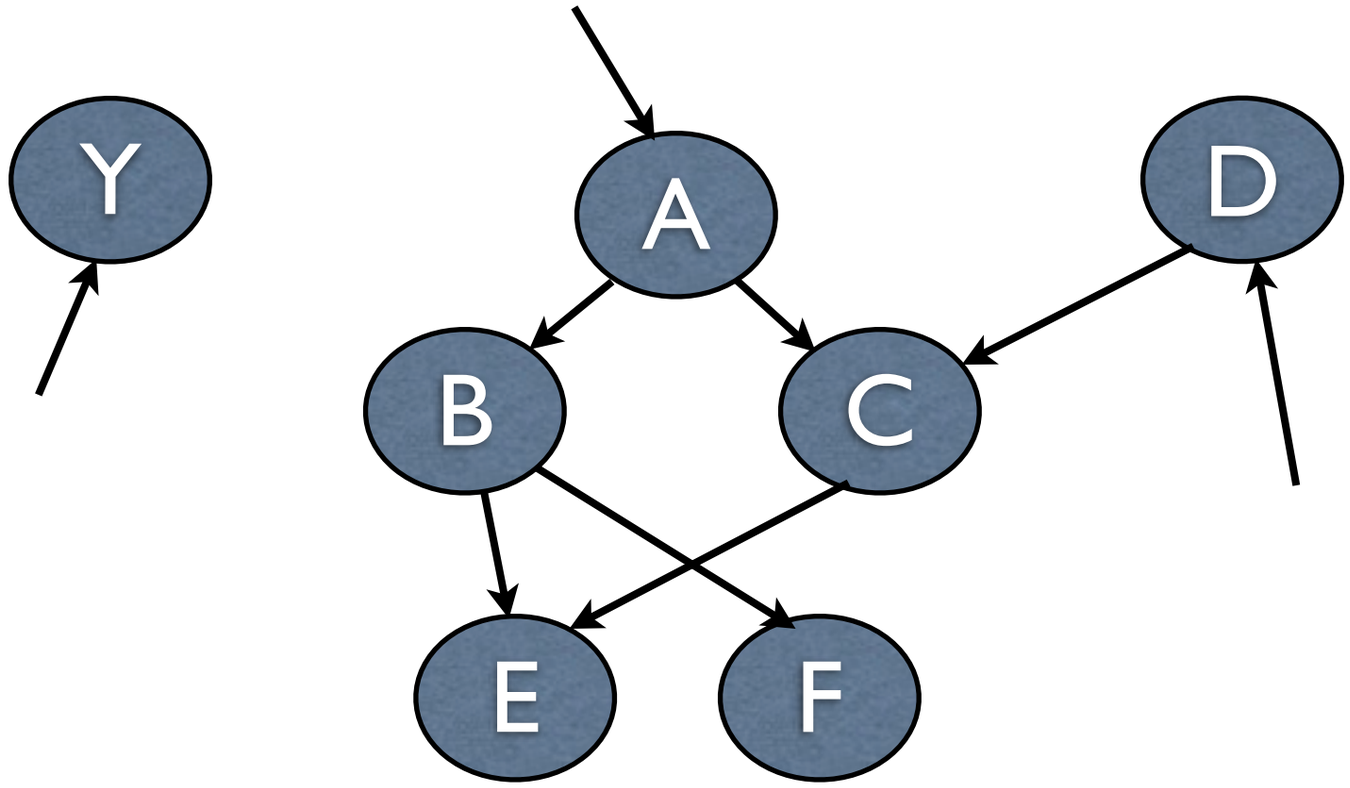
proxy

Secondary memory

Swap in

bytes (A,B,C,E,F)

Primary memory



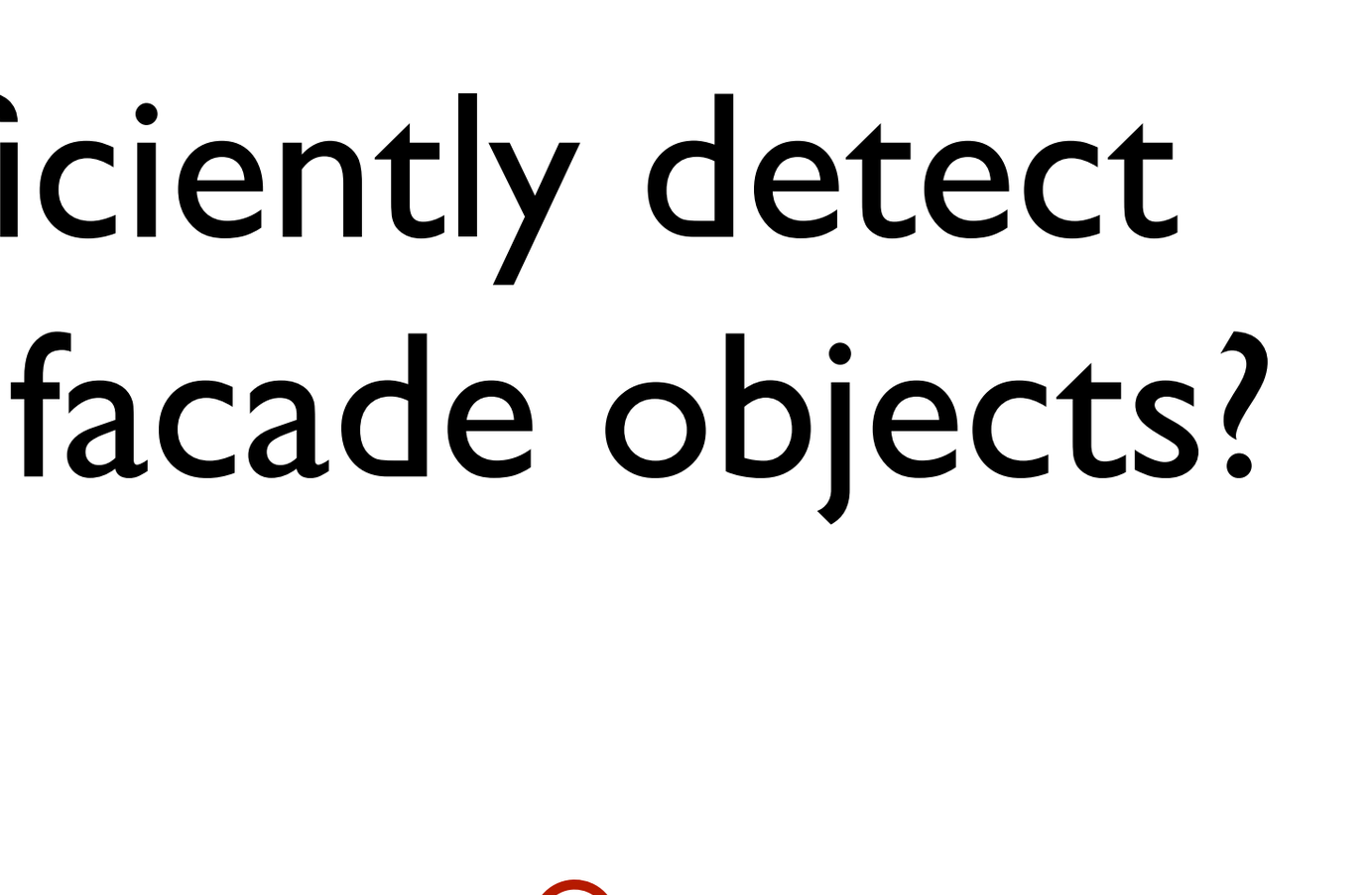
facade objects

proxy

Secondary memory

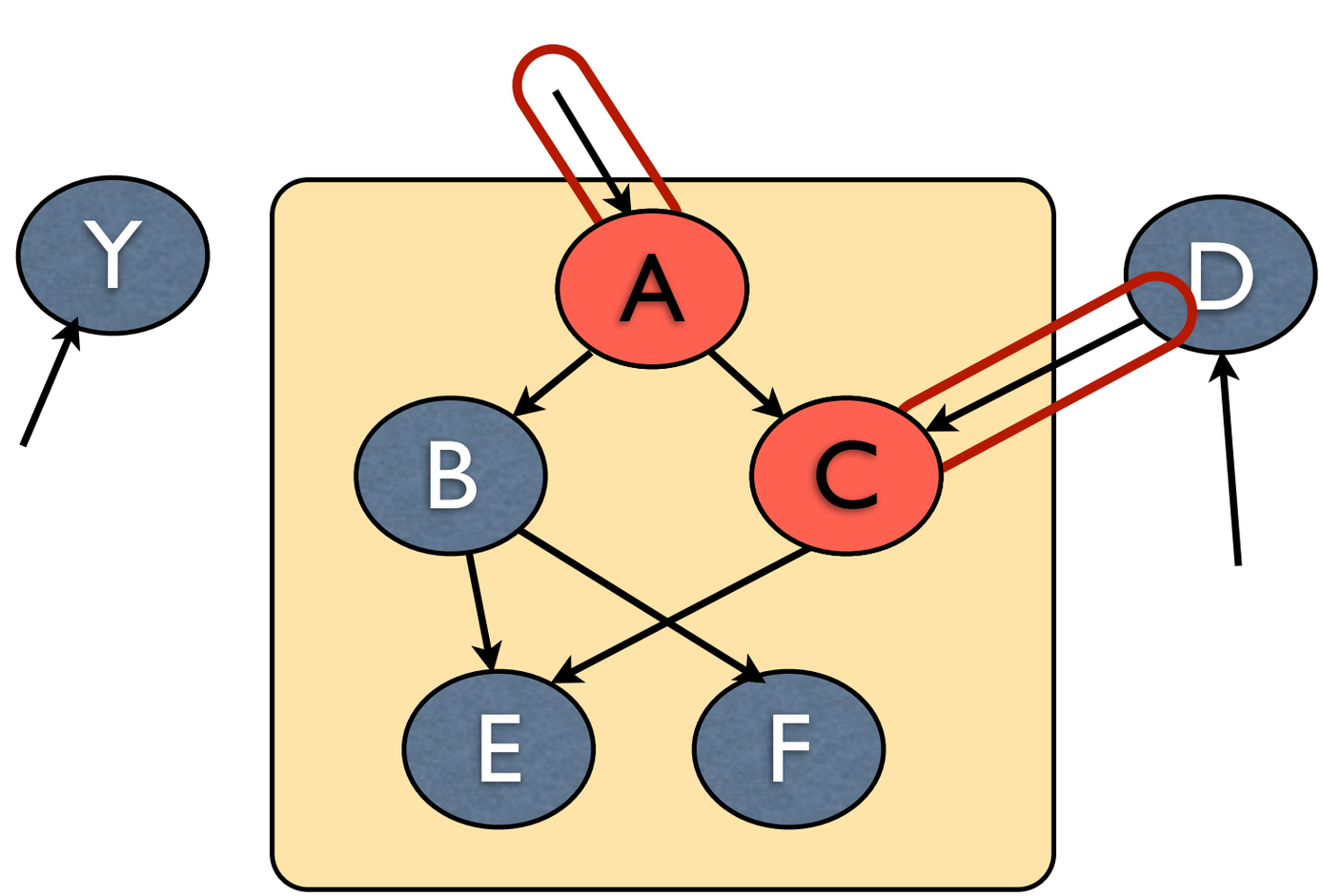
How to efficiently detect and manage facade objects?

- Back-pointers.
- Whole memory scan.



How to efficiently detect and manage facade objects?

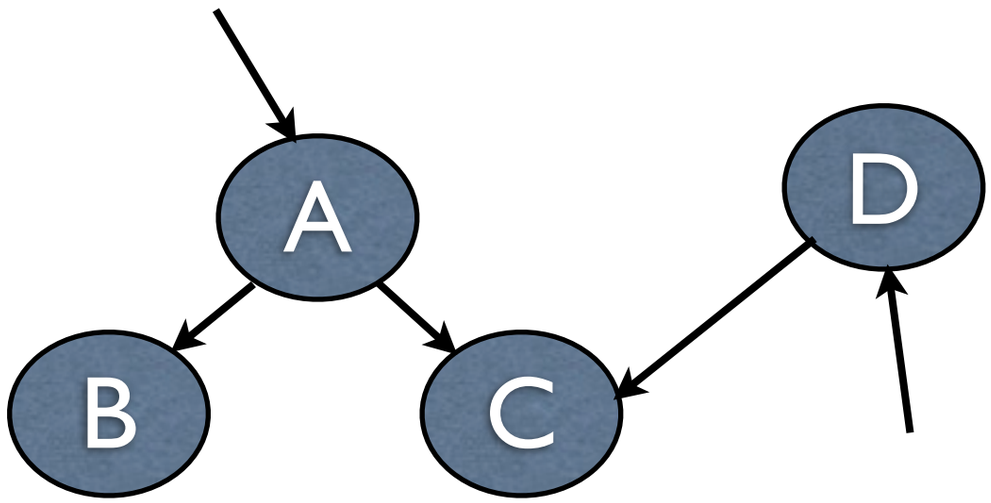
- Back-pointers.
- Whole memory scan.

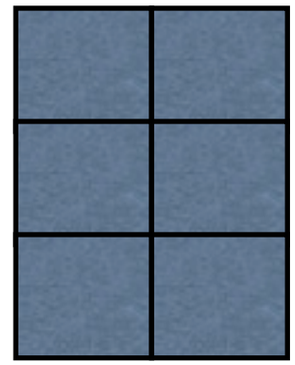


Marea provides an efficient solution.

Swapping out

Primary memory



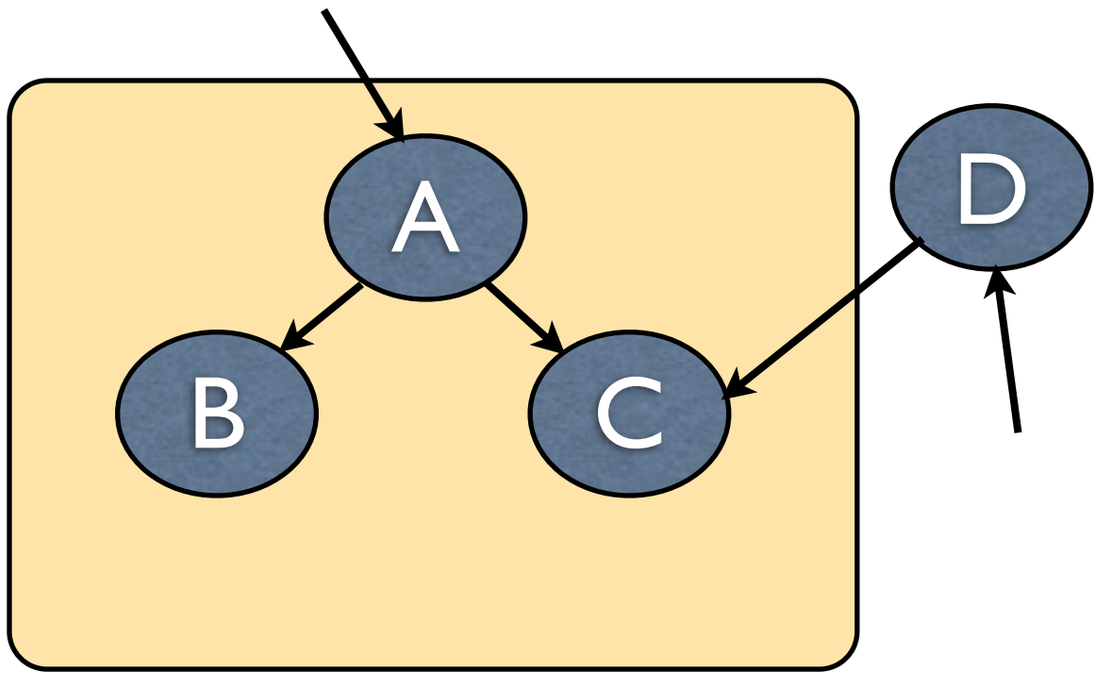


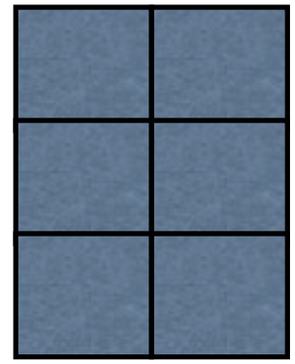
GraphTable

Secondary memory

Swapping out

Primary memory



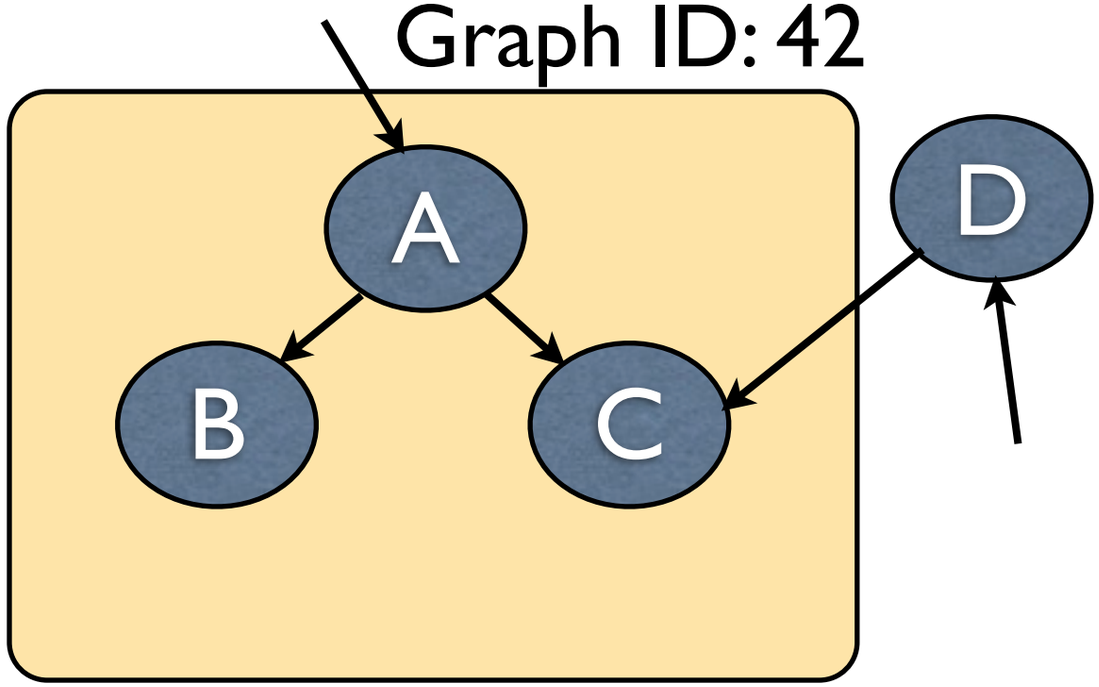


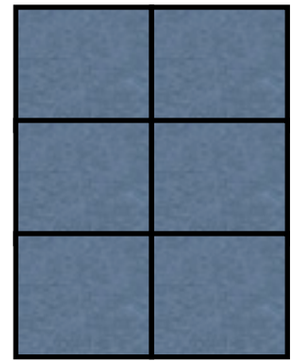
GraphTable

Secondary memory

Swapping out

Primary memory





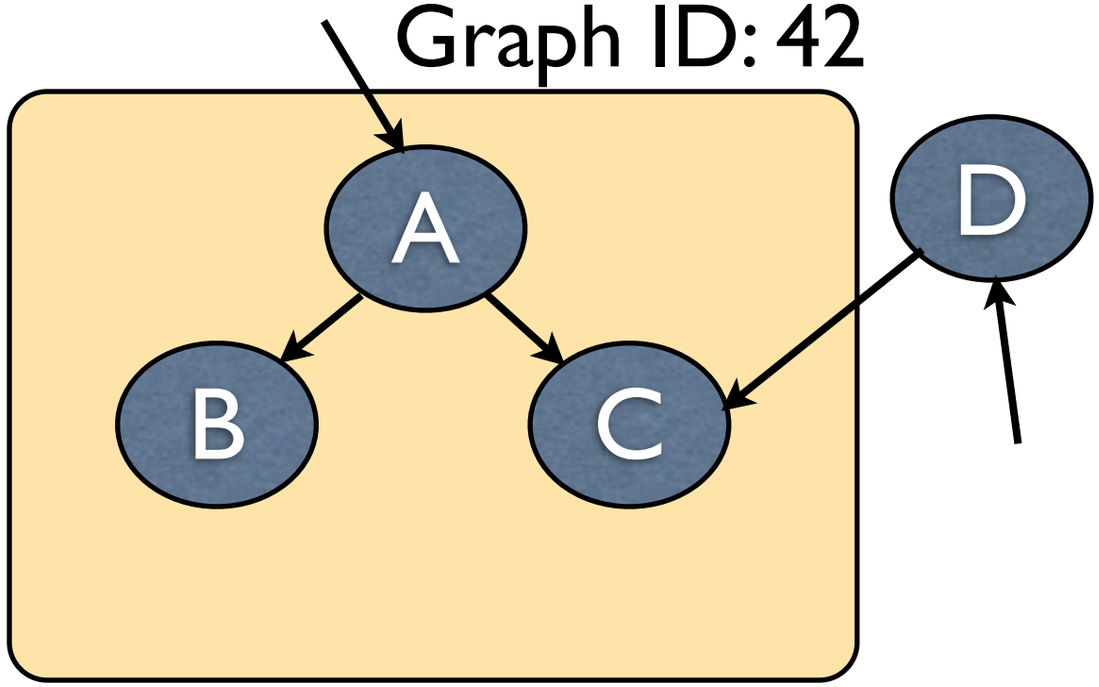
GraphTable

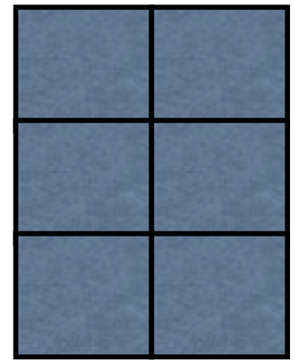
Secondary memory

1) Assign graph ID

Swapping out

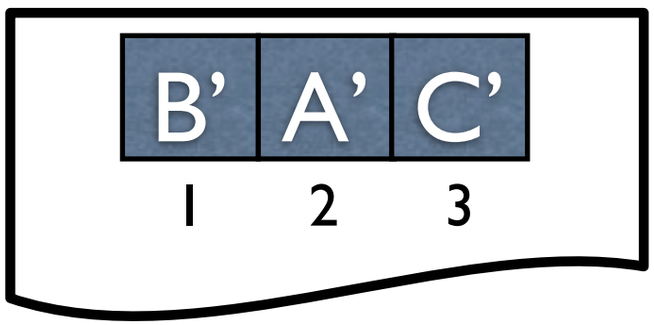
Primary memory





GraphTable

Secondary memory

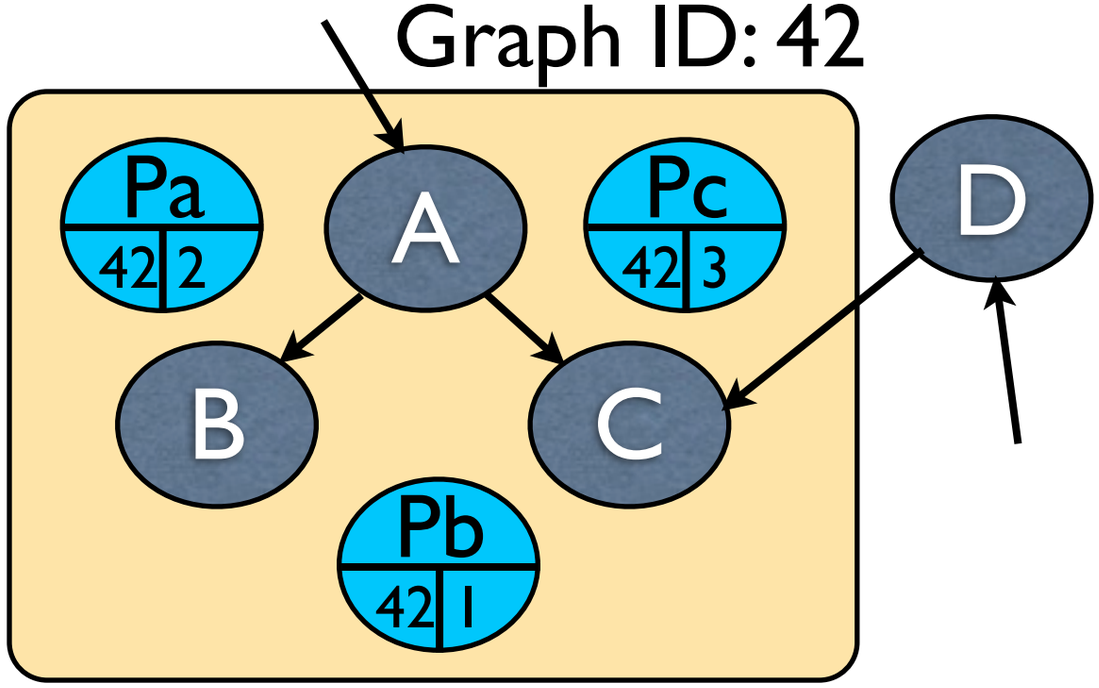


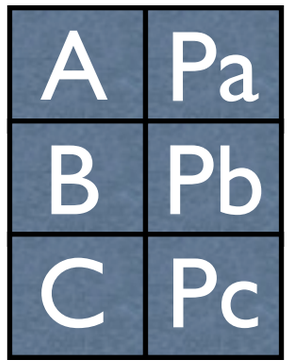
42.swap

1) Assign graph ID
2) Serialize the object graph

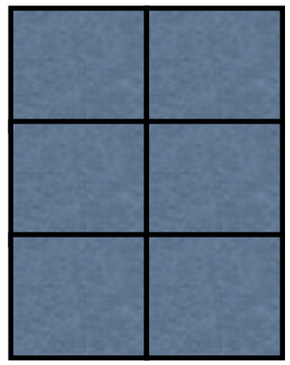
Swapping out

Primary memory



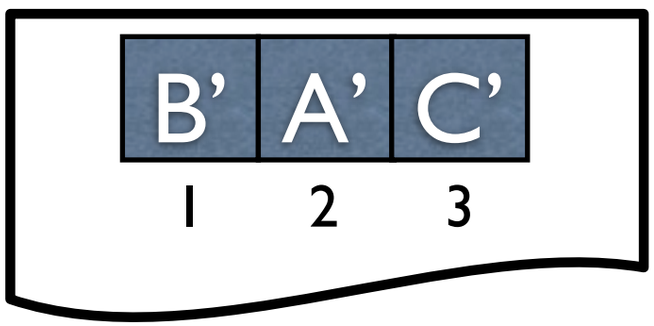


proxiesDict



GraphTable

Secondary memory

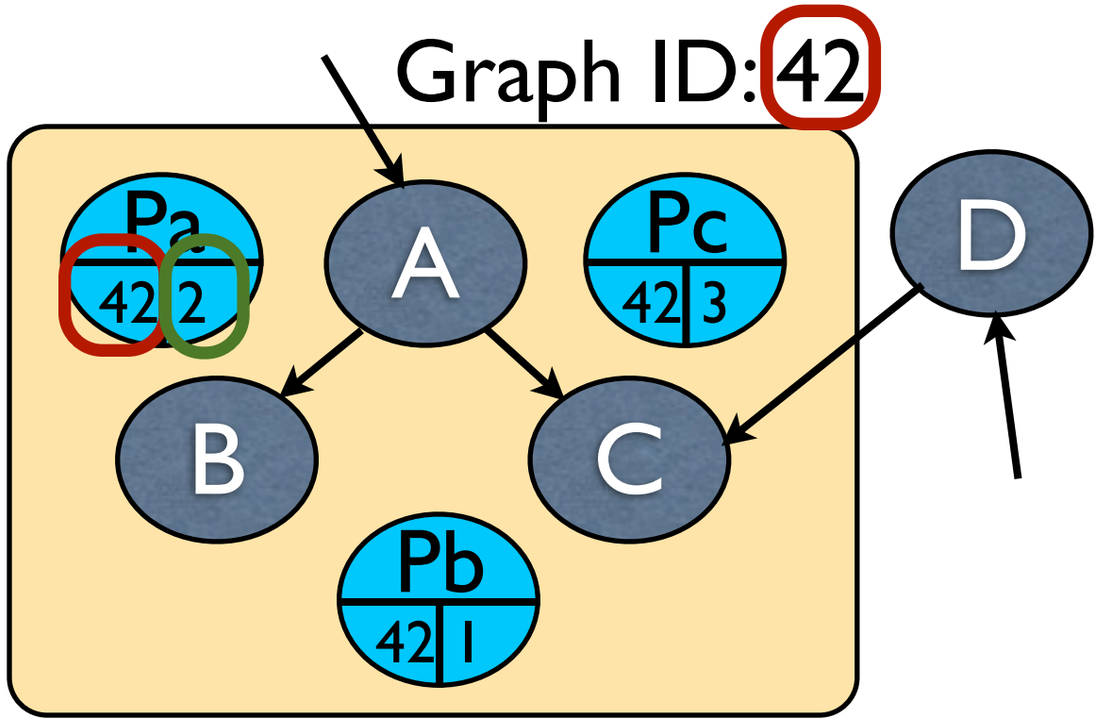


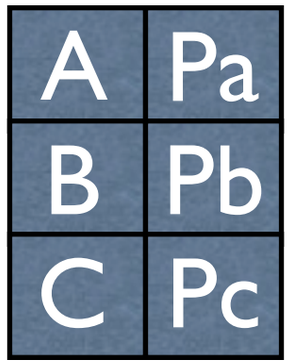
42.swap

1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies

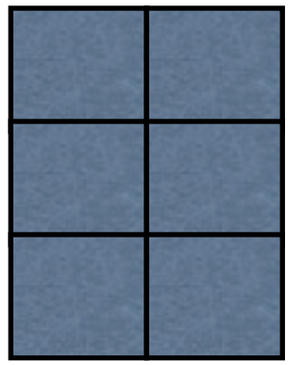
Swapping out

Primary memory





proxiesDict



GraphTable

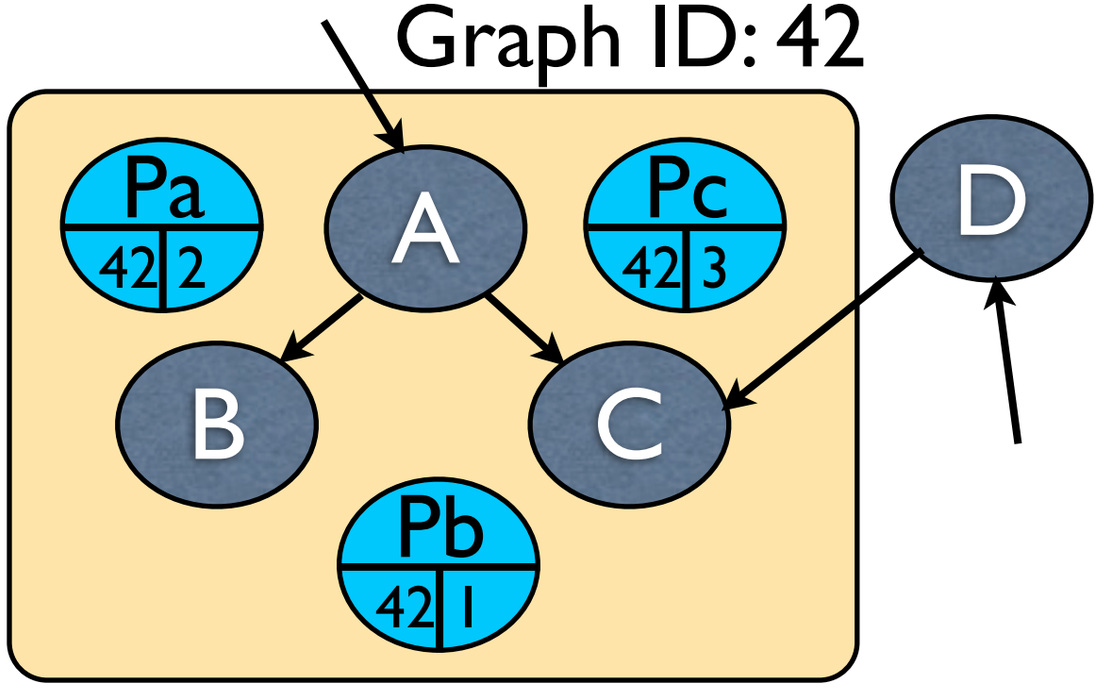
Secondary memory

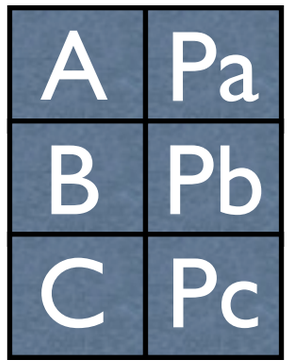
42.swap

1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies

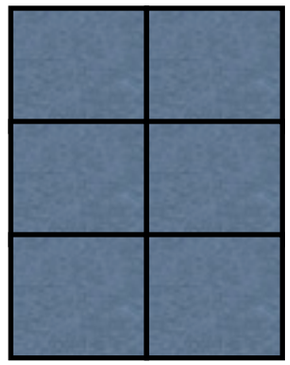
Swapping out

Primary memory



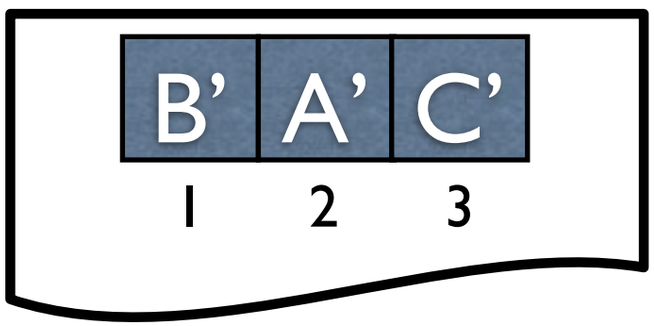


proxiesDict



GraphTable

Secondary memory

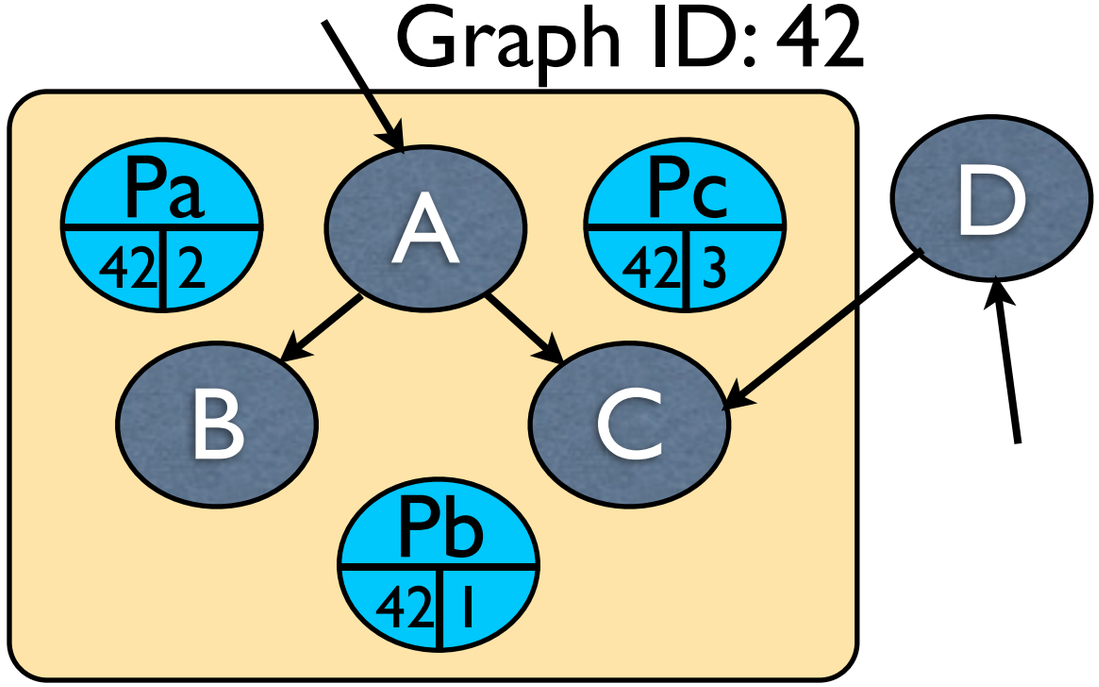


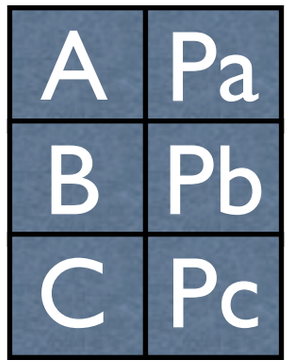
42.swap

1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies

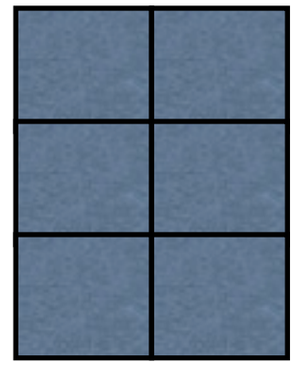
Swapping out

Primary memory



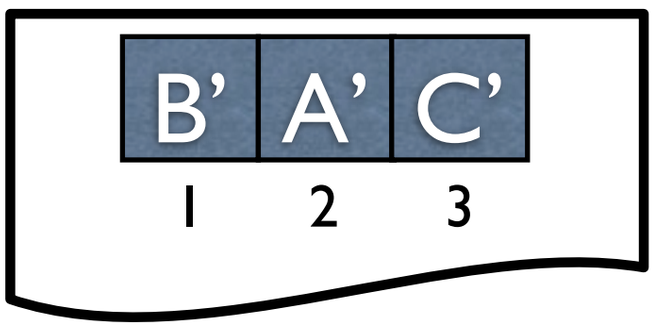


proxiesDict



GraphTable

Secondary memory

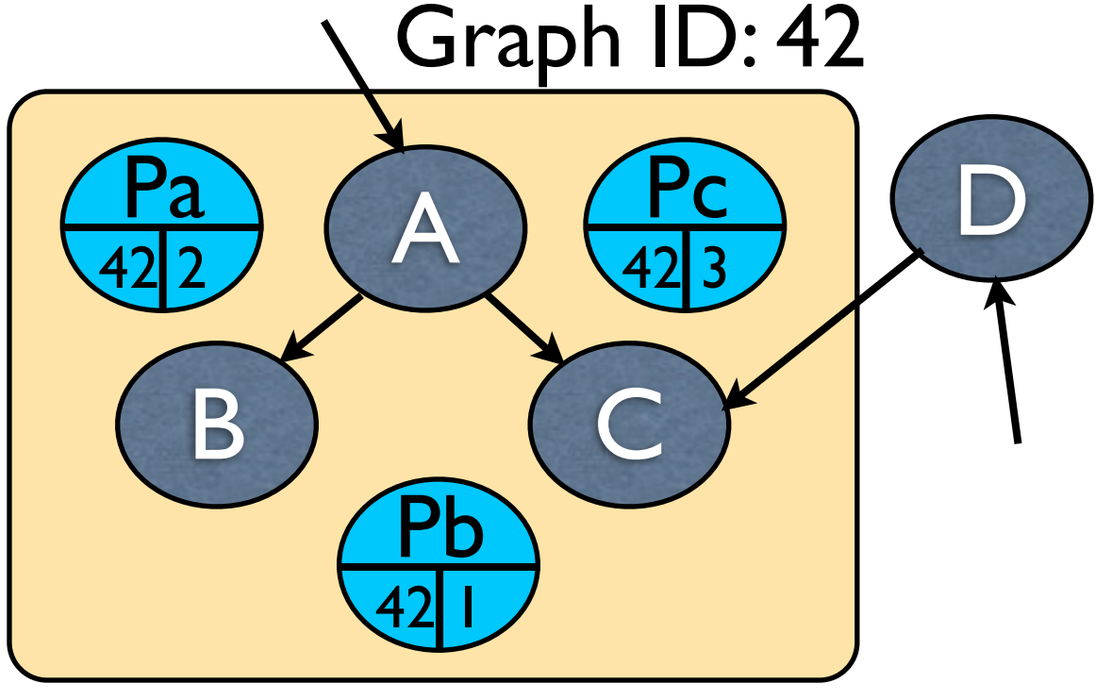


42.swap

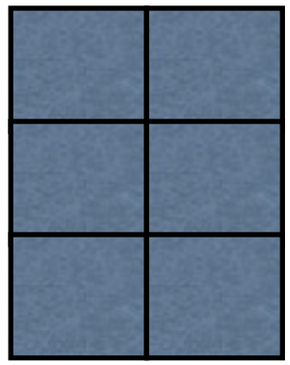
1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies

Swapping out

Primary memory

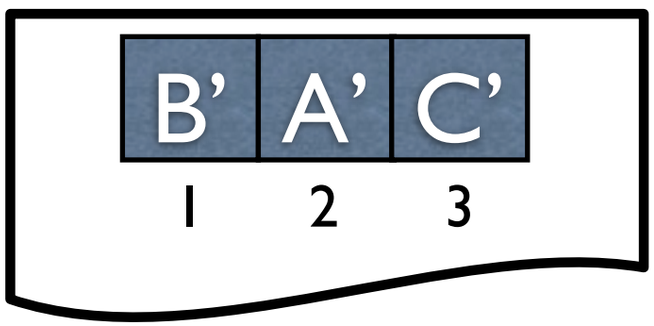


proxiesDict



GraphTable

Secondary memory

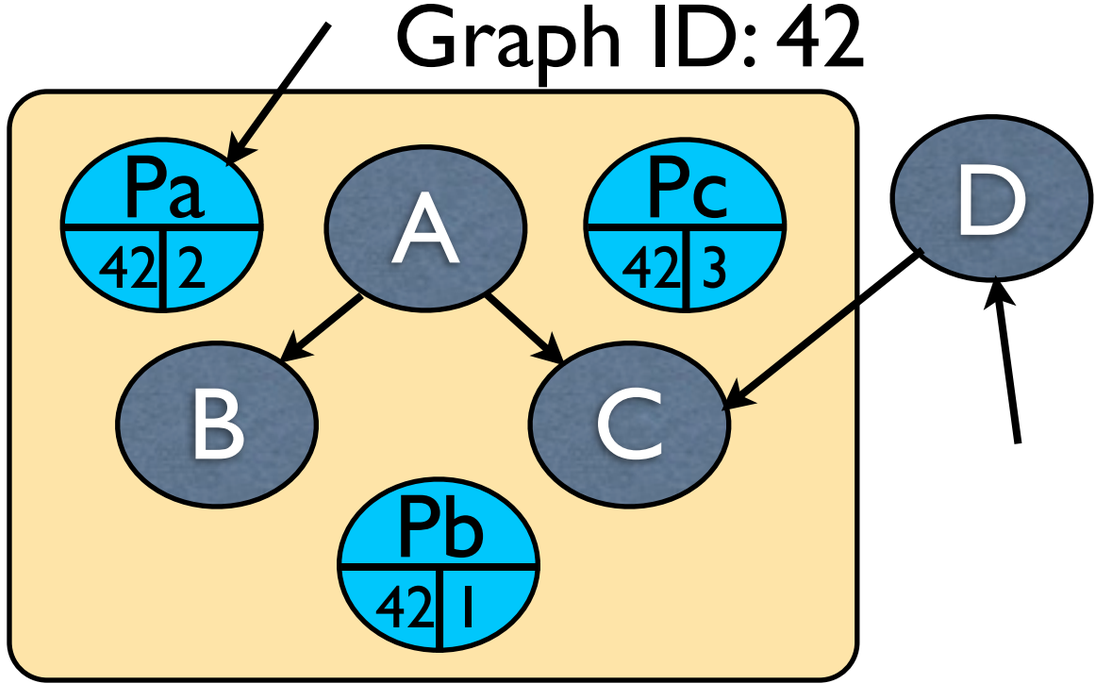


42.swap

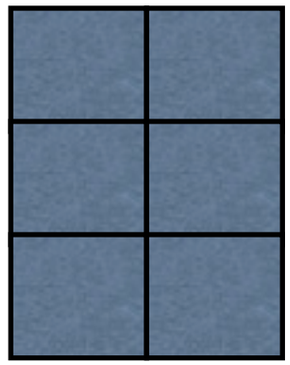
1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies

Swapping out

Primary memory

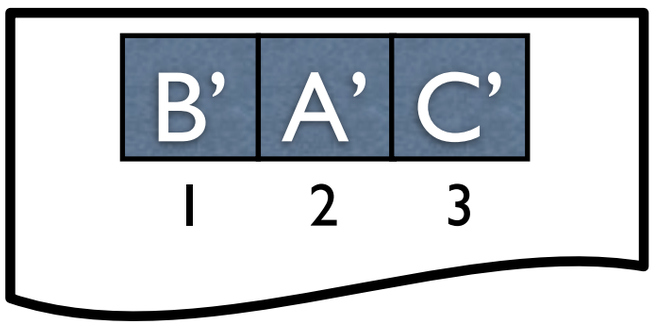


proxiesDict



GraphTable

Secondary memory

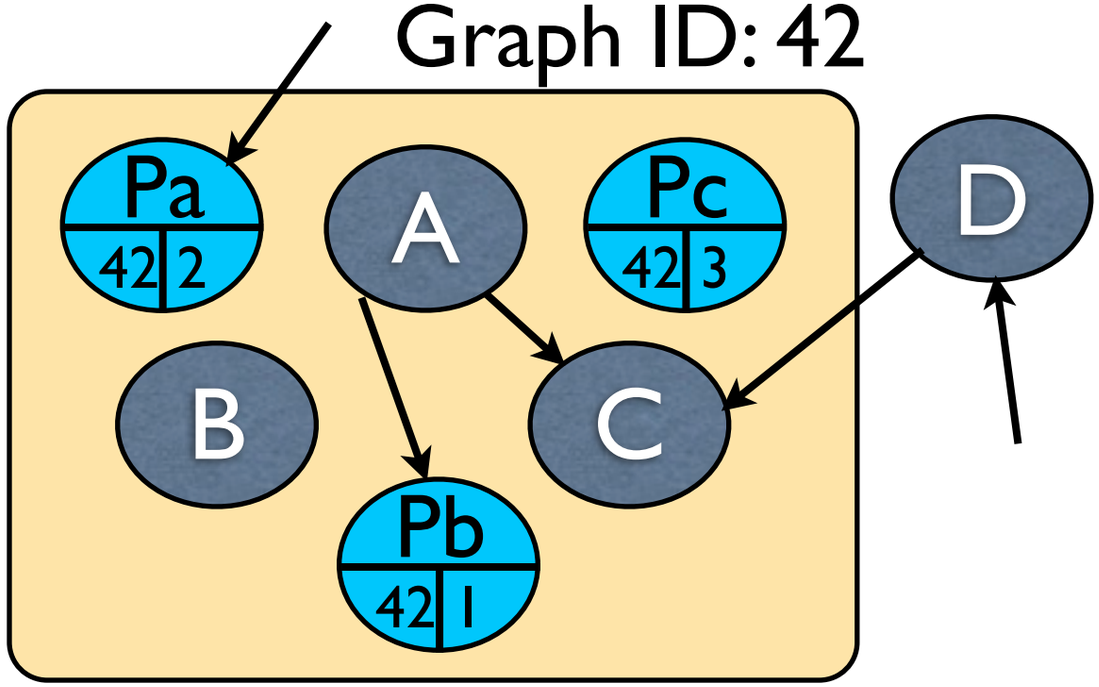


42.swap

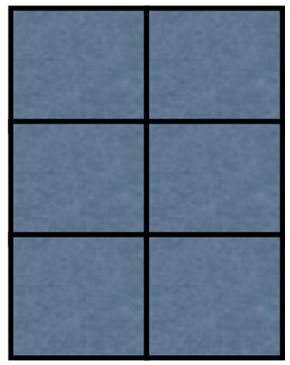
1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies

Swapping out

Primary memory

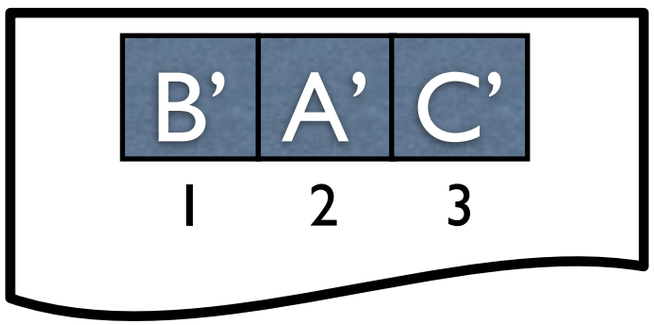


proxiesDict



GraphTable

Secondary memory

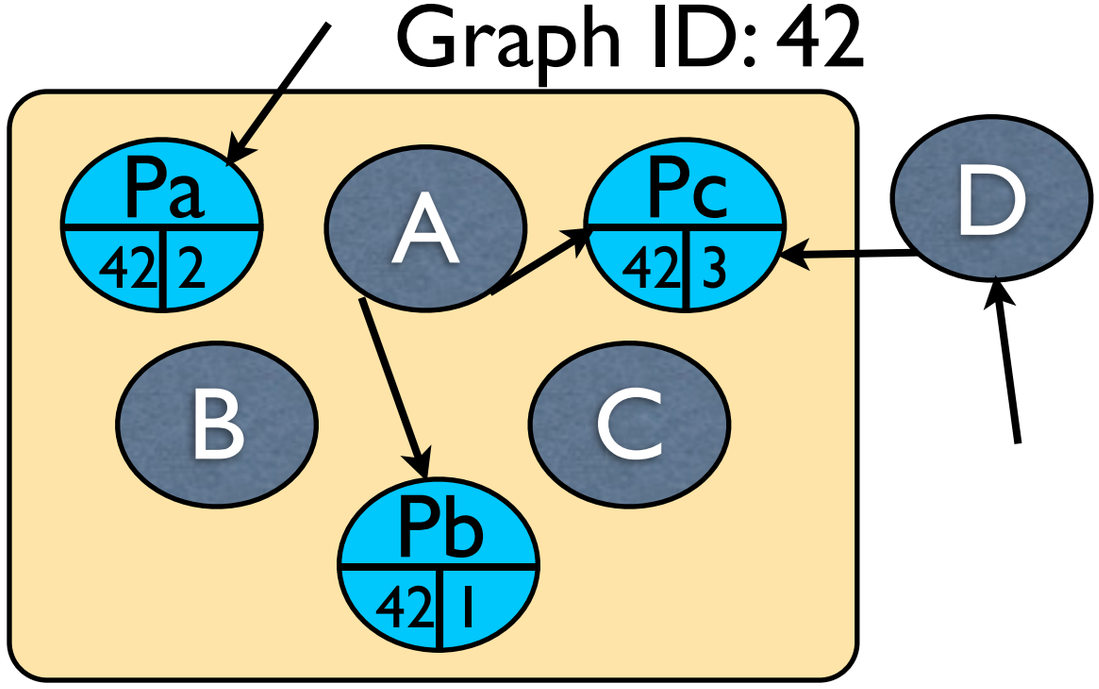


42.swap

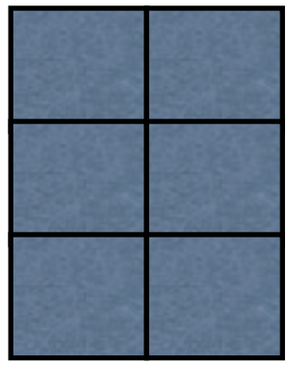
1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies

Swapping out

Primary memory

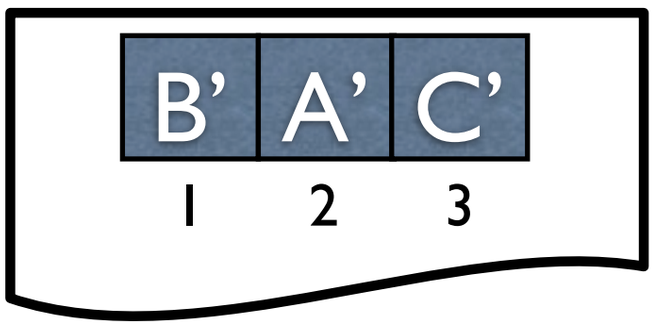


proxiesDict



GraphTable

Secondary memory

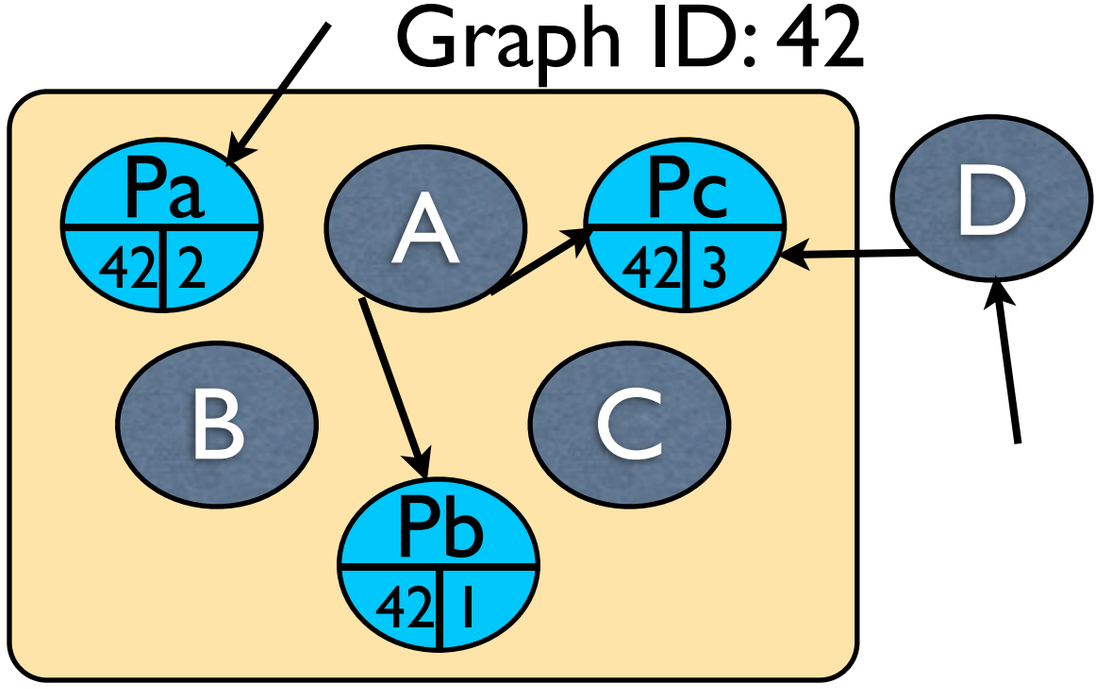


42.swap

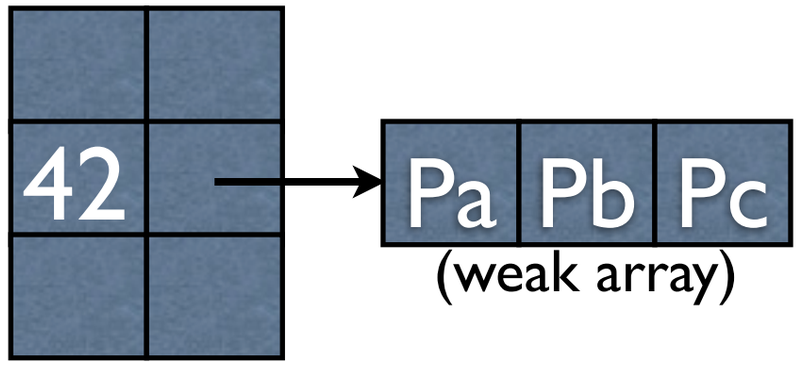
1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies

Swapping out

Primary memory

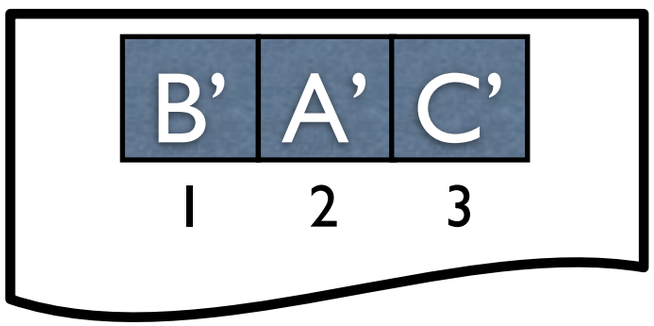


proxiesDict



GraphTable

Secondary memory

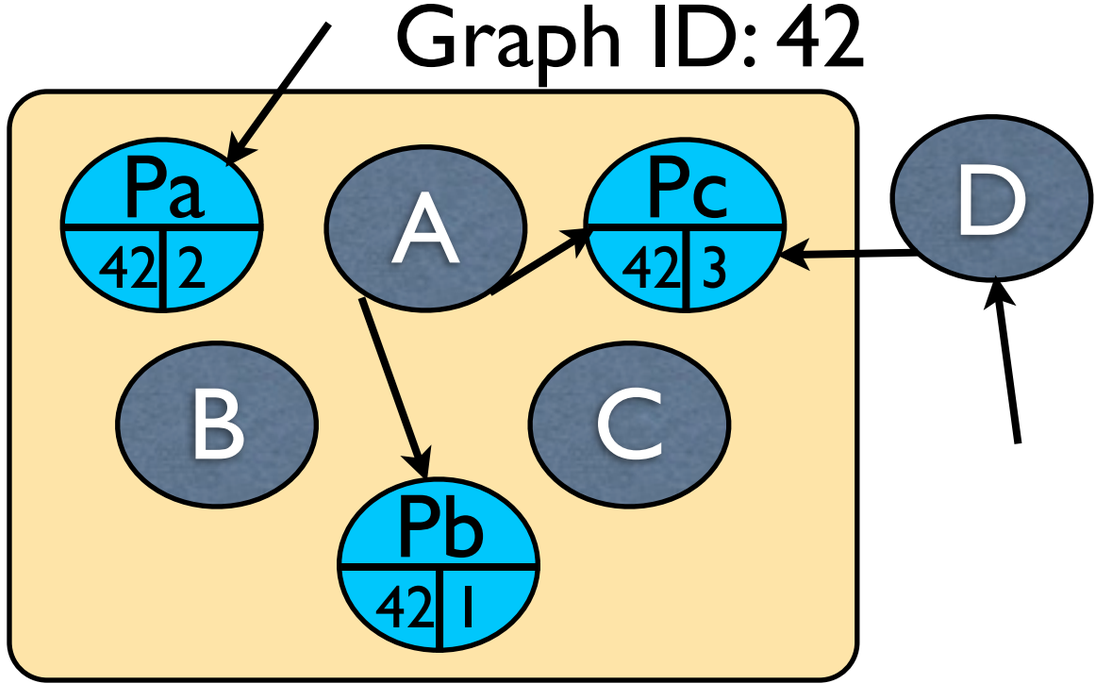


42.swap

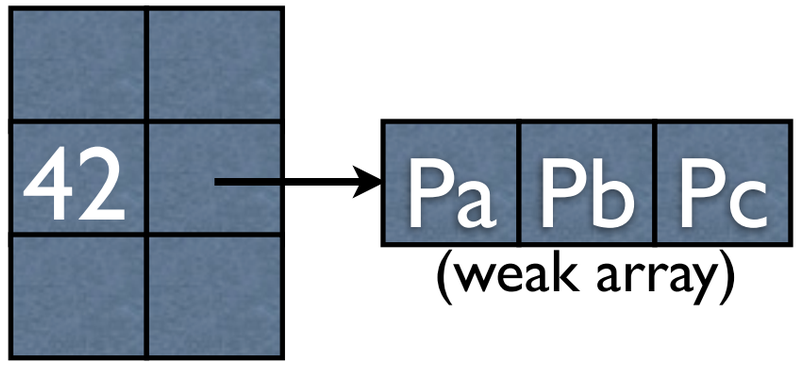
1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable

Swapping out

Primary memory

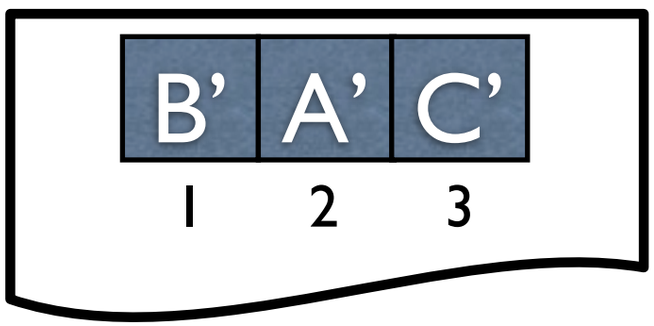


proxiesDict



GraphTable

Secondary memory

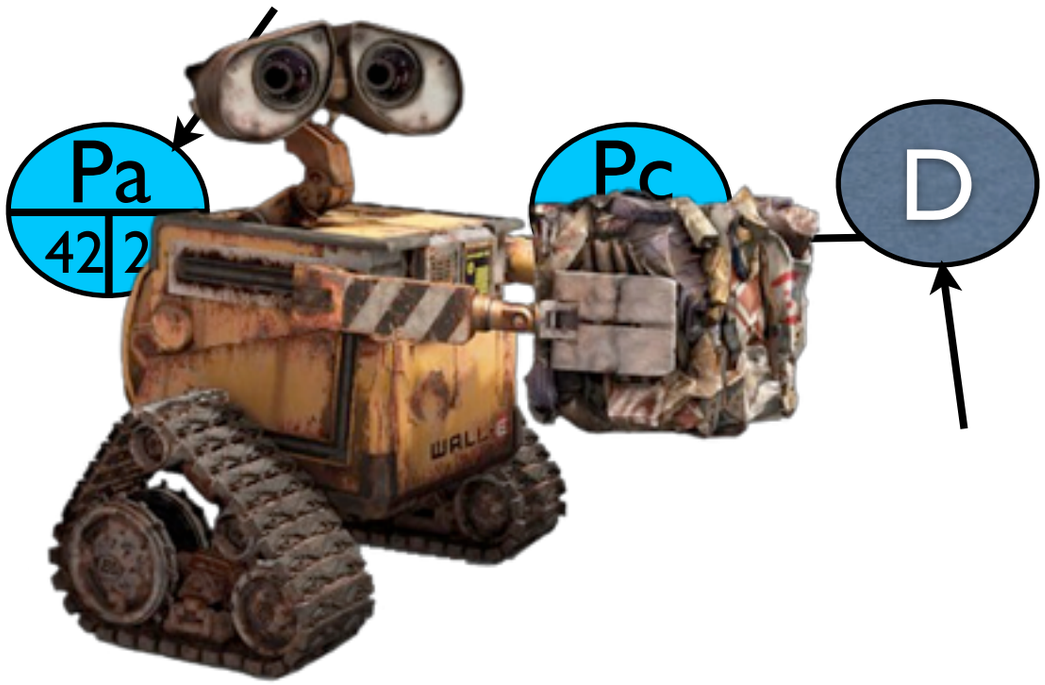


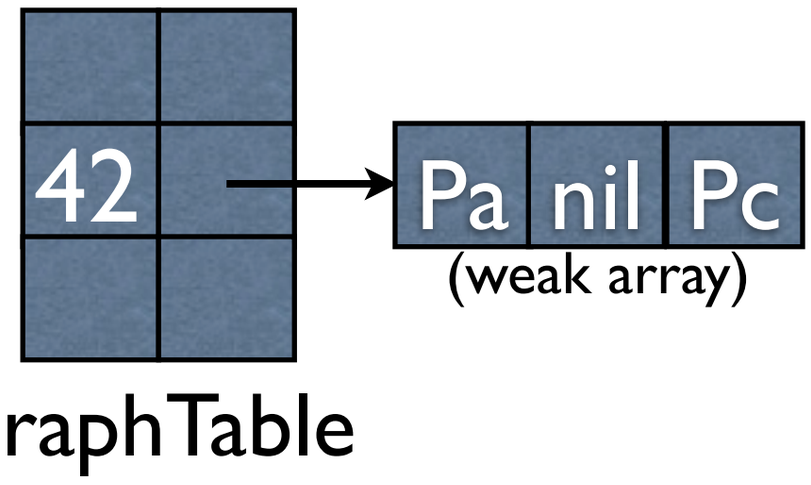
42.swap

1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs

Swapping out

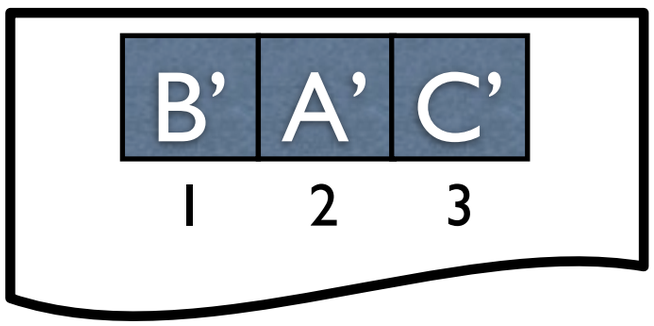
Primary memory





GraphTable

Secondary memory

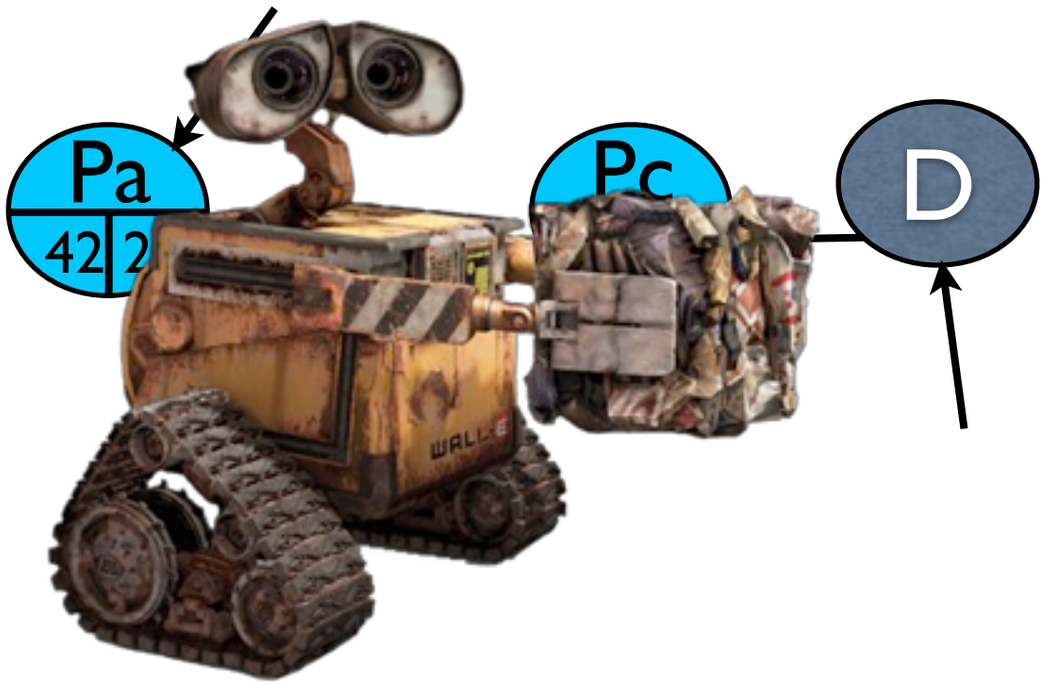


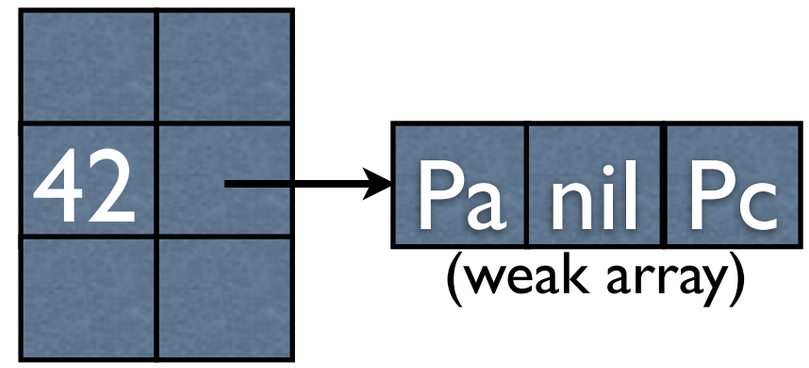
42.swap

1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs

Swapping out

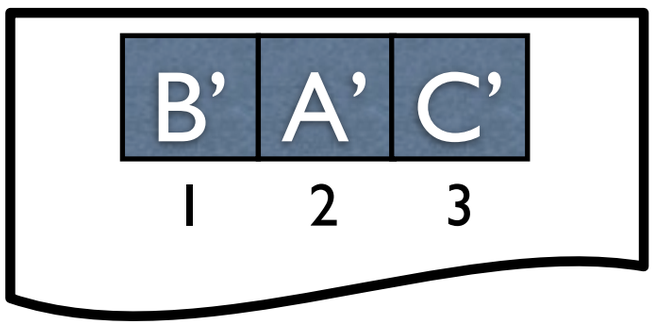
Primary memory





GraphTable

Secondary memory

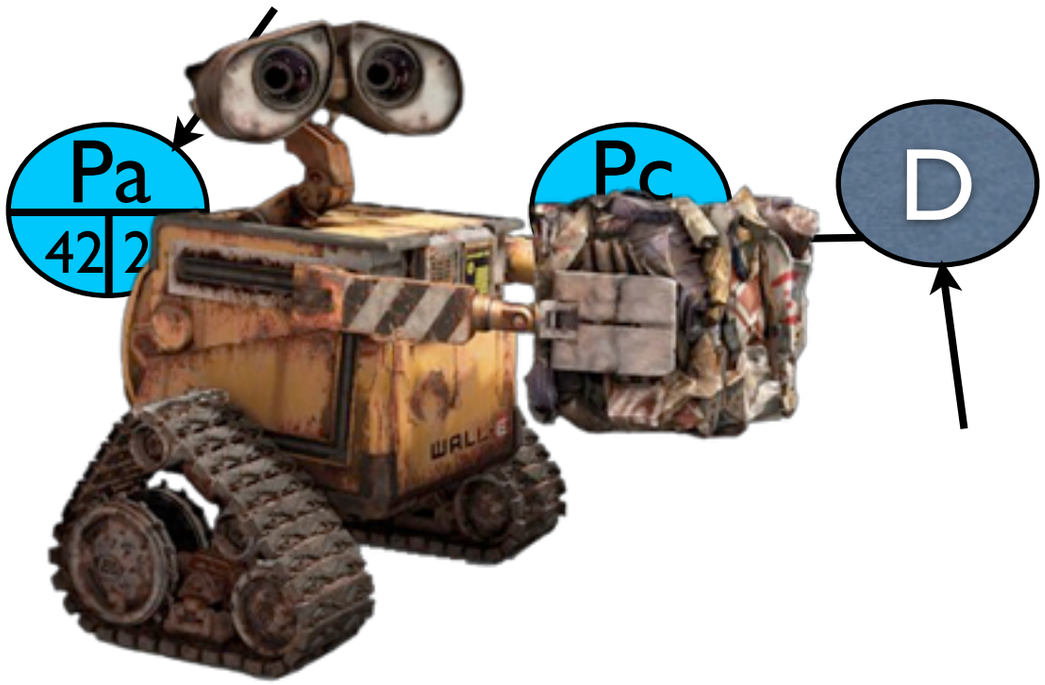


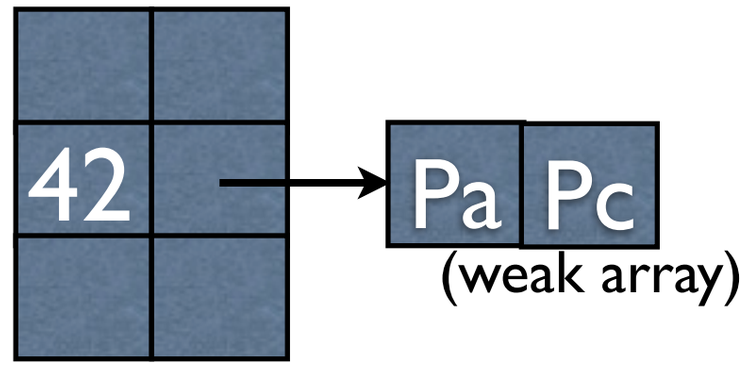
42.swap

1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Compact (optional)

Swapping out

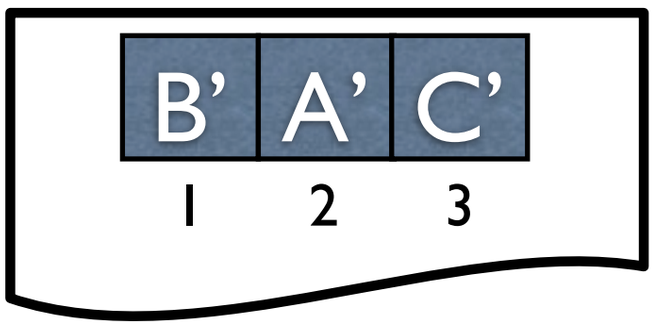
Primary memory





GraphTable

Secondary memory

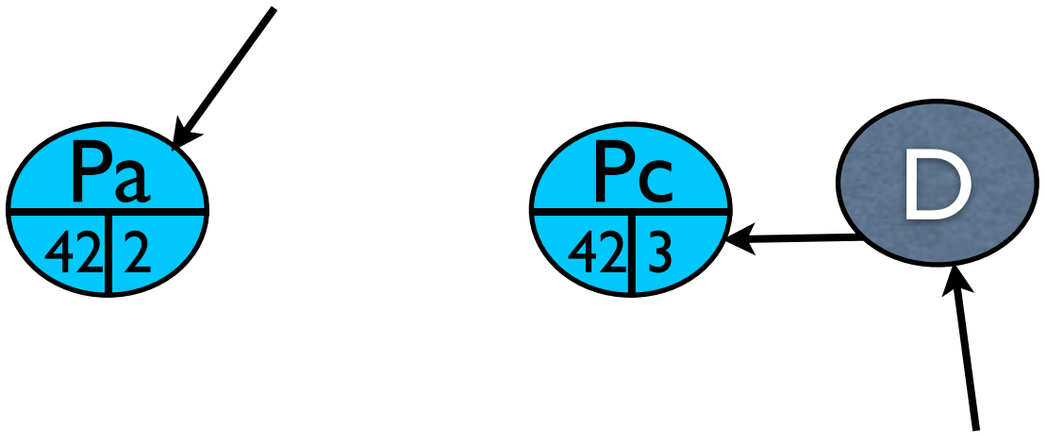


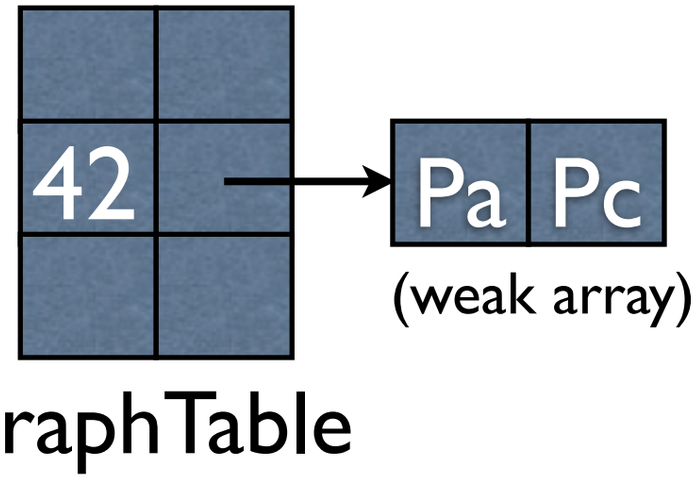
42.swap

1) Assign graph ID
2) Serialize the object graph
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Compact (optional)

Swapping in

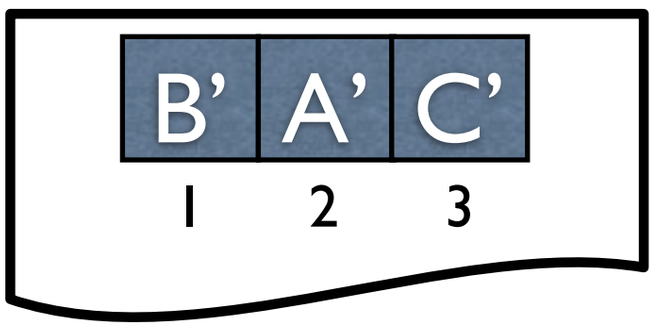
Primary memory





GraphTable

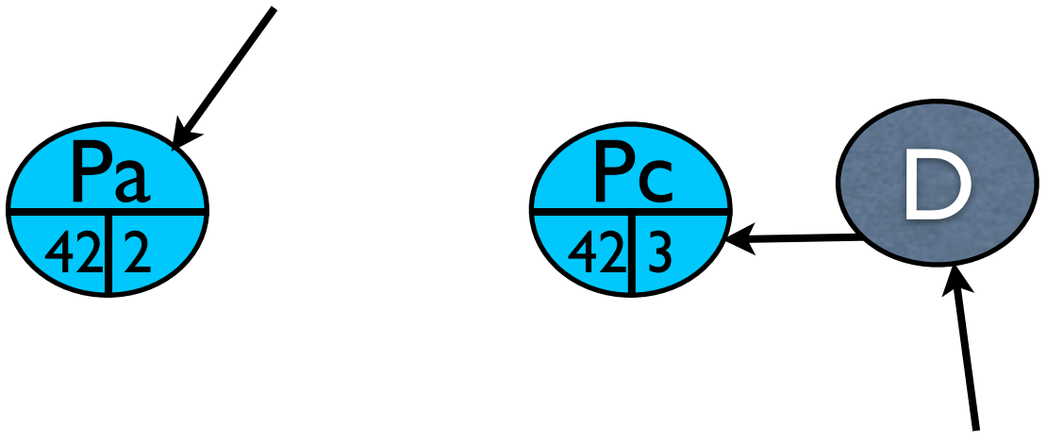
Secondary memory

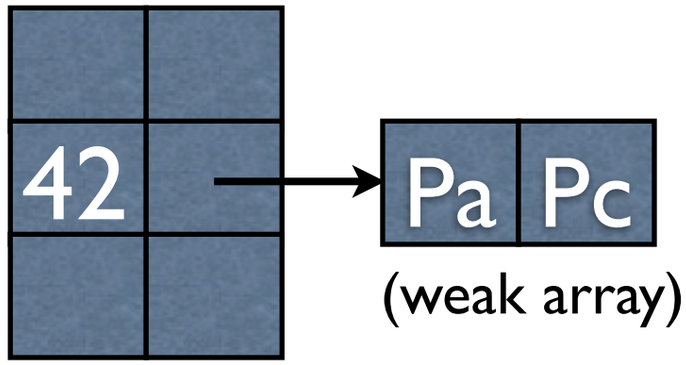


42.swap

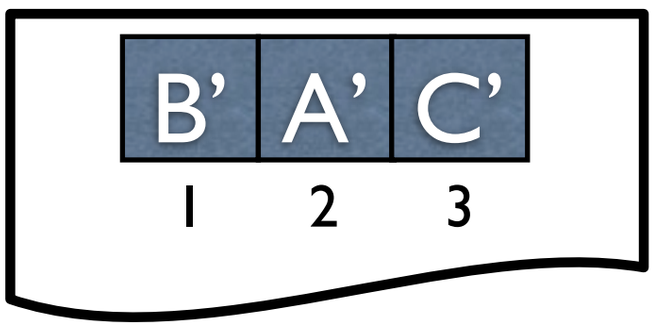
Swapping in

Primary memory





Secondary memory

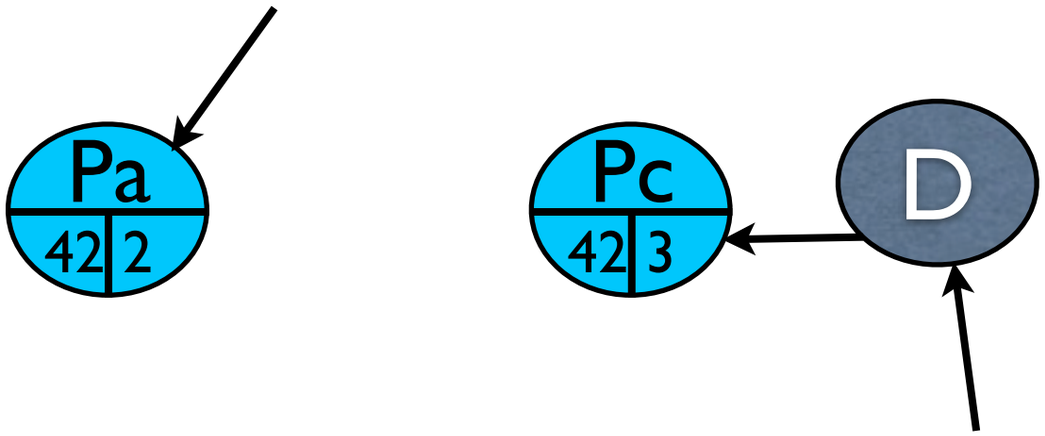


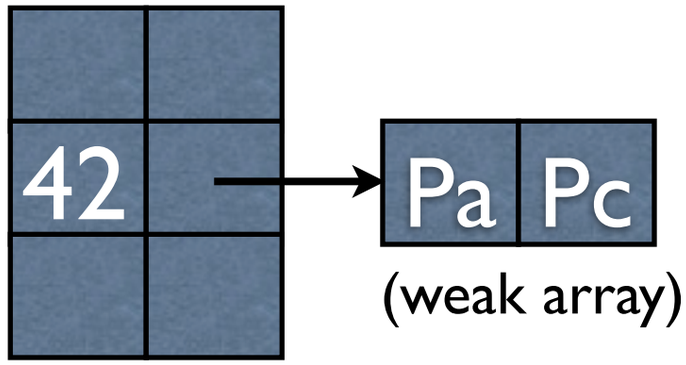
42.swap

0) A proxy intercepts a message...

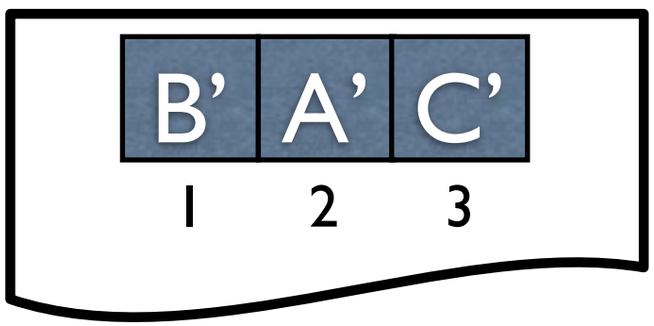
Swapping in

Primary memory





Secondary memory

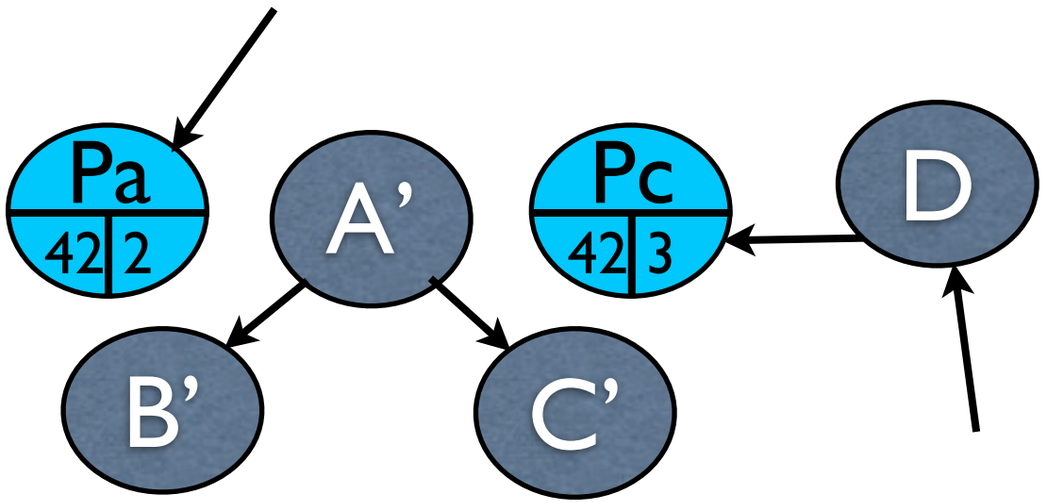


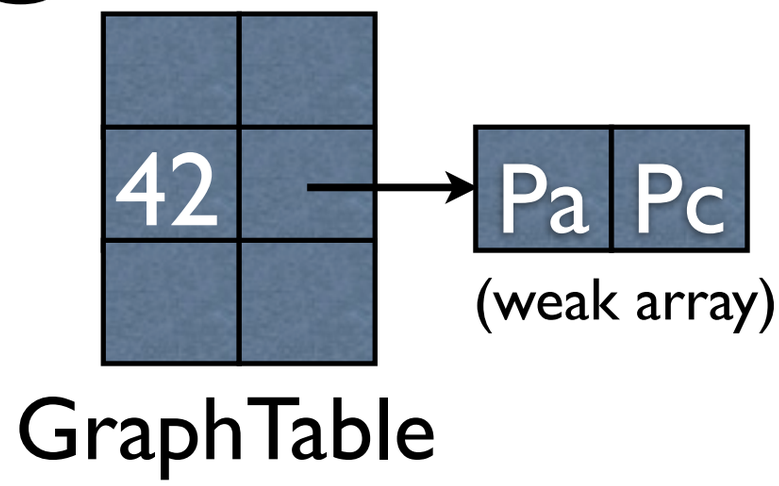
42.swap

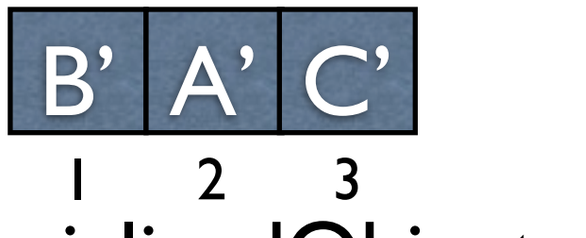
0) A proxy intercepts a message...

Swapping in

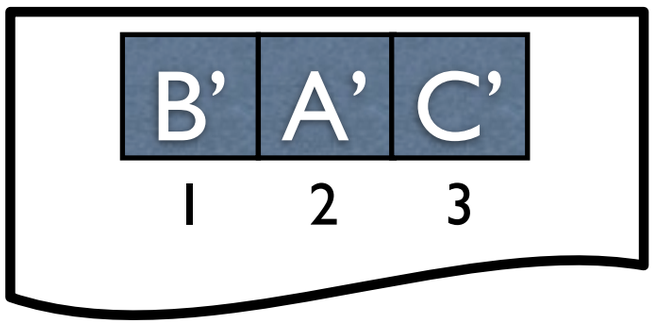
Primary memory







Secondary memory



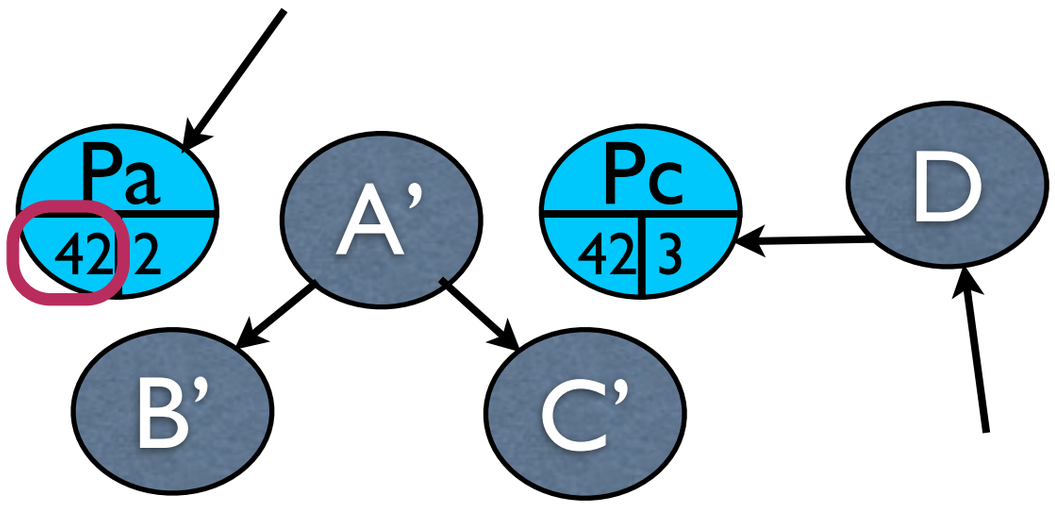
42.swap

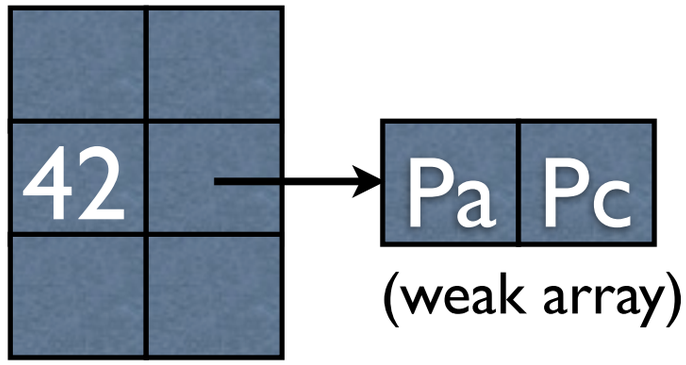
0) A proxy intercepts a message...

1) Materialize the object graph

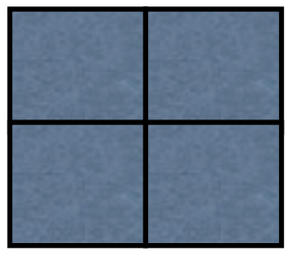
Swapping in

Primary memory





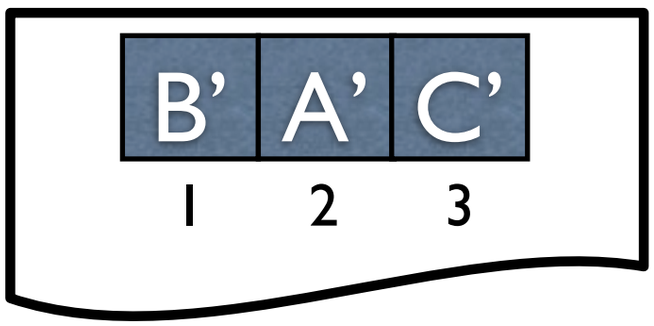
GraphTable



proxiesDict

materializedObjects

Secondary memory



42.swap

0) A proxy intercepts a message...

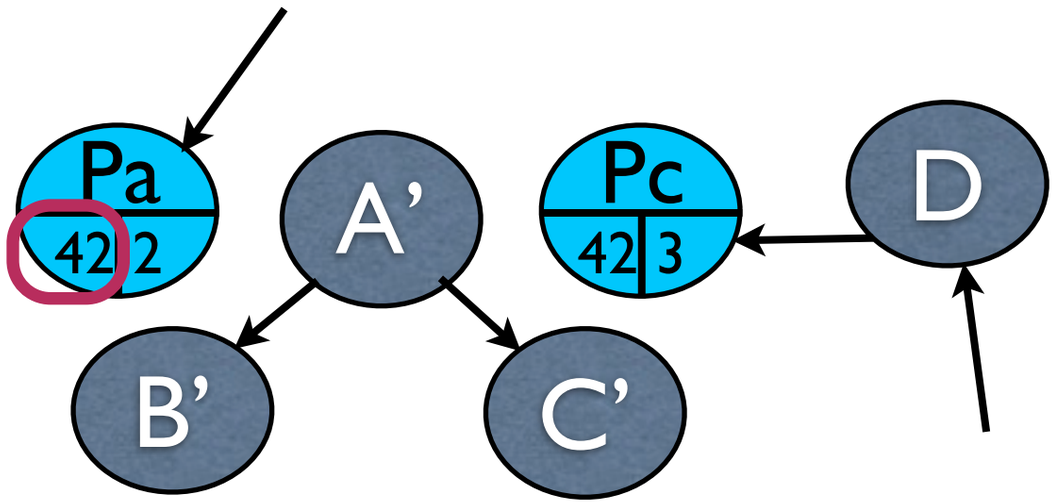
---

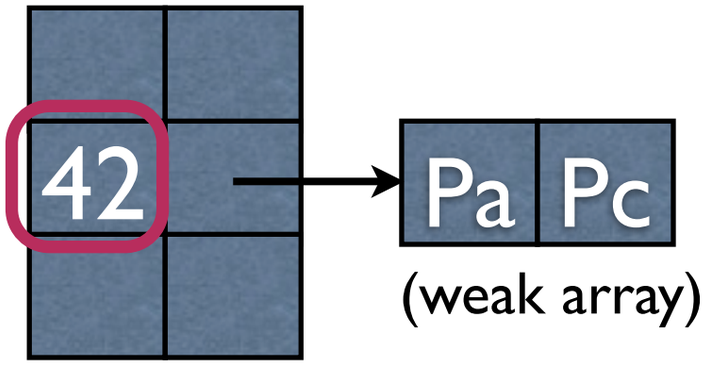
1) Materialize the object graph

2) Associate proxies with materialized objects

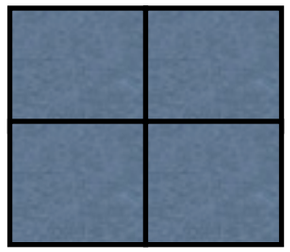
Swapping in

Primary memory





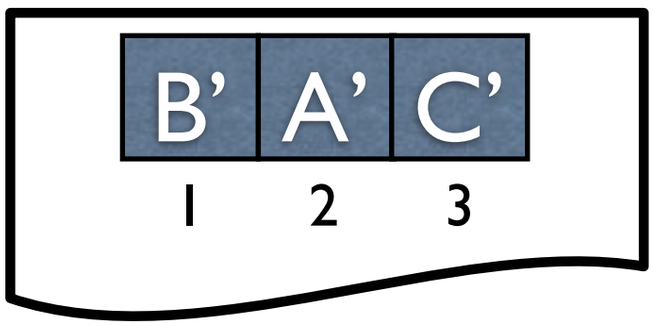
GraphTable



proxiesDict

materializedObjects

Secondary memory



42.swap

0) A proxy intercepts a message...

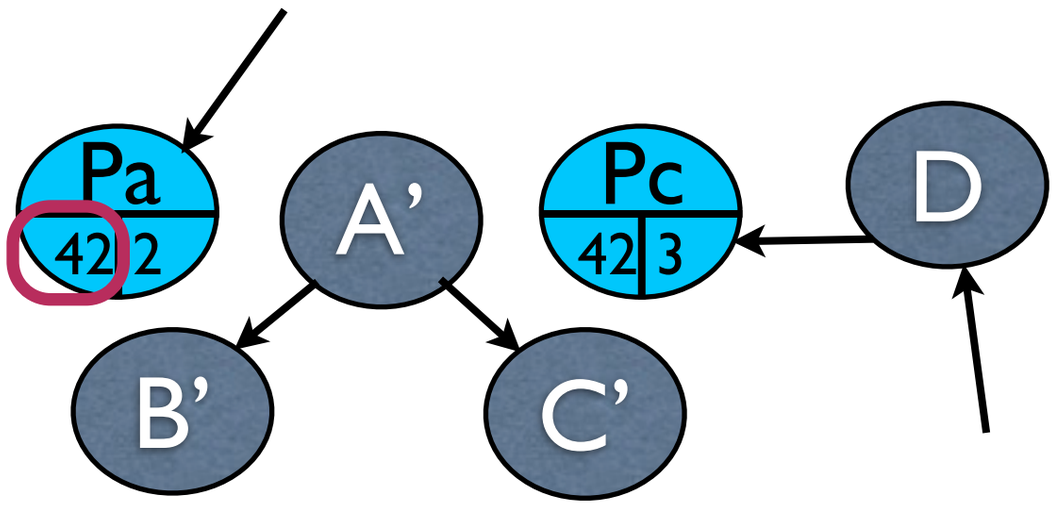
---

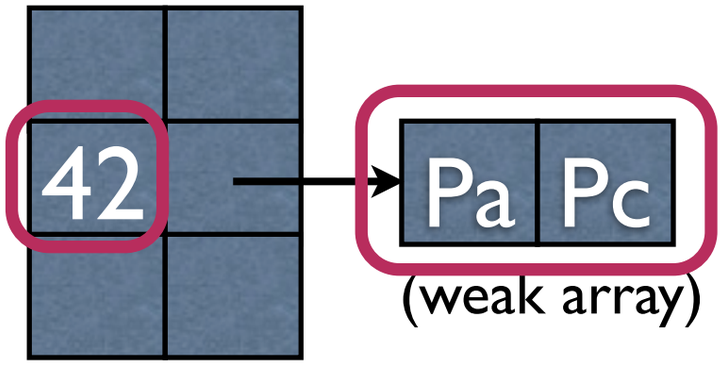
1) Materialize the object graph

2) Associate proxies with materialized objects

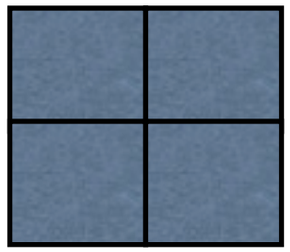
Swapping in

Primary memory





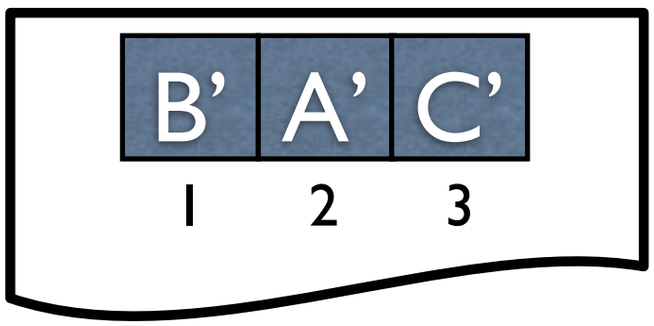
GraphTable



proxiesDict

materializedObjects

Secondary memory



42.swap

0) A proxy intercepts a message...

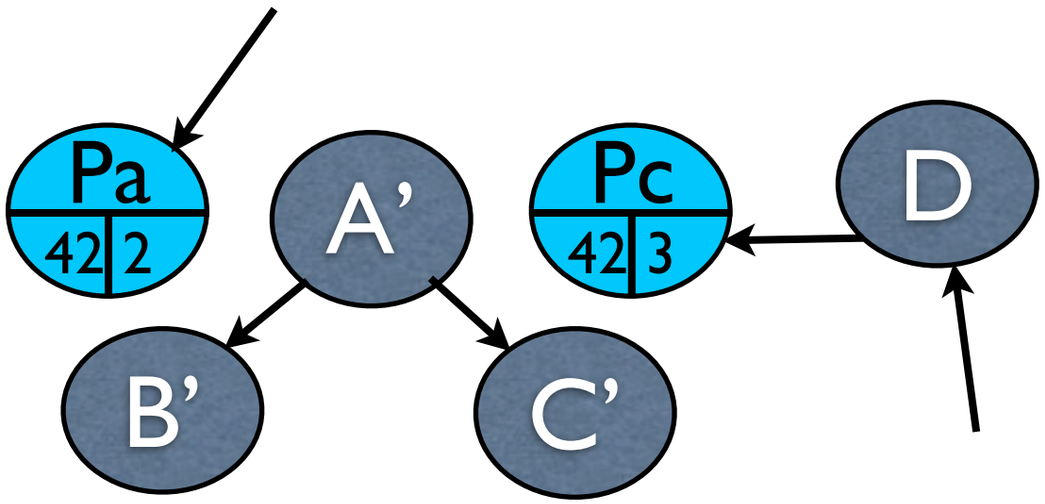
---

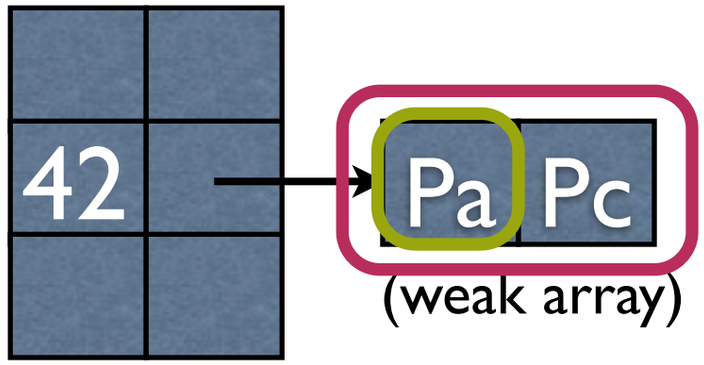
1) Materialize the object graph

2) Associate proxies with materialized objects

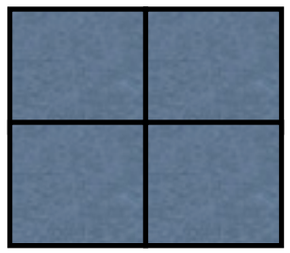
Swapping in

Primary memory





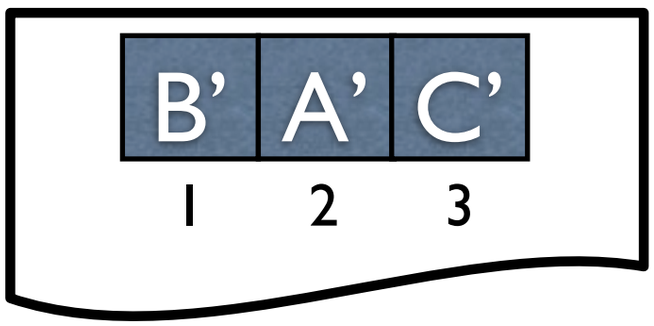
GraphTable



proxiesDict

materializedObjects

Secondary memory



42.swap

0) A proxy intercepts a message...

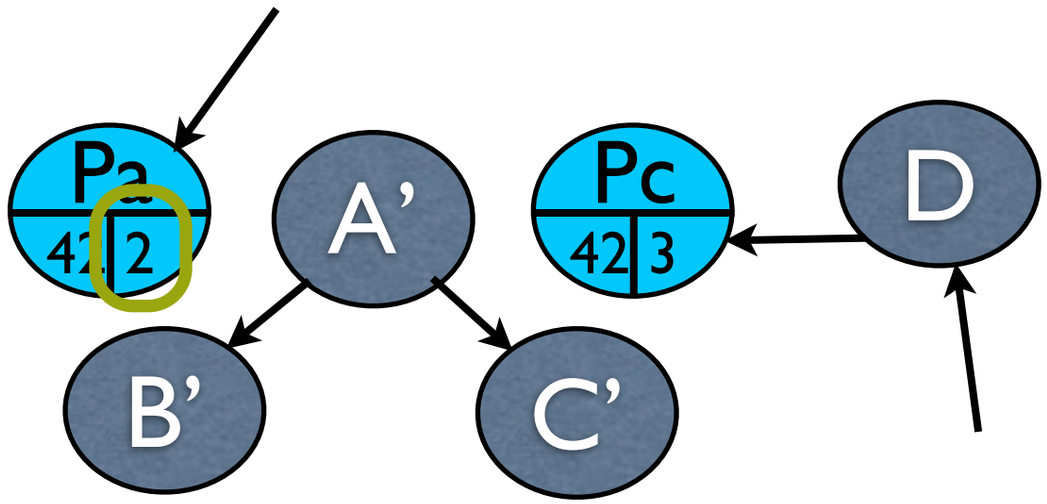
---

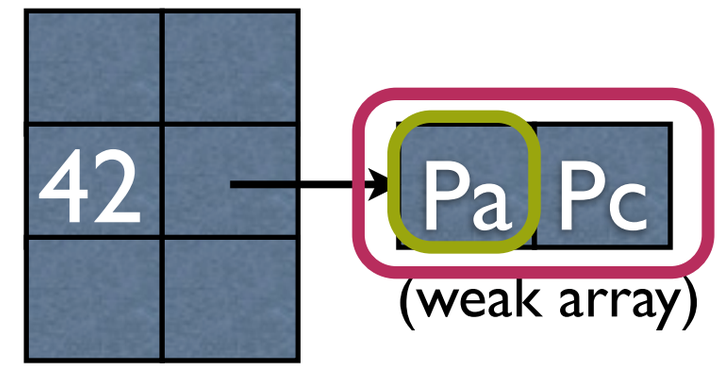
1) Materialize the object graph

2) Associate proxies with materialized objects

Swapping in

Primary memory



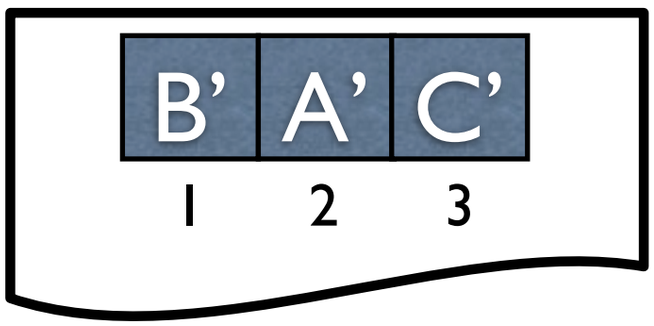


GraphTable

proxiesDict

materializedObjects

Secondary memory



42.swap

0) A proxy intercepts a message...

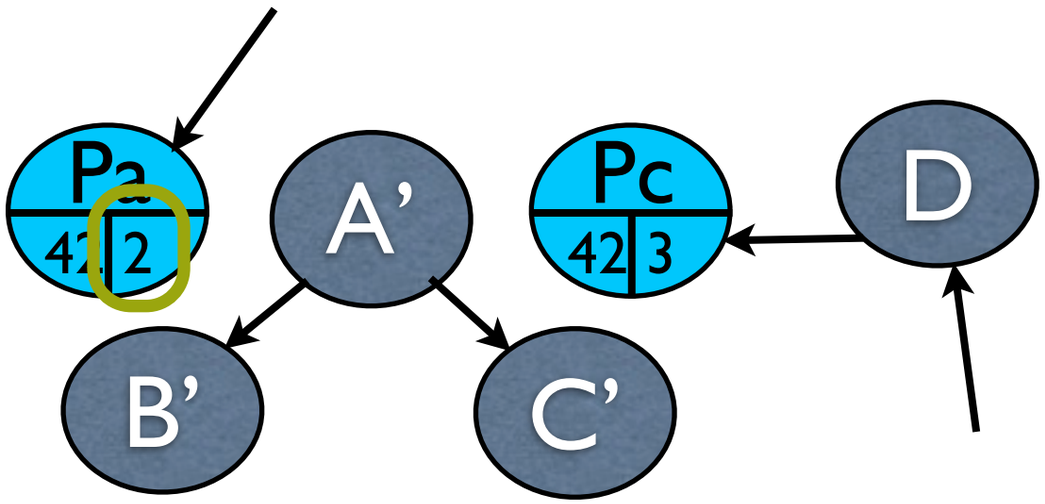
---

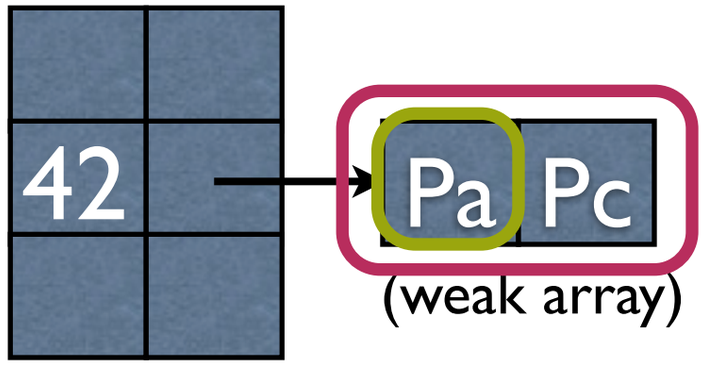
1) Materialize the object graph

2) Associate proxies with materialized objects

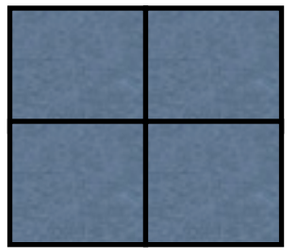
Swapping in

Primary memory

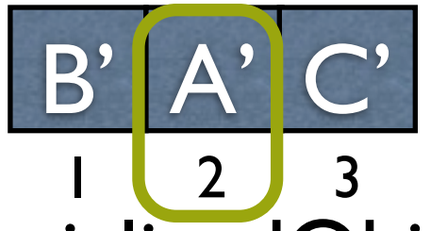




GraphTable

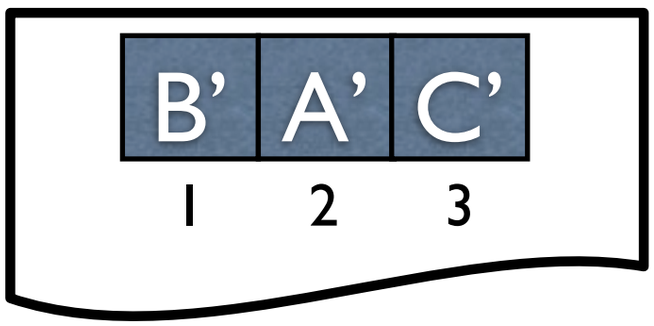


proxiesDict



materializedObjects

Secondary memory



42.swap

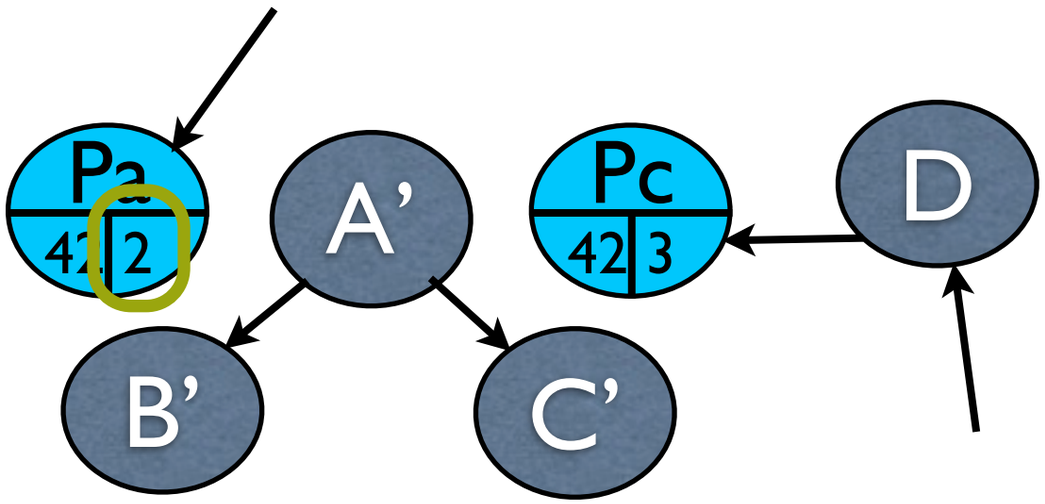
0) A proxy intercepts a message...

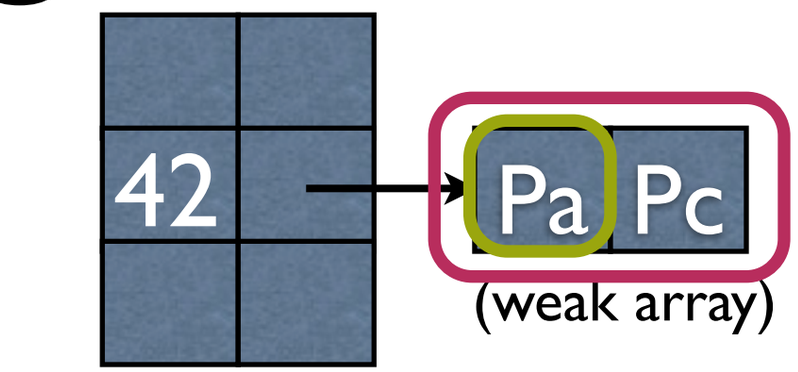
---

1) Materialize the object graph
2) Associate proxies with materialized objects

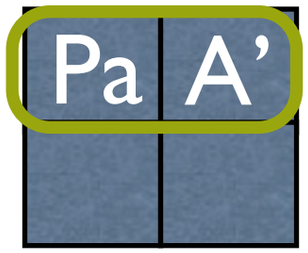
Swapping in

Primary memory

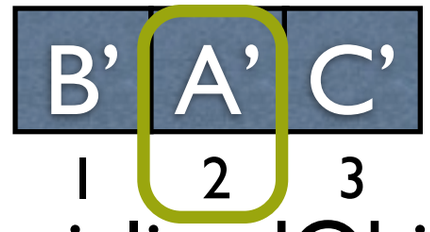




GraphTable

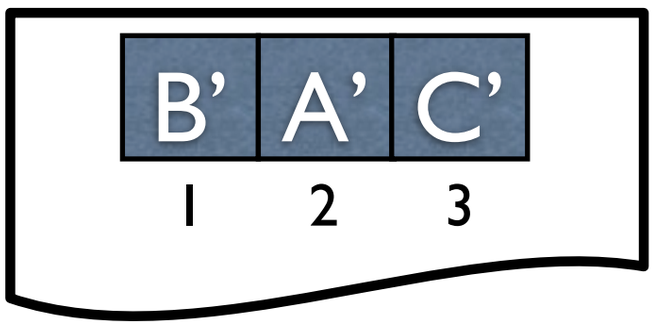


proxiesDict



materializedObjects

Secondary memory



42.swap

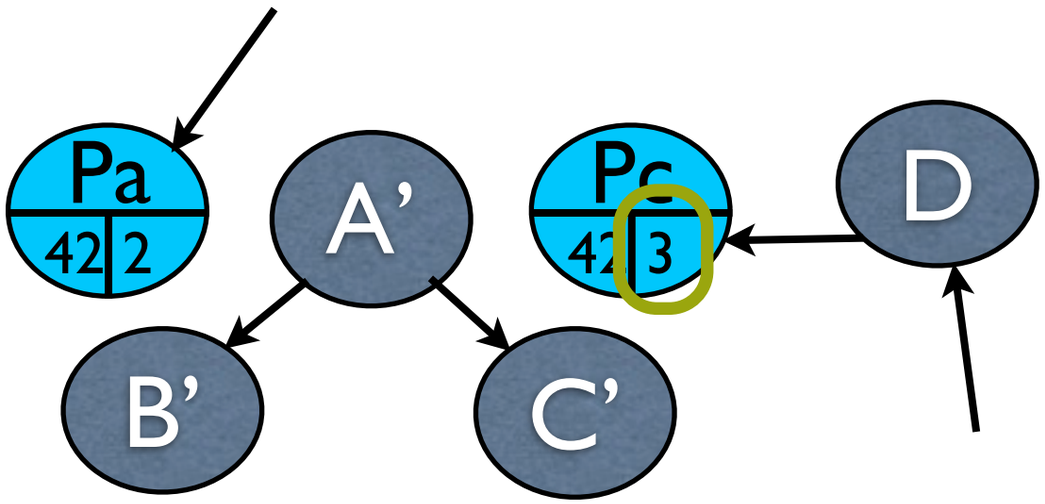
0) A proxy intercepts a message...

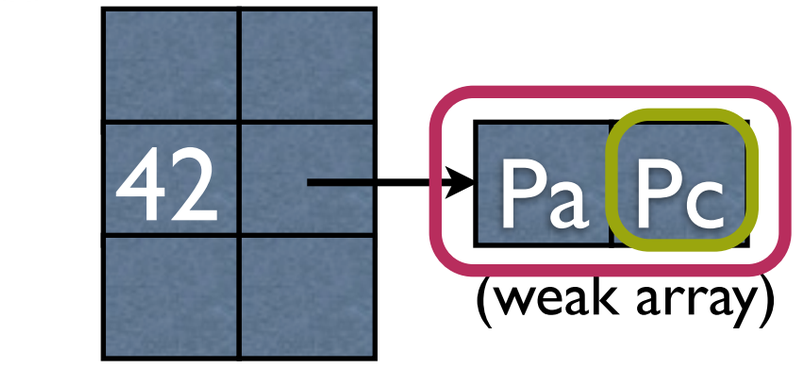
---

1) Materialize the object graph
2) Associate proxies with materialized objects

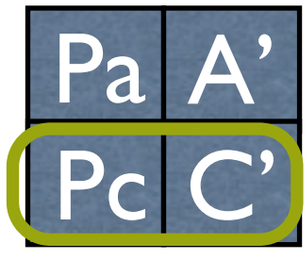
Swapping in

Primary memory

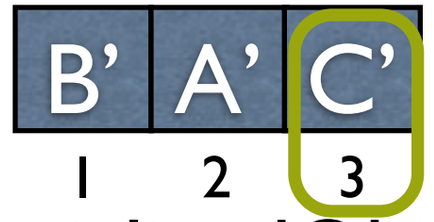




GraphTable

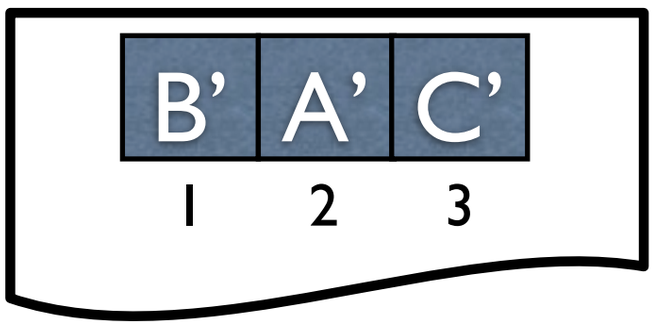


proxiesDict



materializedObjects

Secondary memory



42.swap

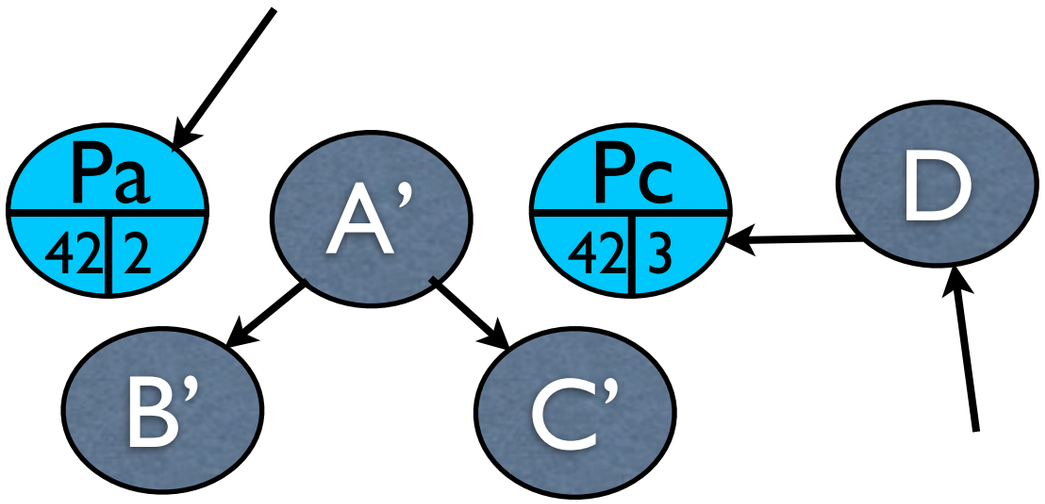
0) A proxy intercepts a message...

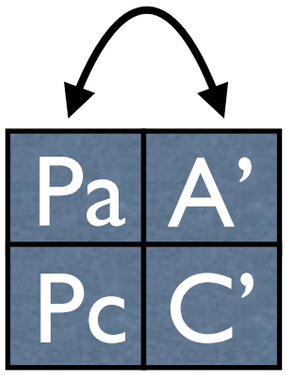
---

1) Materialize the object graph
2) Associate proxies with materialized objects

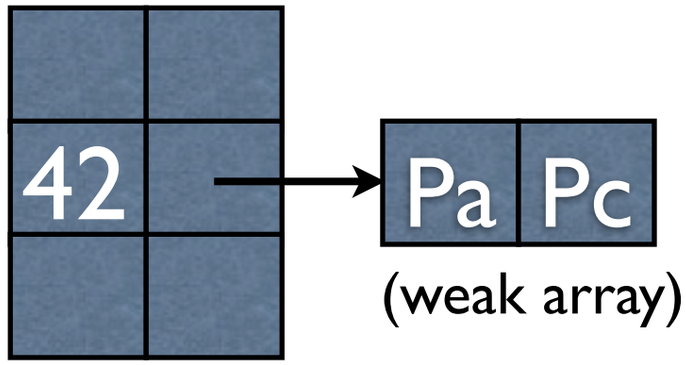
Swapping in

Primary memory





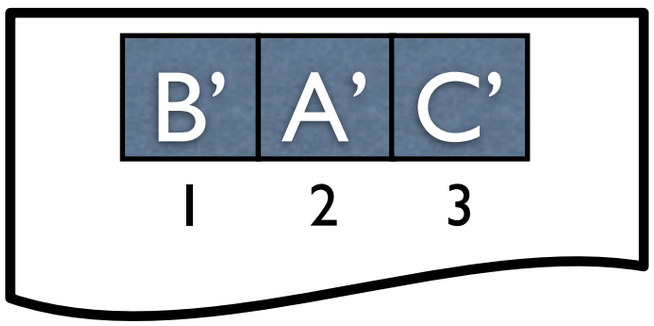
proxiesDict



GraphTable

materializedObjects

Secondary memory



42.swap

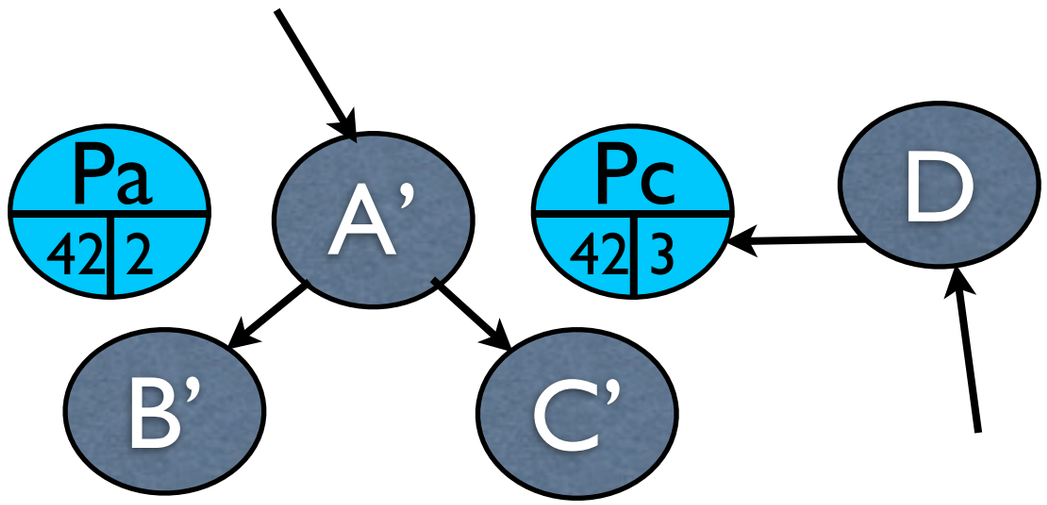
0) A proxy intercepts a message...

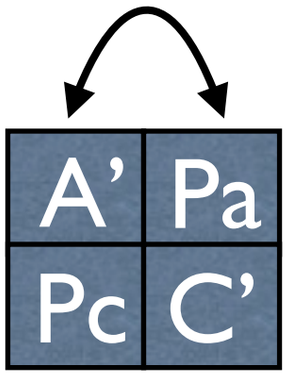
---

1) Materialize the object graph
2) Associate proxies with materialized objects
3) Replace proxies with original objects

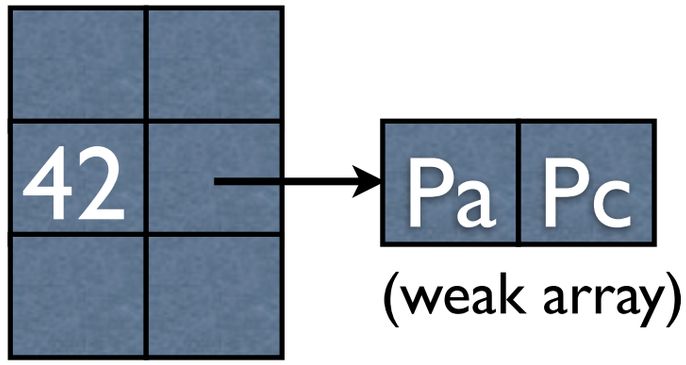
Swapping in

Primary memory





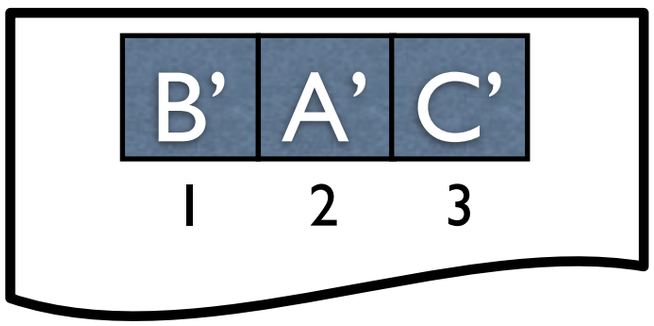
proxiesDict



GraphTable

materializedObjects

Secondary memory



42.swap

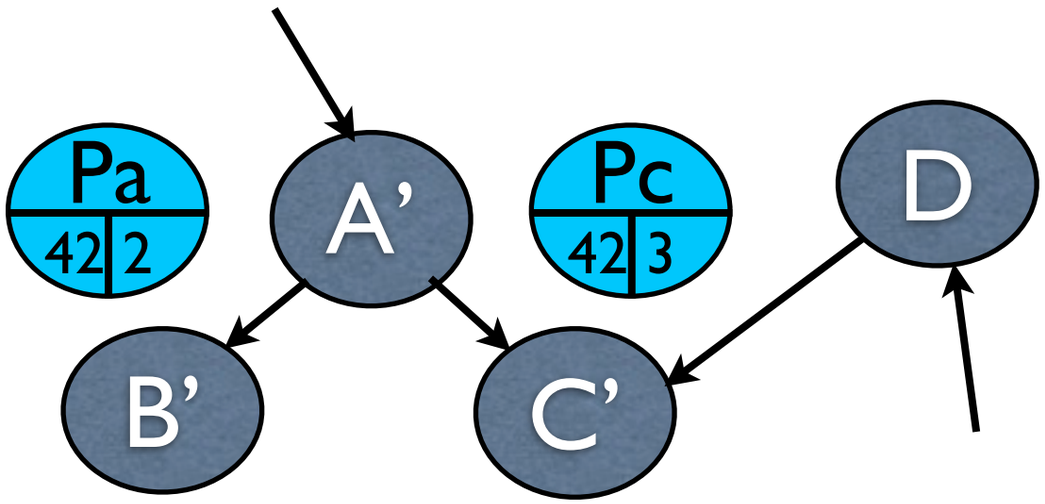
0) A proxy intercepts a message...

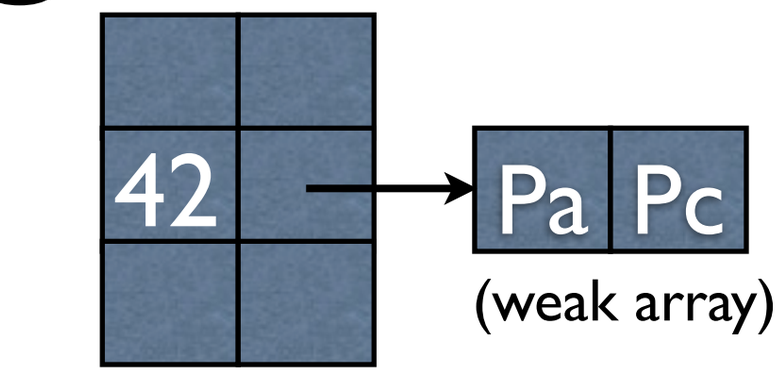
---

1) Materialize the object graph
2) Associate proxies with materialized objects
3) Replace proxies with original objects

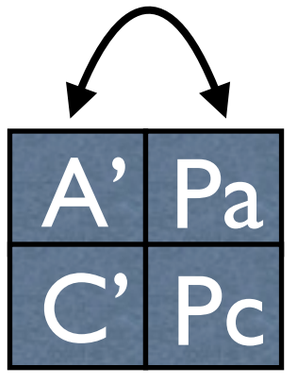
Swapping in

Primary memory





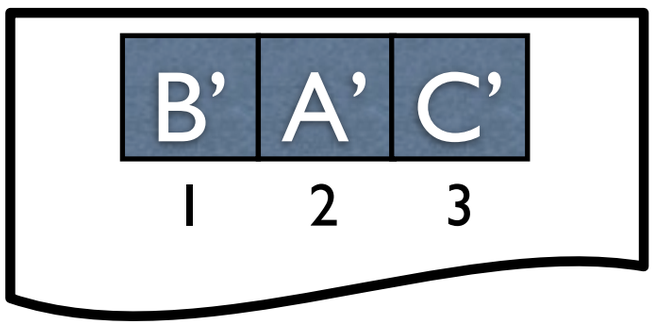
GraphTable



proxiesDict

materializedObjects

Secondary memory



42.swap

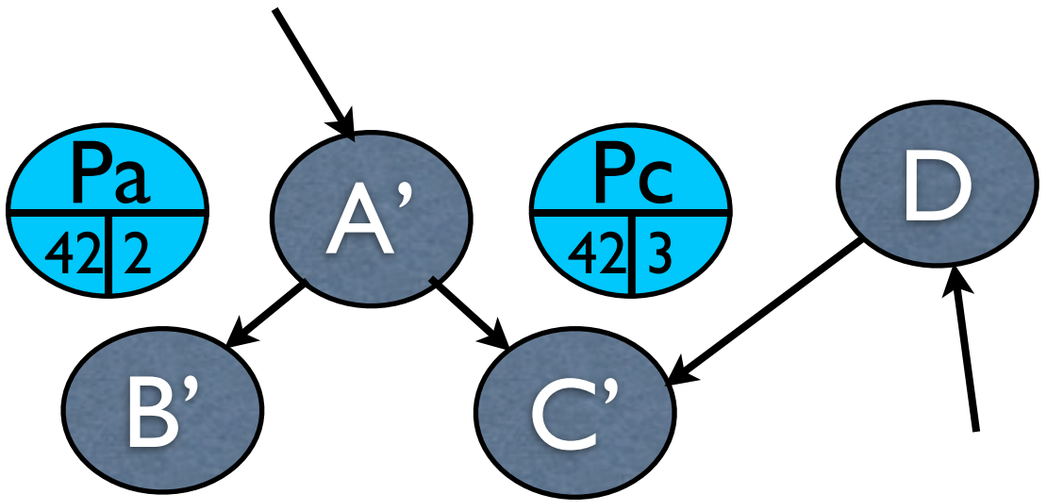
0) A proxy intercepts a message...

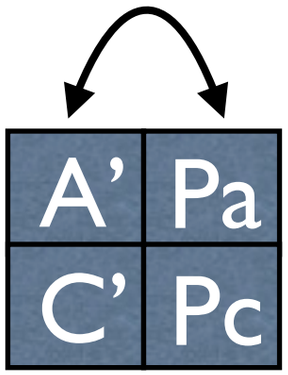
---

1) Materialize the object graph
2) Associate proxies with materialized objects
3) Replace proxies with original objects

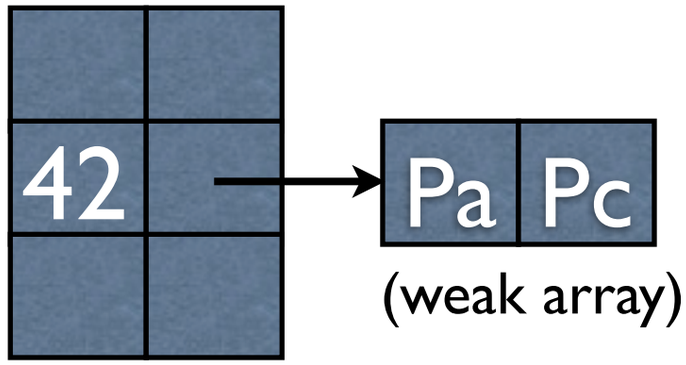
Swapping in

Primary memory





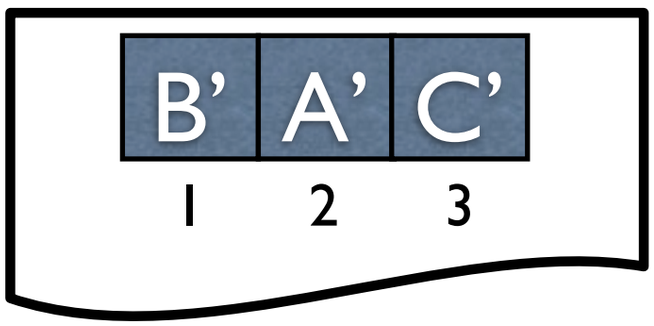
proxiesDict



GraphTable

materializedObjects

Secondary memory



42.swap

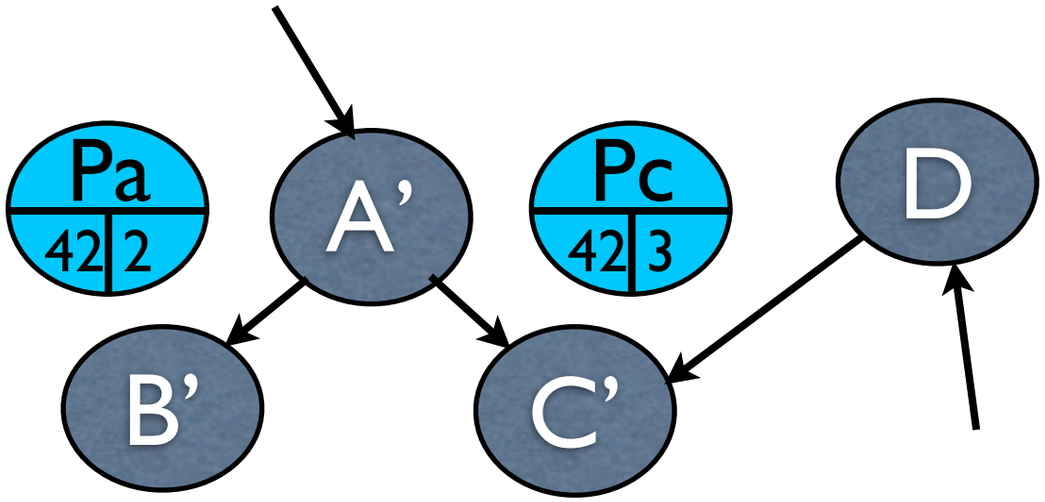
0) A proxy intercepts a message...

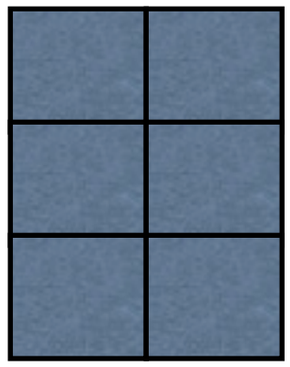
---

1) Materialize the object graph
2) Associate proxies with materialized objects
3) Replace proxies with original objects
4) Clean

Swapping in

Primary memory





GraphTable

Secondary memory

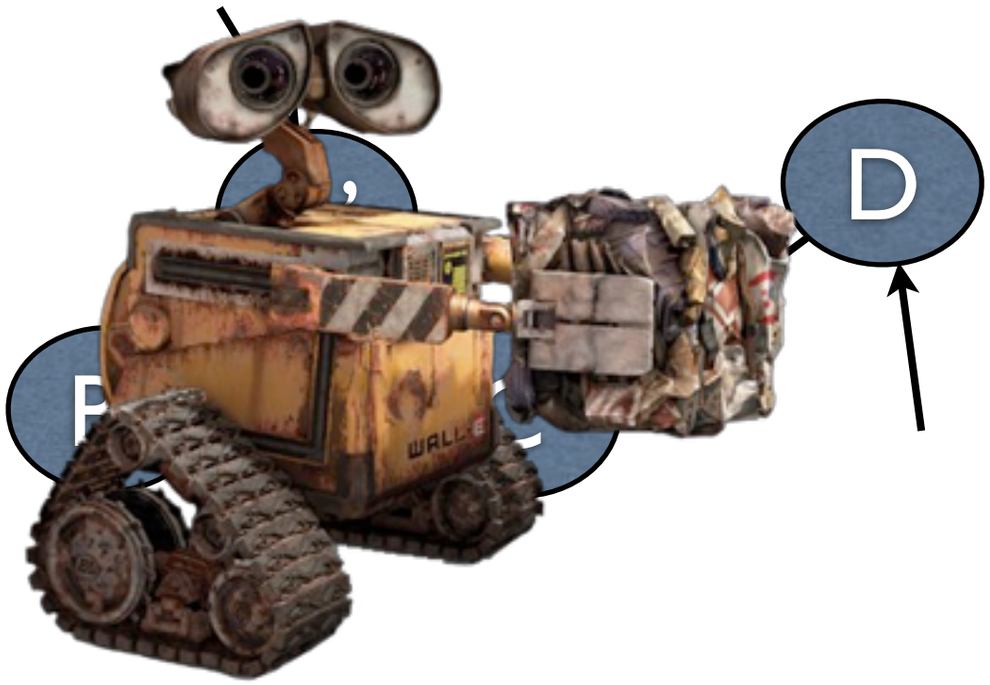
0) A proxy intercepts a message...

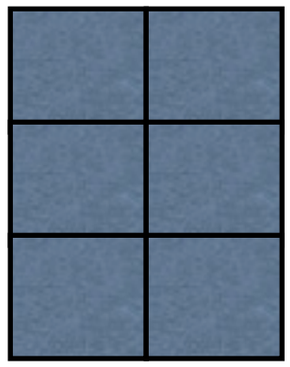
---

1) Materialize the object graph
2) Associate proxies with materialized objects
3) Replace proxies with original objects
4) Clean

Swapping in

Primary memory





GraphTable

Secondary memory

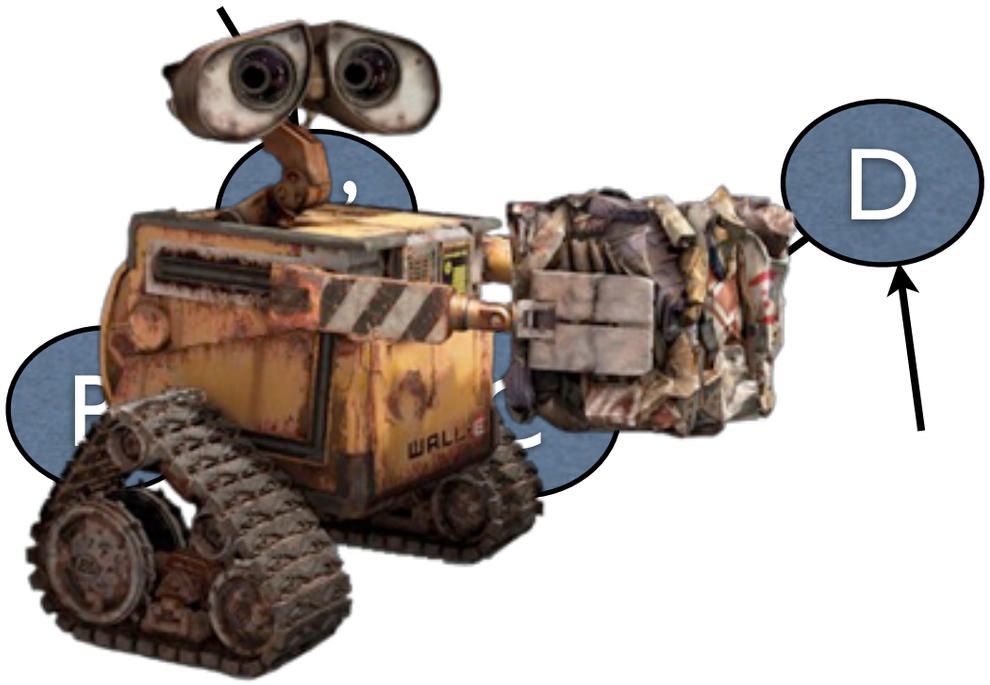
0) A proxy intercepts a message...

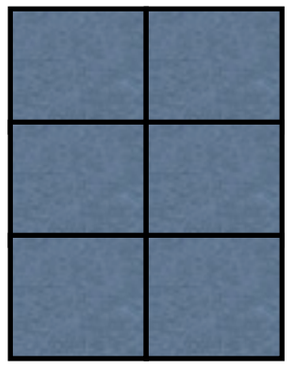
---

1) Materialize the object graph
2) Associate proxies with materialized objects
3) Replace proxies with original objects
4) Clean
5) GC runs

Swapping in

Primary memory





GraphTable

Secondary memory

0) A proxy intercepts a message...

---

1) Materialize the object graph

2) Associate proxies with materialized objects

3) Replace proxies with original objects

4) Clean

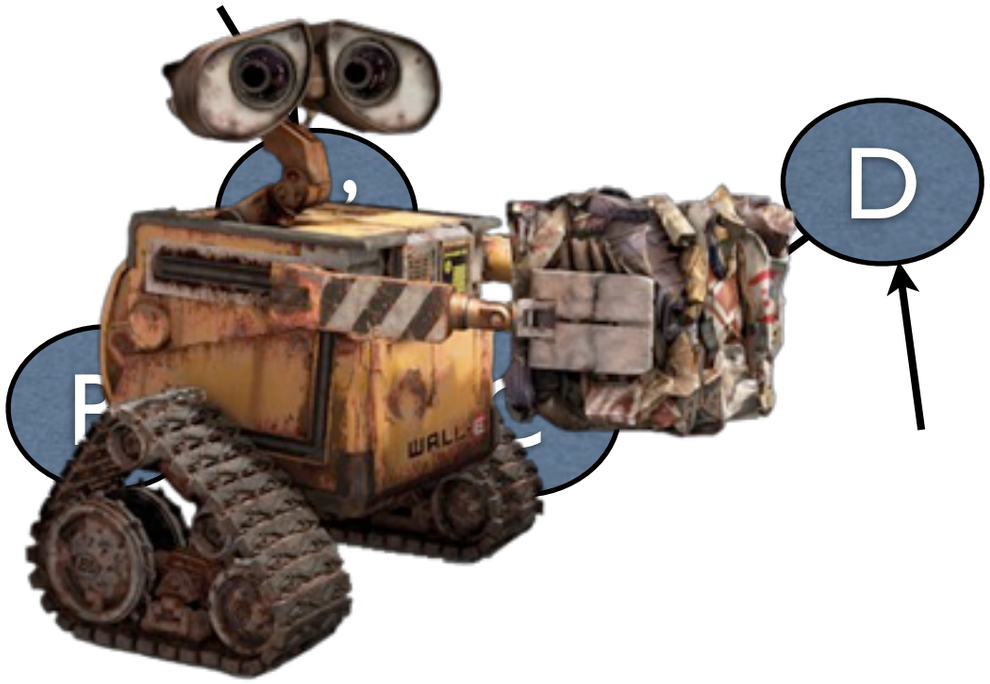
5) GC runs

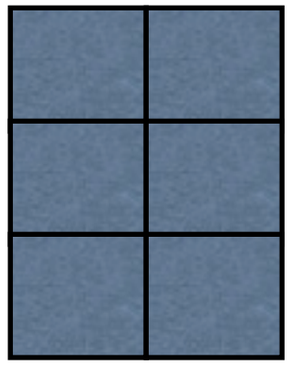
---

6) Forward message to object

Swapping in

Primary memory





GraphTable

Secondary memory

0) A proxy intercepts a message...

---

1) Materialize the object graph

2) Associate proxies with materialized objects

3) Replace proxies with original objects

4) Clean

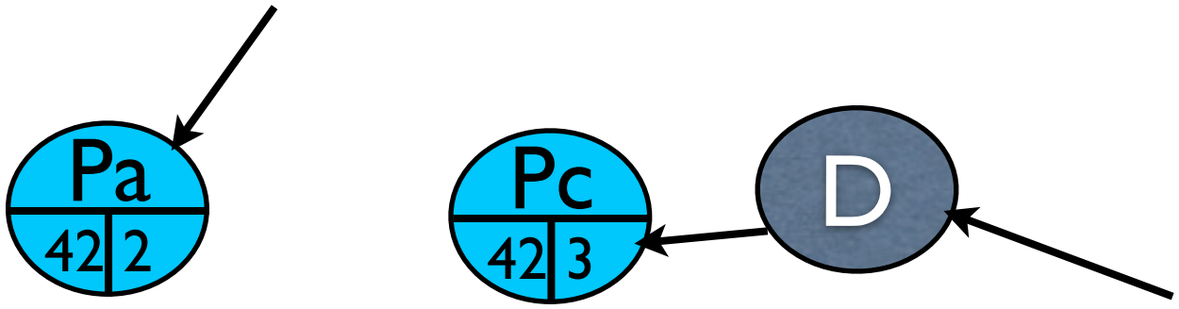
5) GC runs

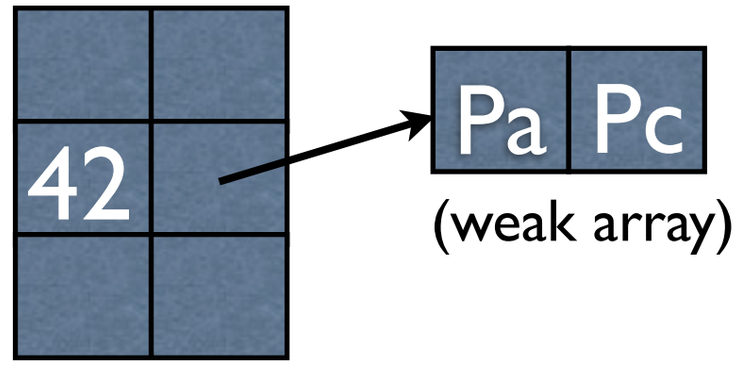
---

6) Forward message to object

Intersections...

Primary memory





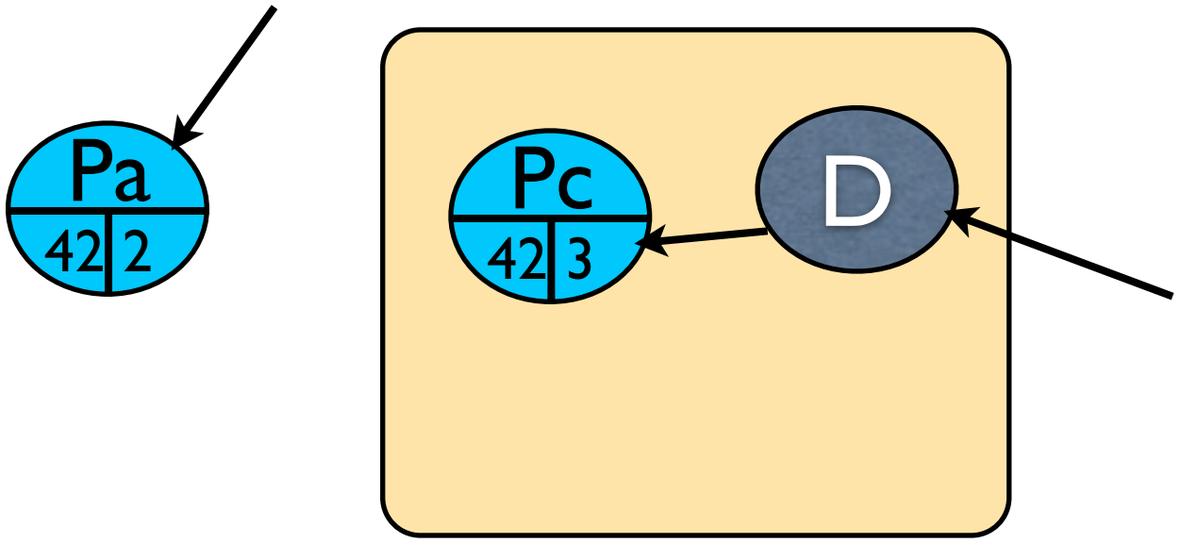
GraphTable

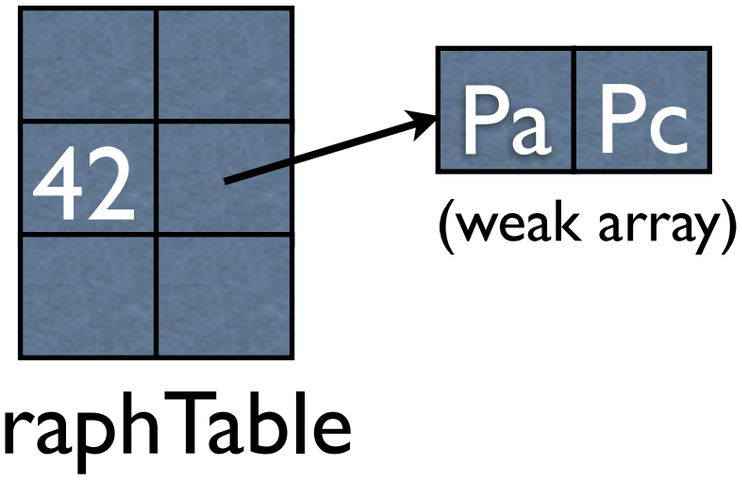
Secondary memory

42.swap

Intersections...

Primary memory

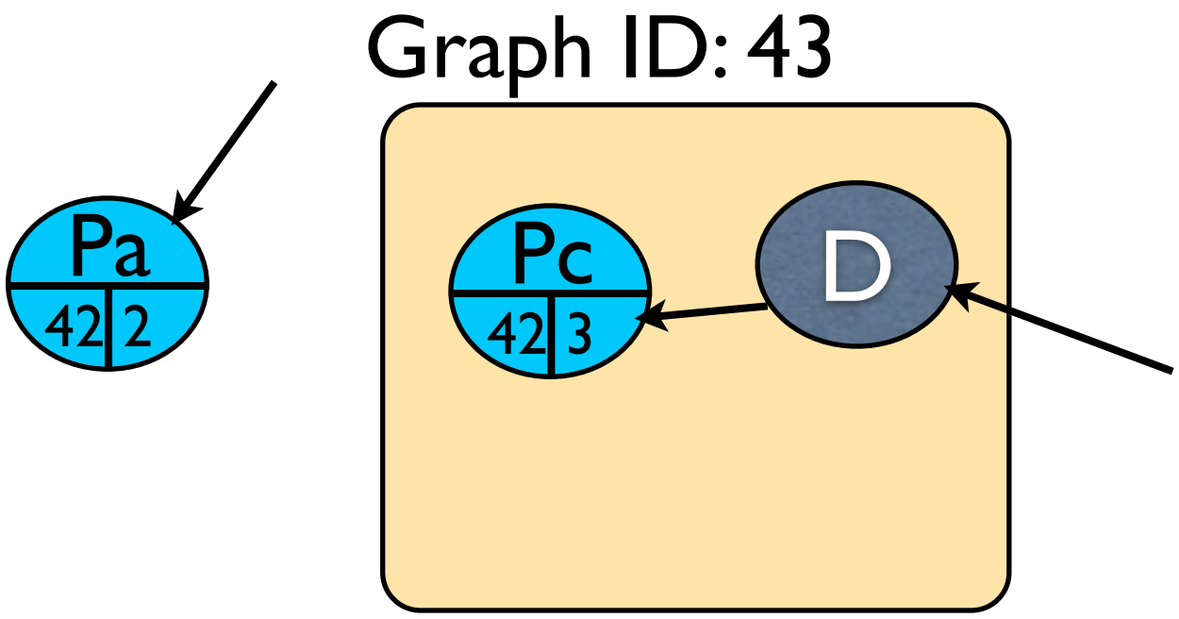


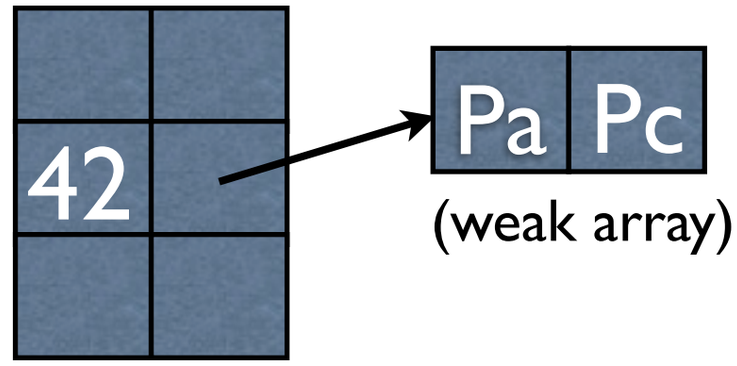


Secondary memory

Intersections...

Primary memory





GraphTable

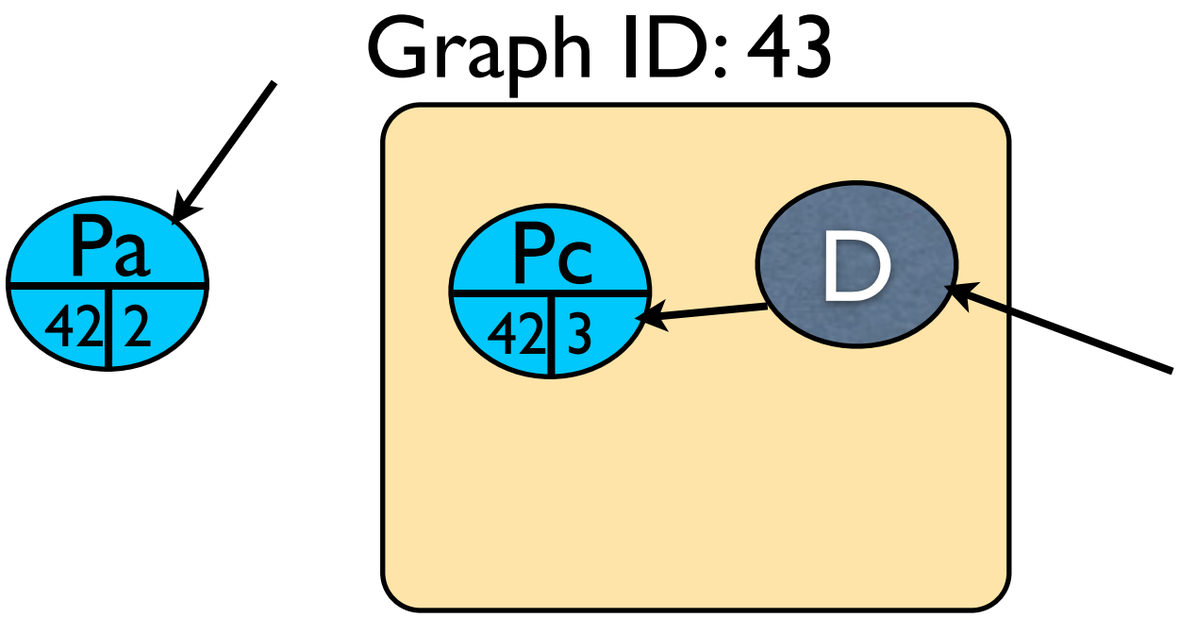
Secondary memory

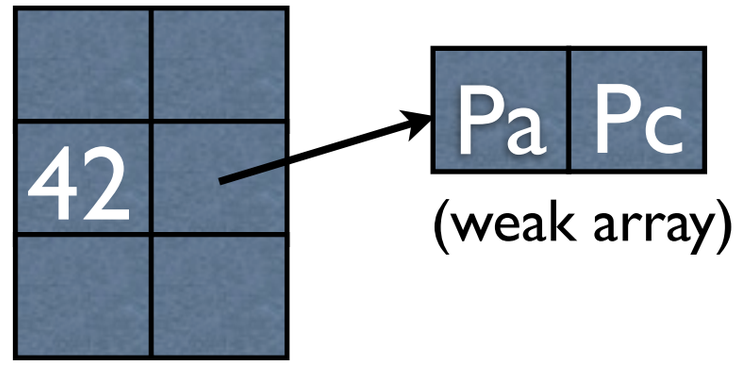
42.swap

1) Assign graph ID

Intersections...

Primary memory





GraphTable

Secondary memory

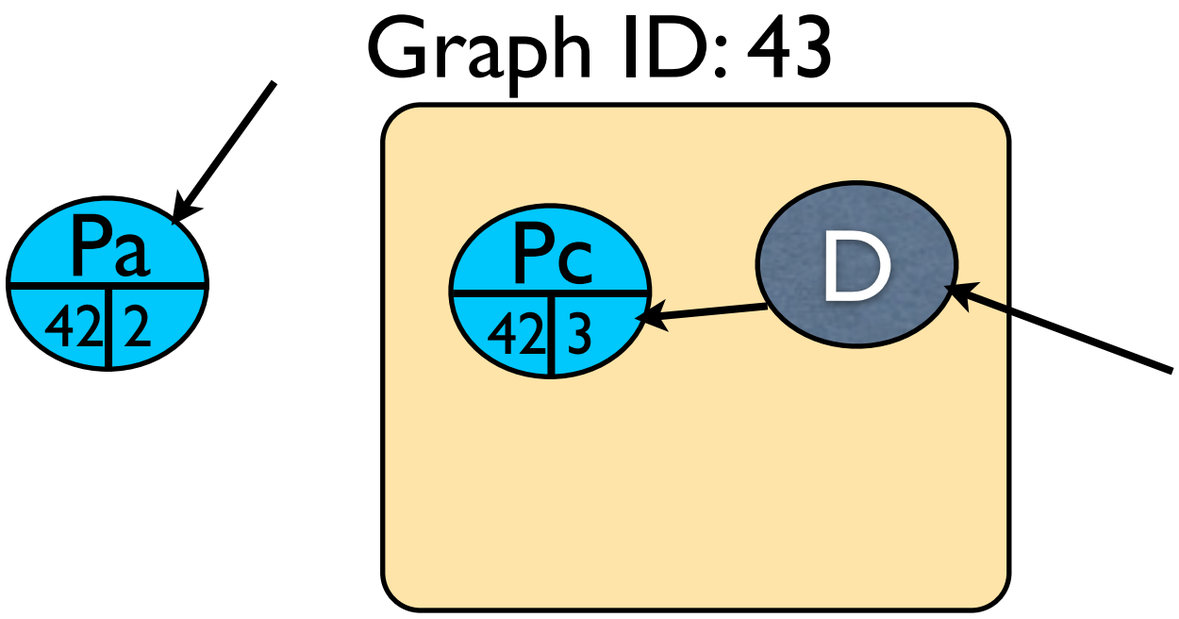
42.swap

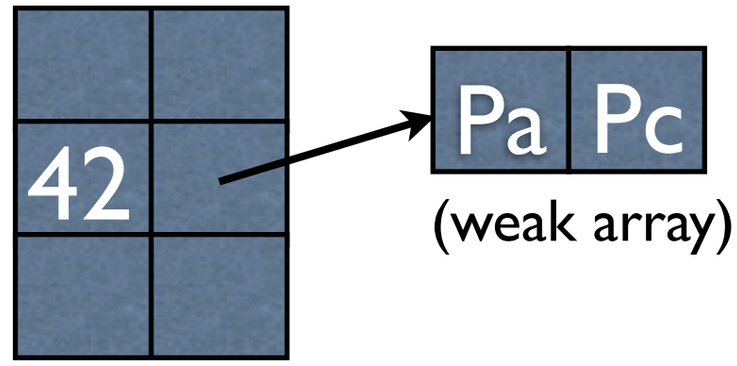
1) Assign graph ID

2) Serialize the object graph

Intersections...

Primary memory





GraphTable

Secondary memory

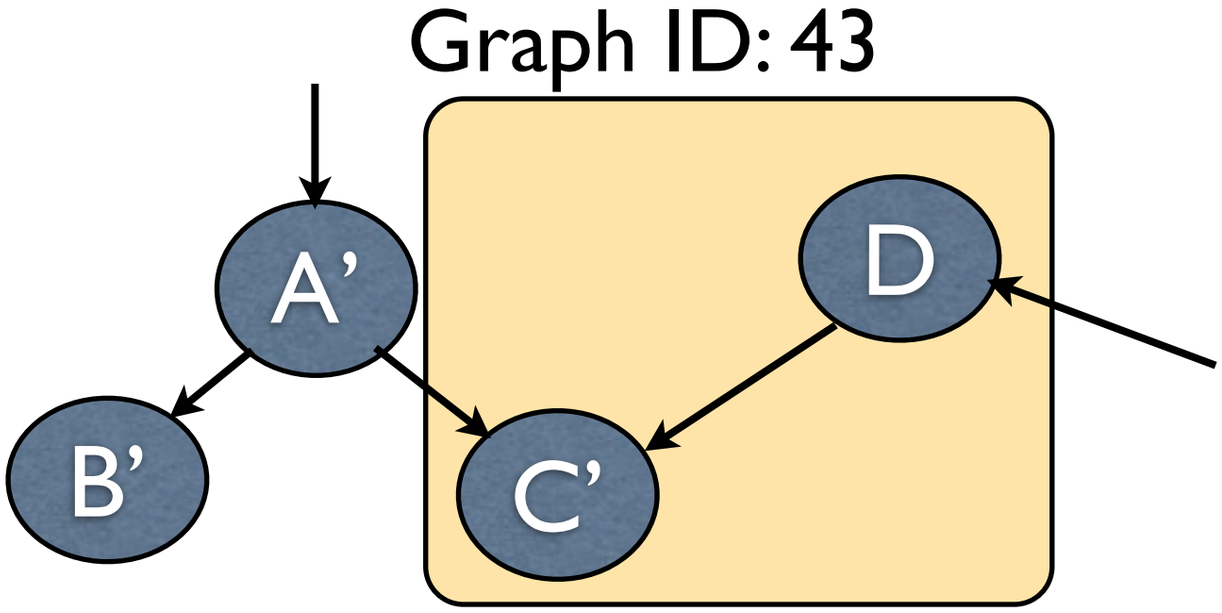
42.swap

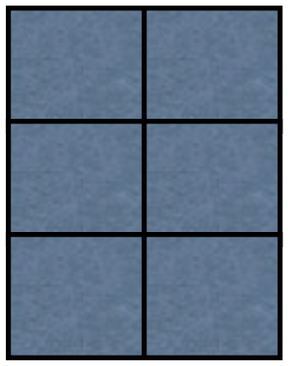
1) Assign graph ID

2) Serialize the object graph

Intersections...

Primary memory





GraphTable

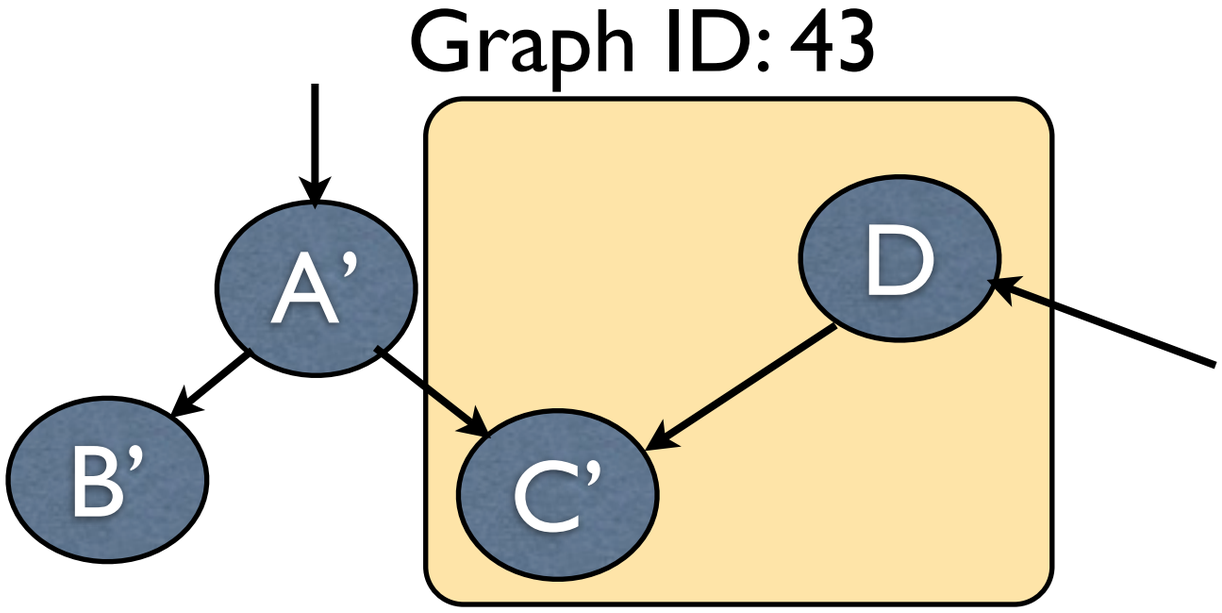
Secondary memory

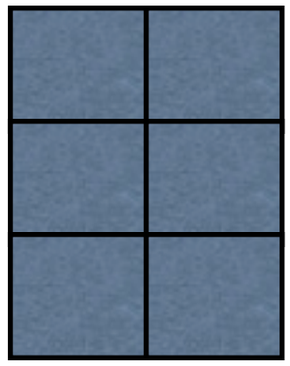
1) Assign graph ID

2) Serialize the object graph

Intersections...

Primary memory





GraphTable

Secondary memory

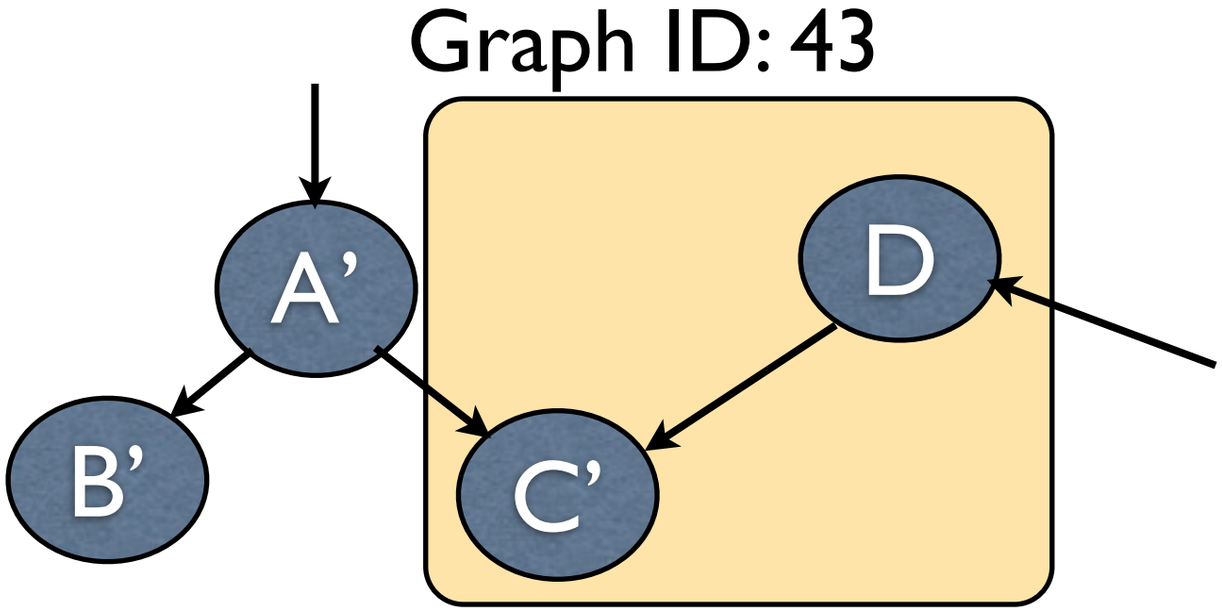
43.swap

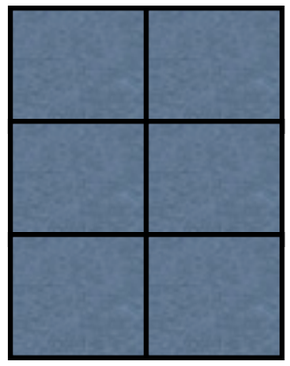
1) Assign graph ID

2) Serialize the object graph

Intersections...

Primary memory





GraphTable

Secondary memory

43.swap

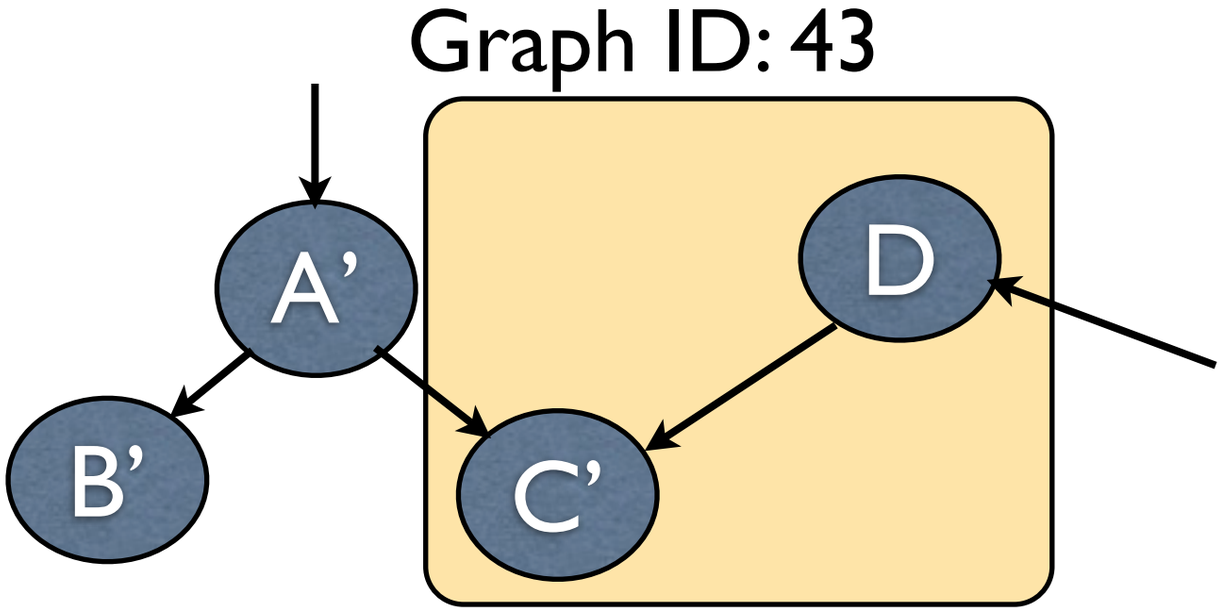
1) Assign graph ID

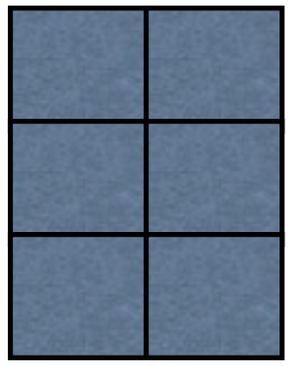
2) Serialize the object graph

...) as usually...

Intersections...

Primary memory





GraphTable

Secondary memory

1) Assign graph ID

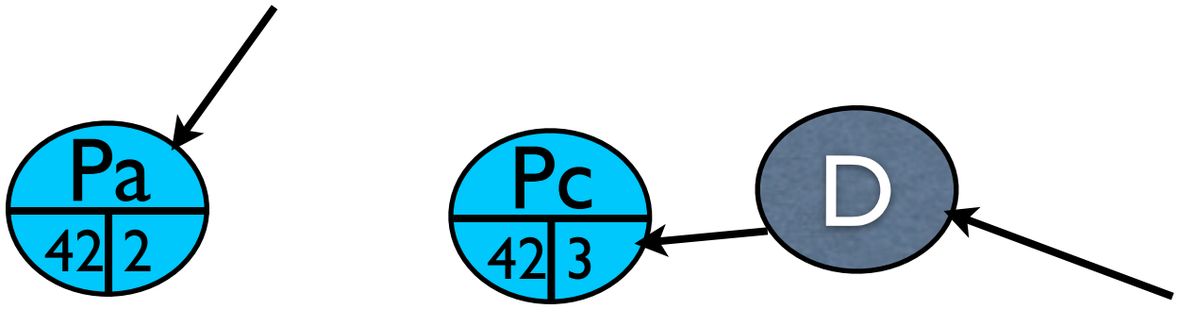
2) Serialize the object graph

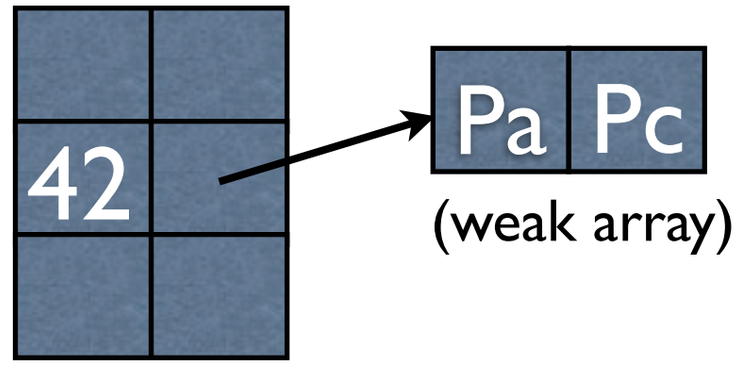
...) as usually...

We don’t want to swap in a graph while swapping out another one

Intersections...

Primary memory





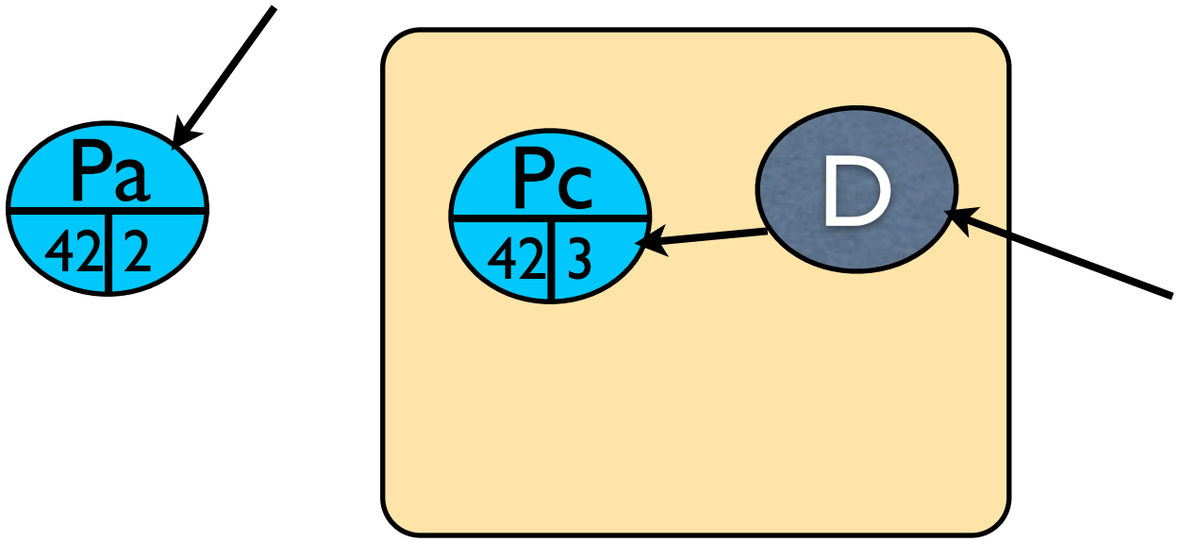
GraphTable

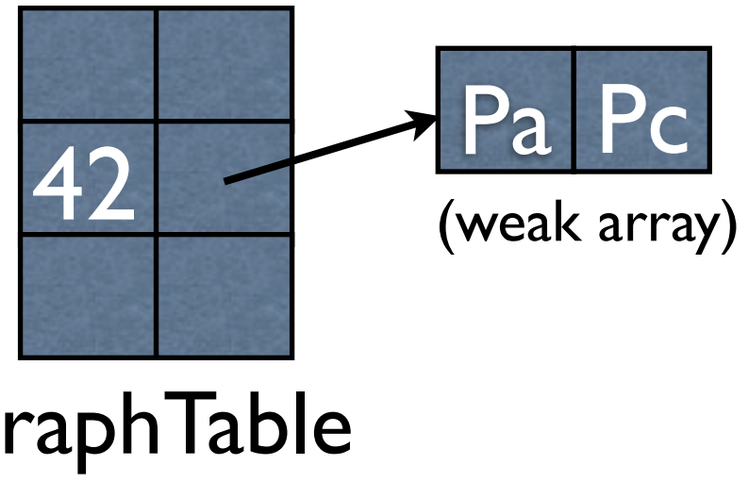
Secondary memory

42.swap

Intersections...

Primary memory





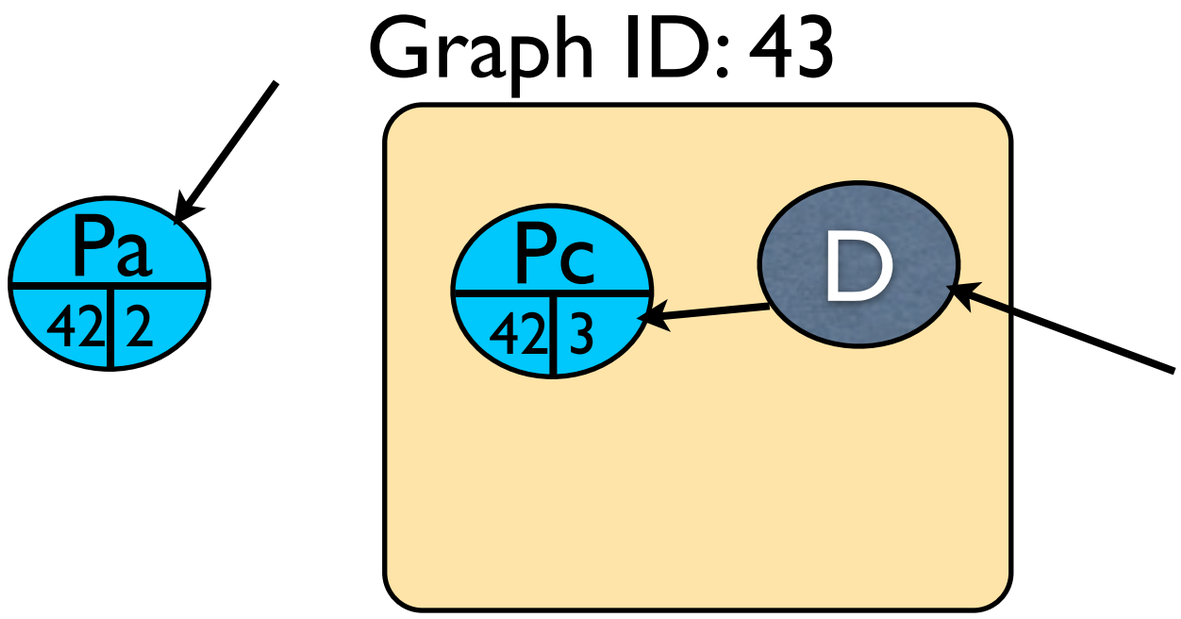
GraphTable

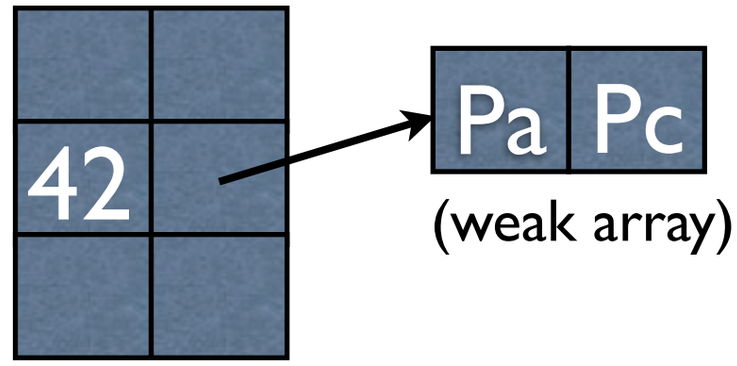
Secondary memory

42.swap

Intersections...

Primary memory





GraphTable

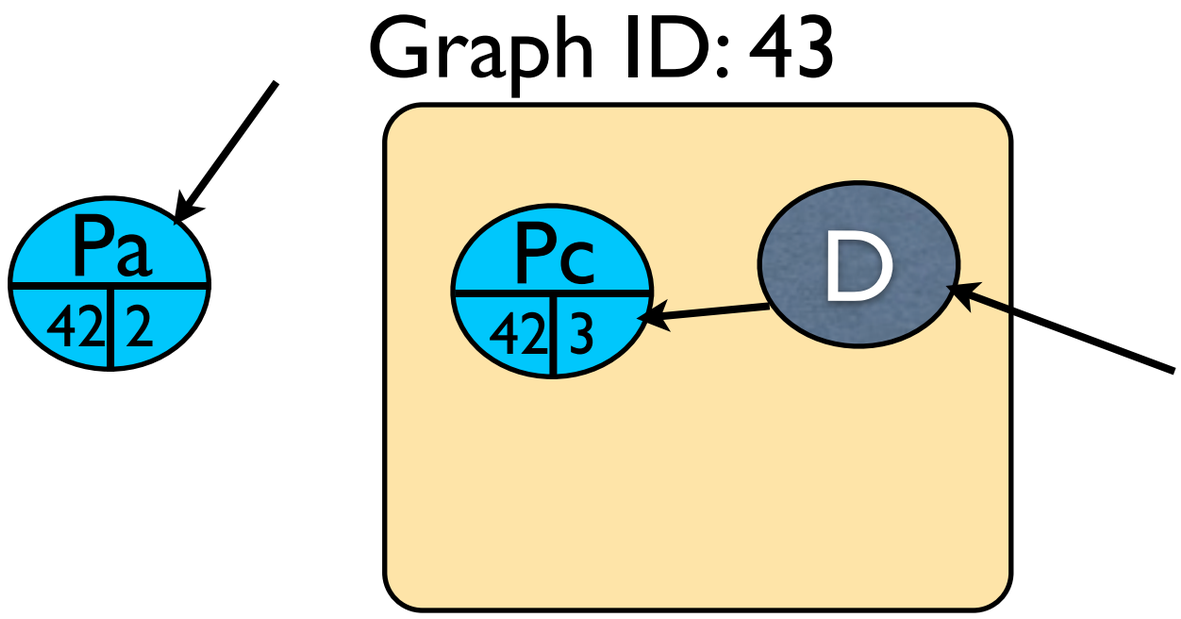
Secondary memory

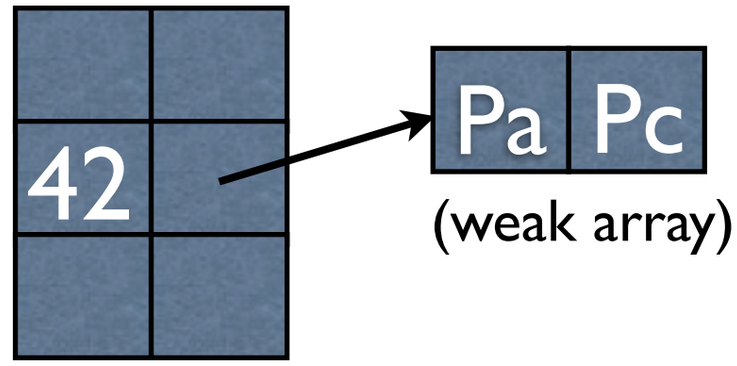
42.swap

1) Assign graph ID

Intersections...

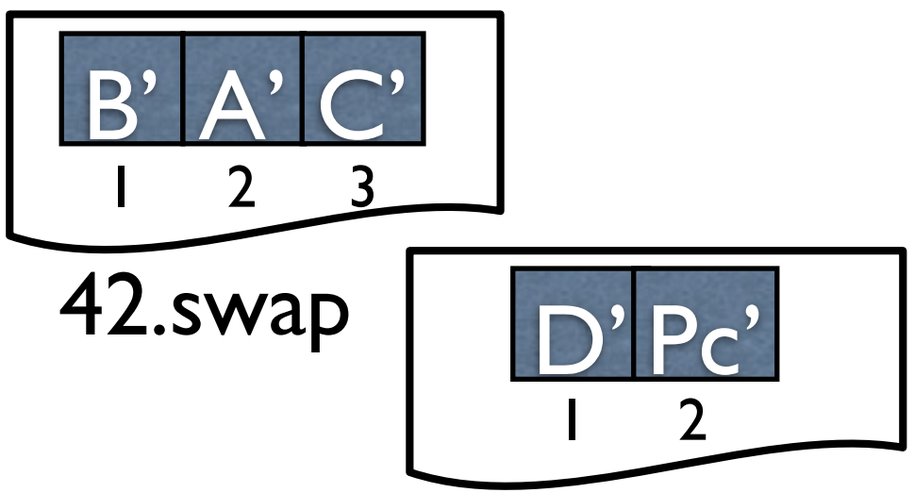
Primary memory





GraphTable

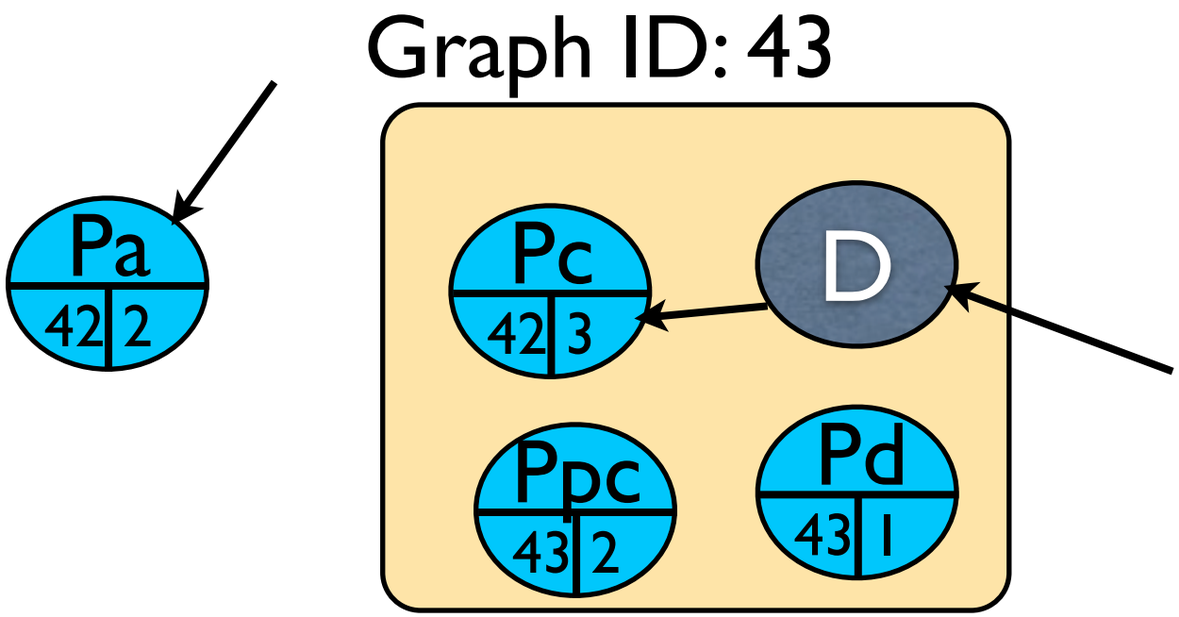
Secondary memory

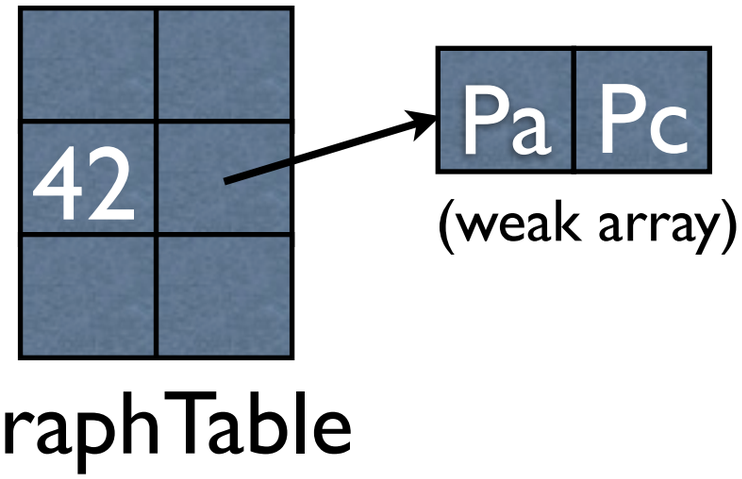


1) Assign graph ID
2) Serialize the object graph (customize serializer)

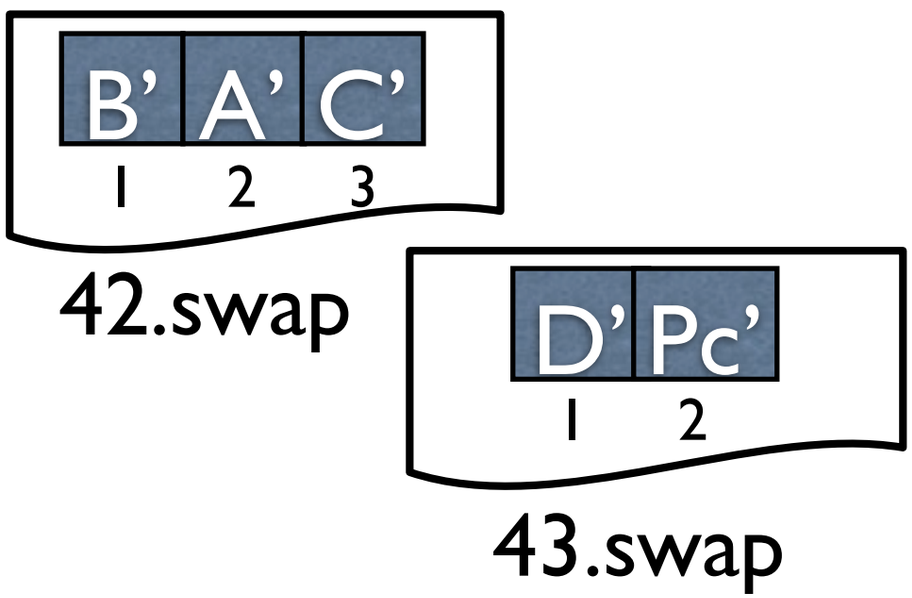
Intersections...

Primary memory





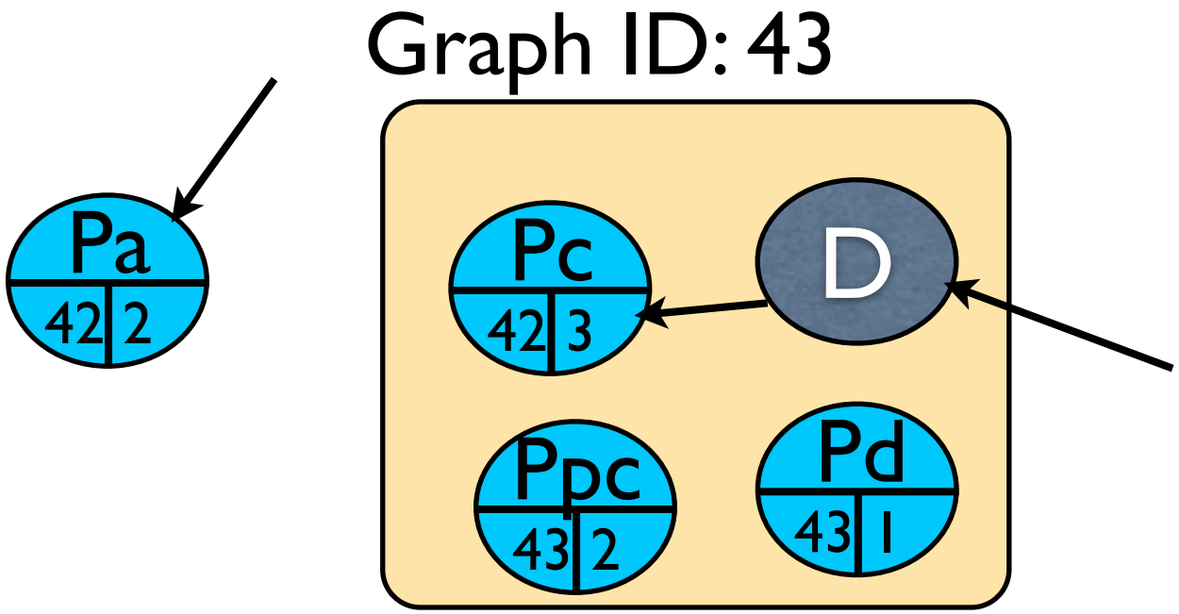
Secondary memory

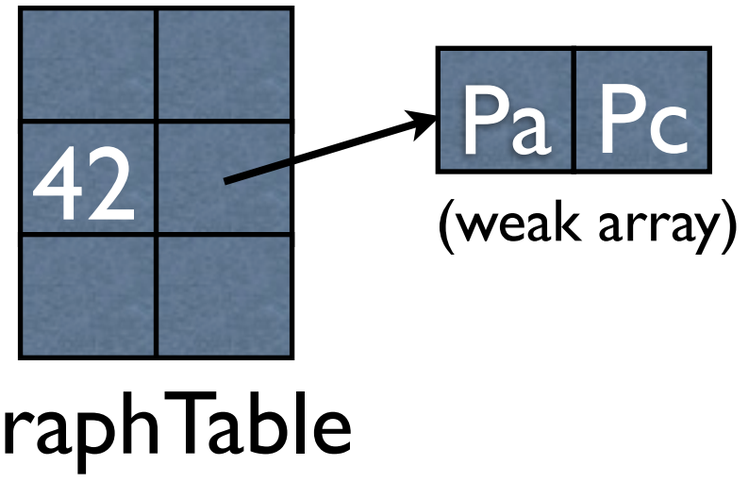


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies

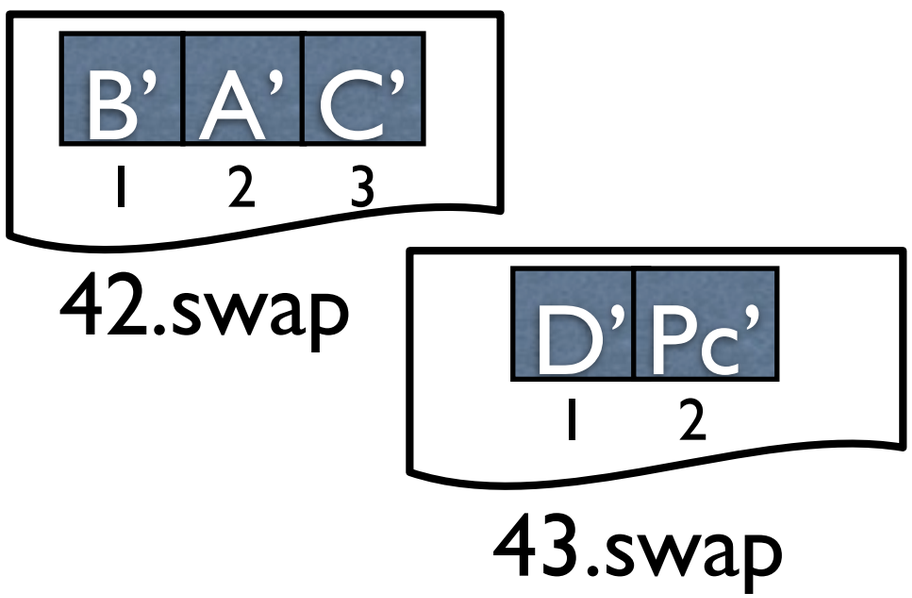
Intersections...

Primary memory





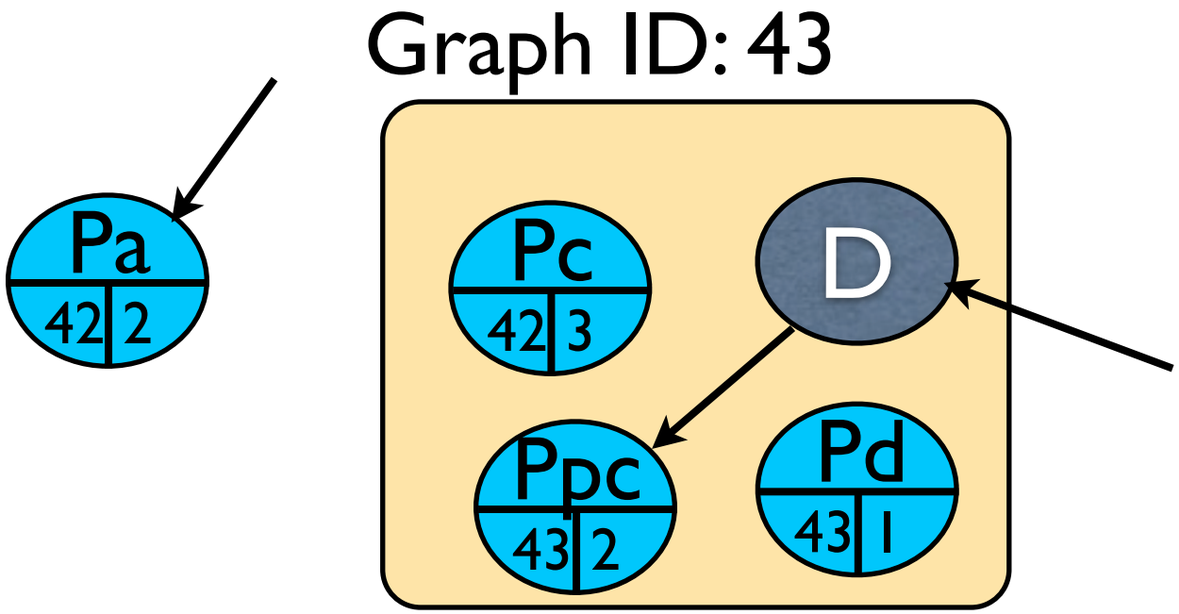
Secondary memory

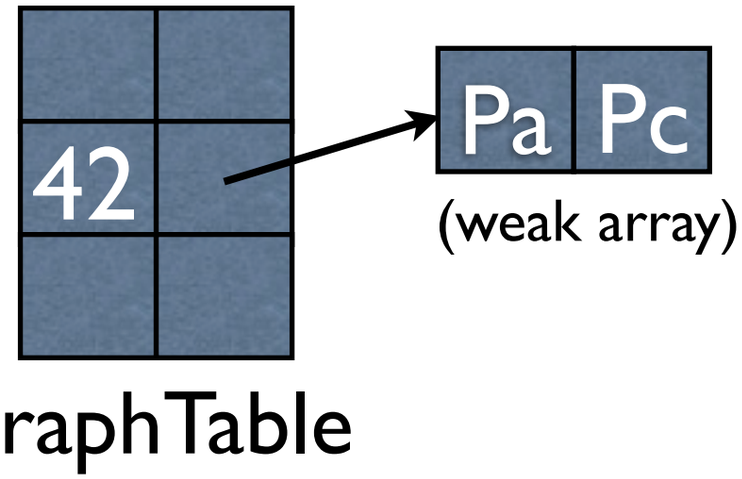


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies

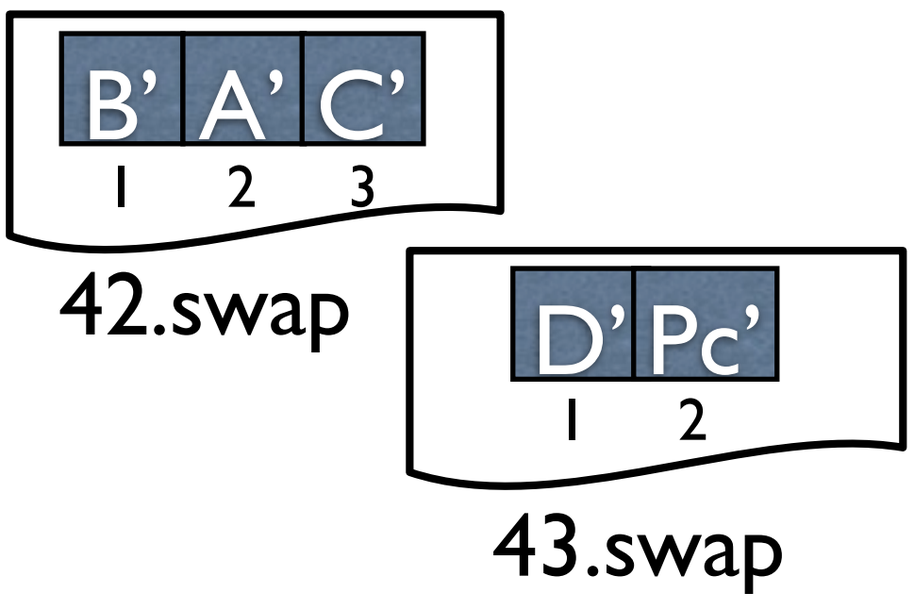
Intersections...

Primary memory





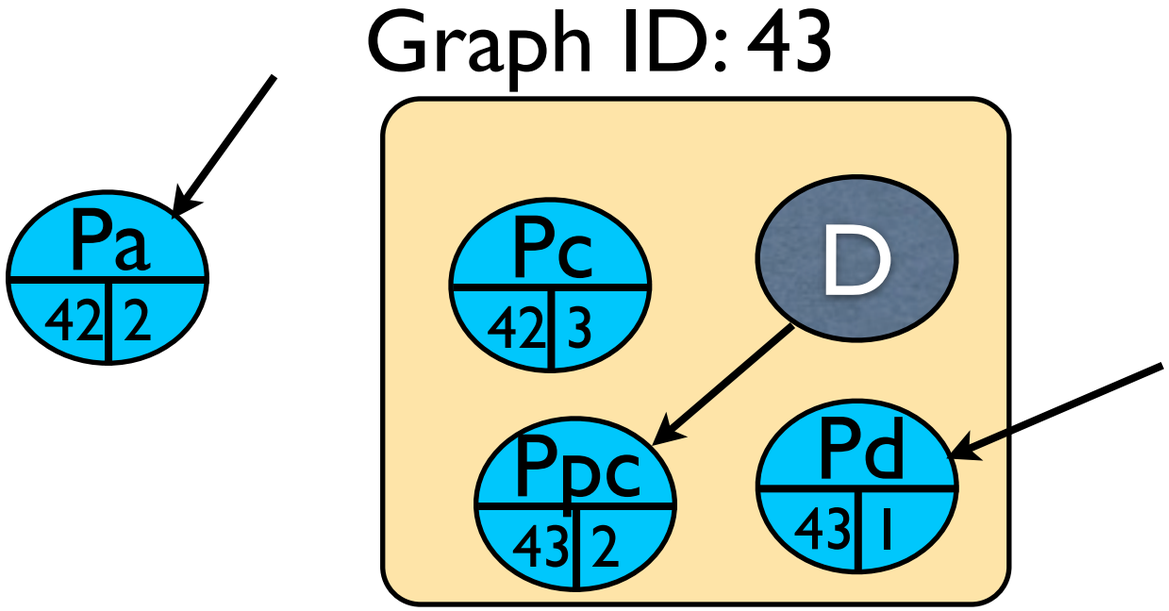
Secondary memory

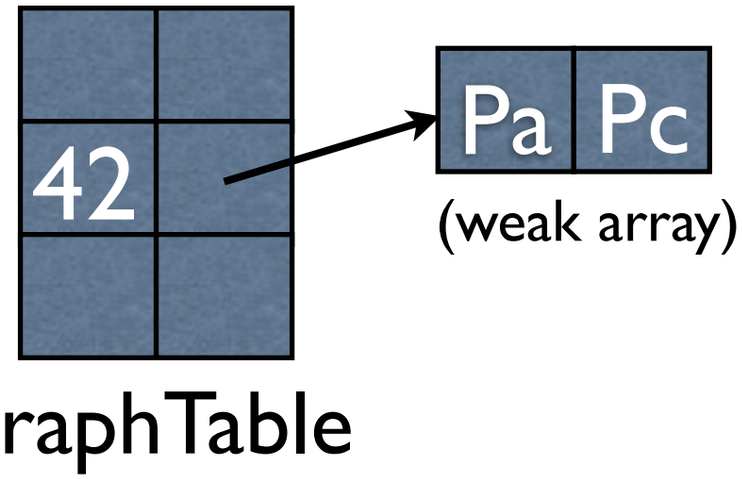


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies

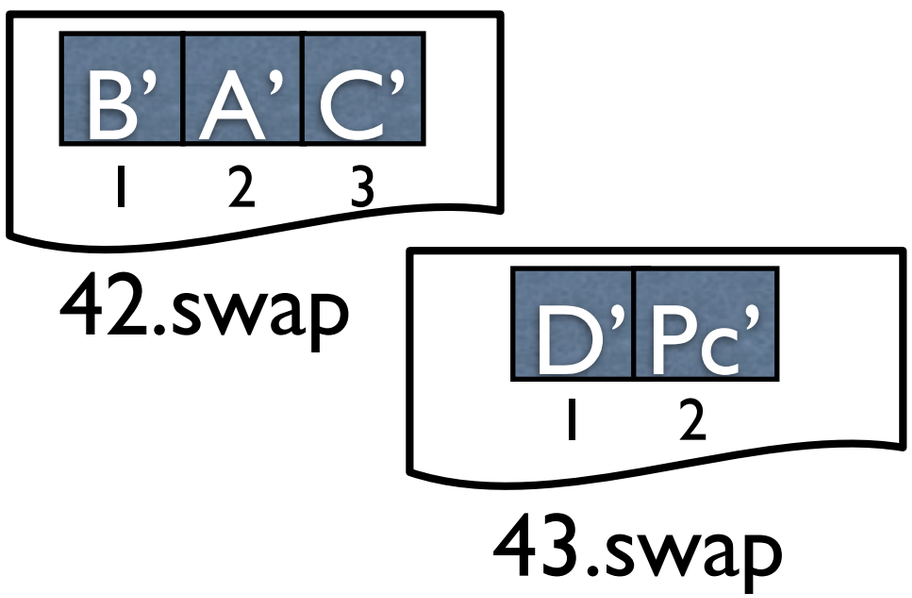
Intersections...

Primary memory





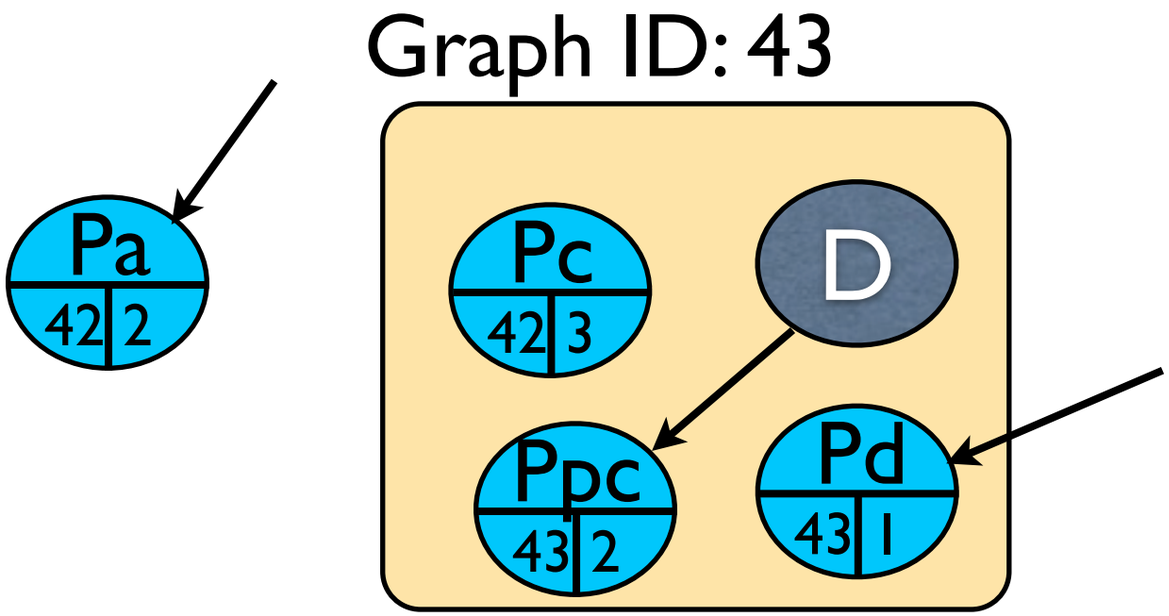
Secondary memory

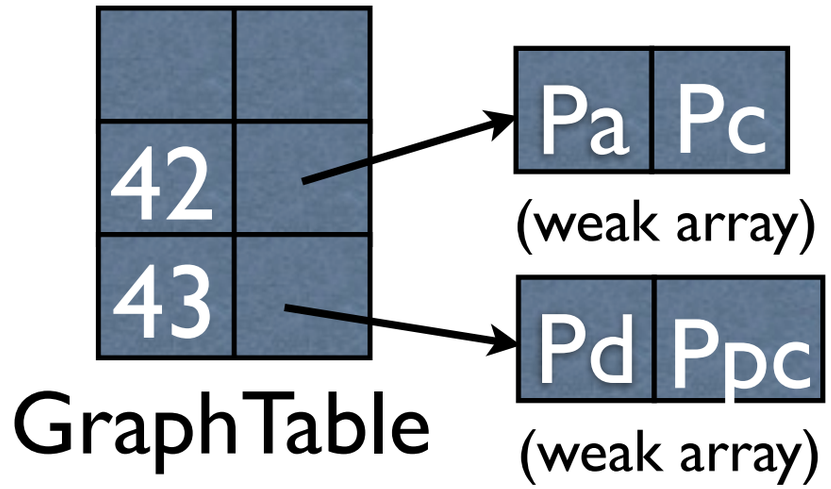


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies

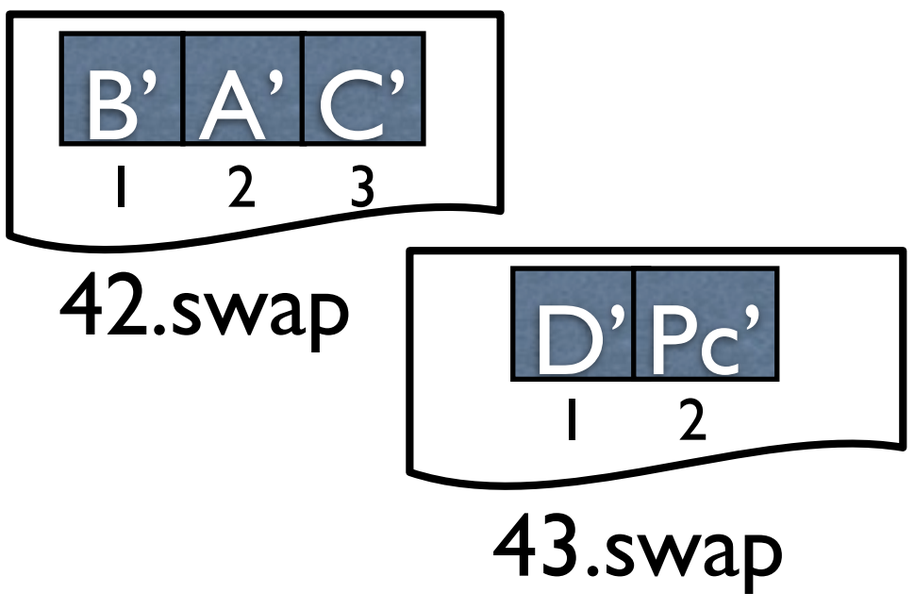
Intersections...

Primary memory





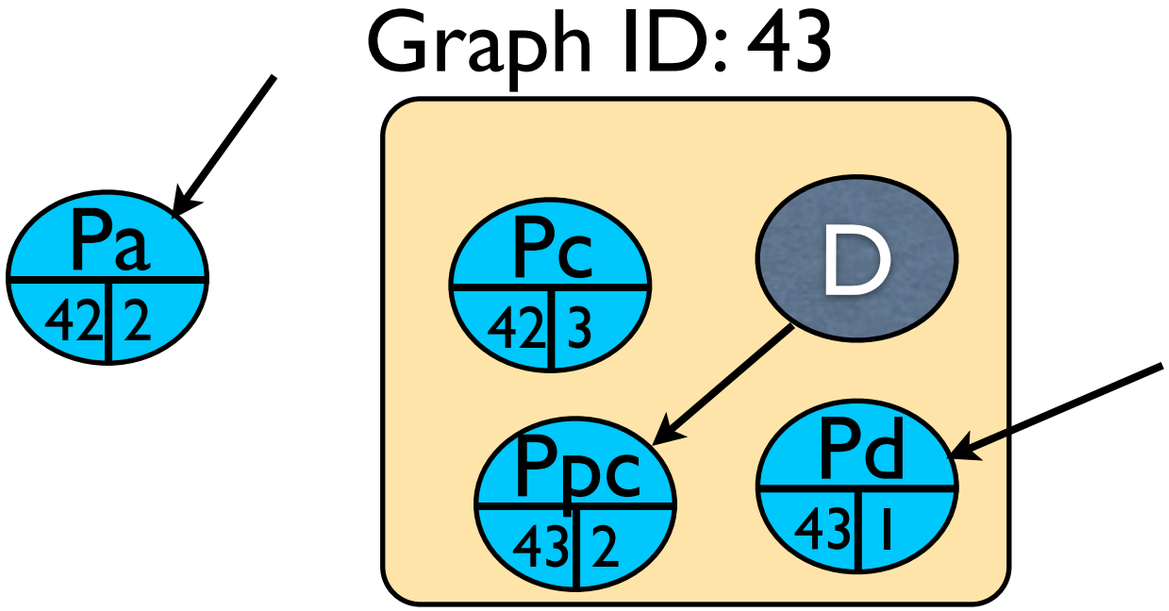
Secondary memory

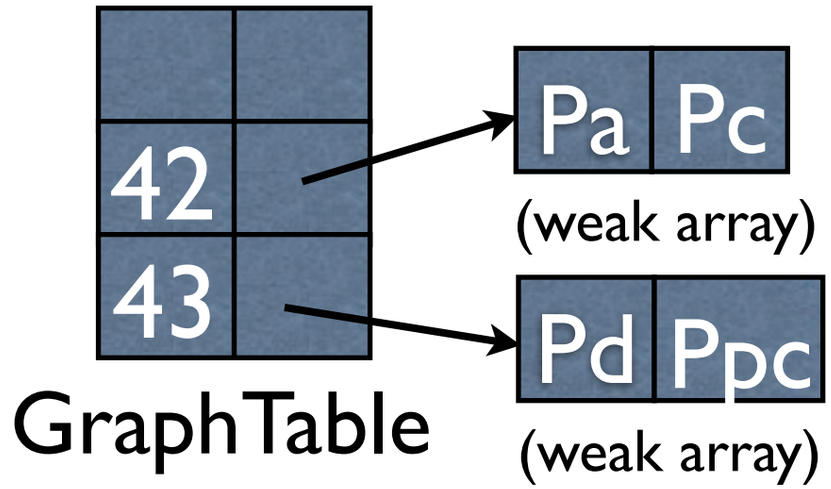


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable

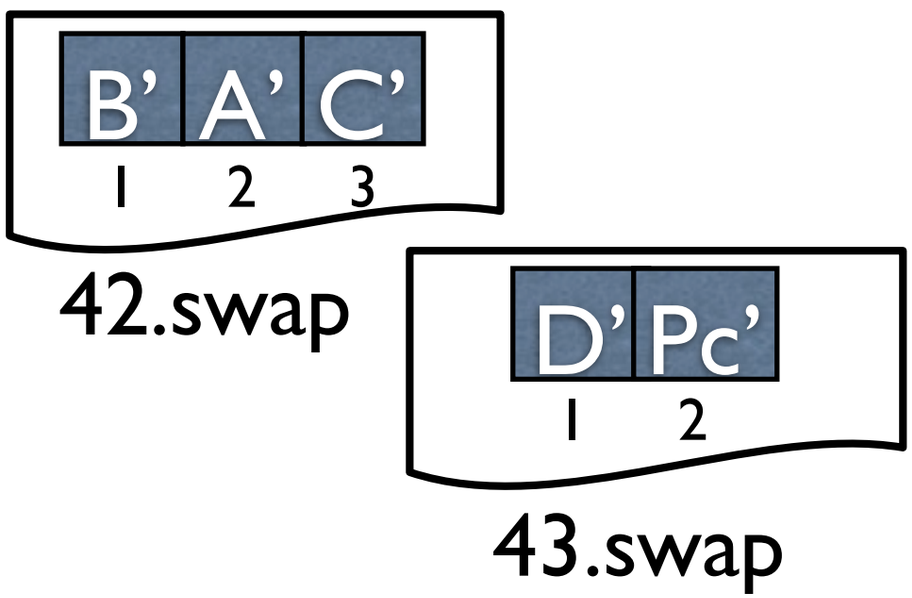
Intersections...

Primary memory





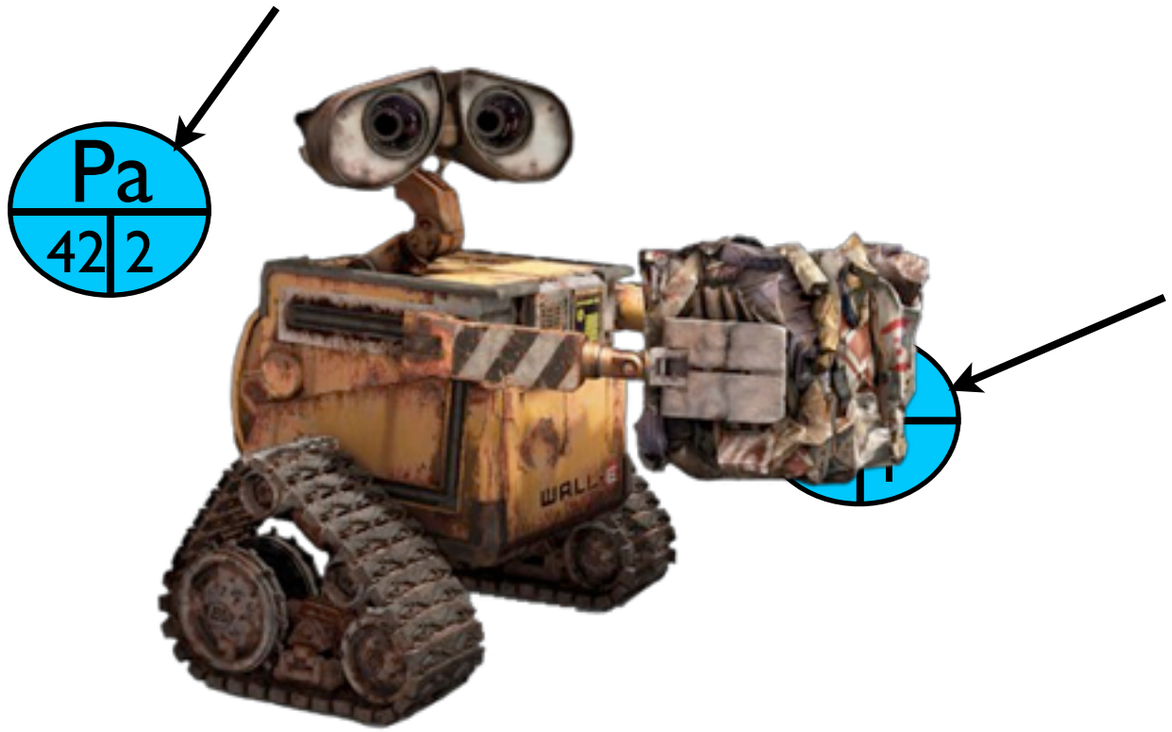
Secondary memory

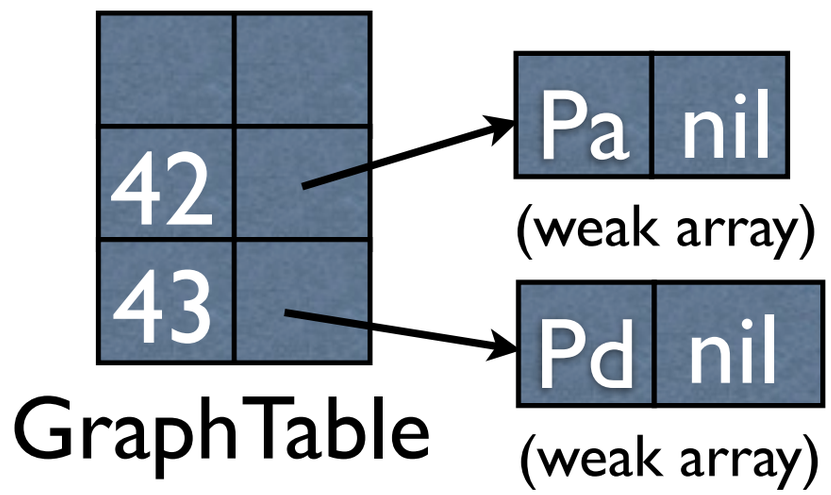


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs

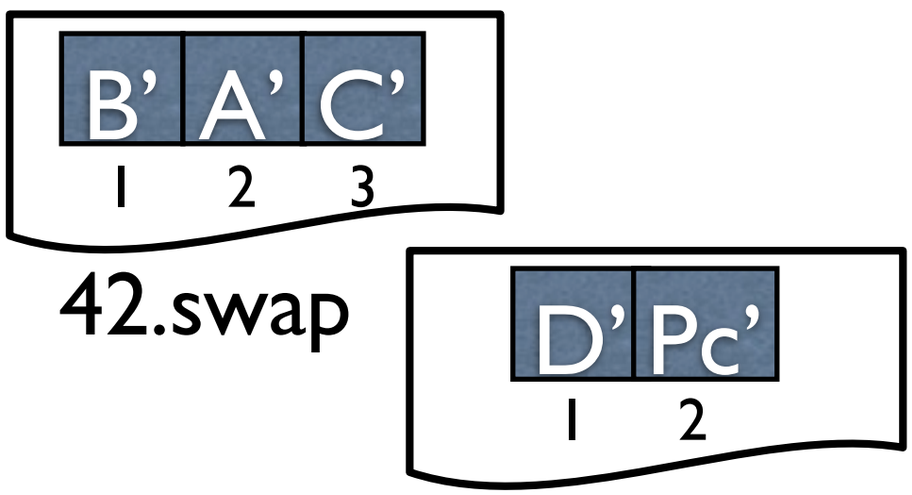
Intersections...

Primary memory





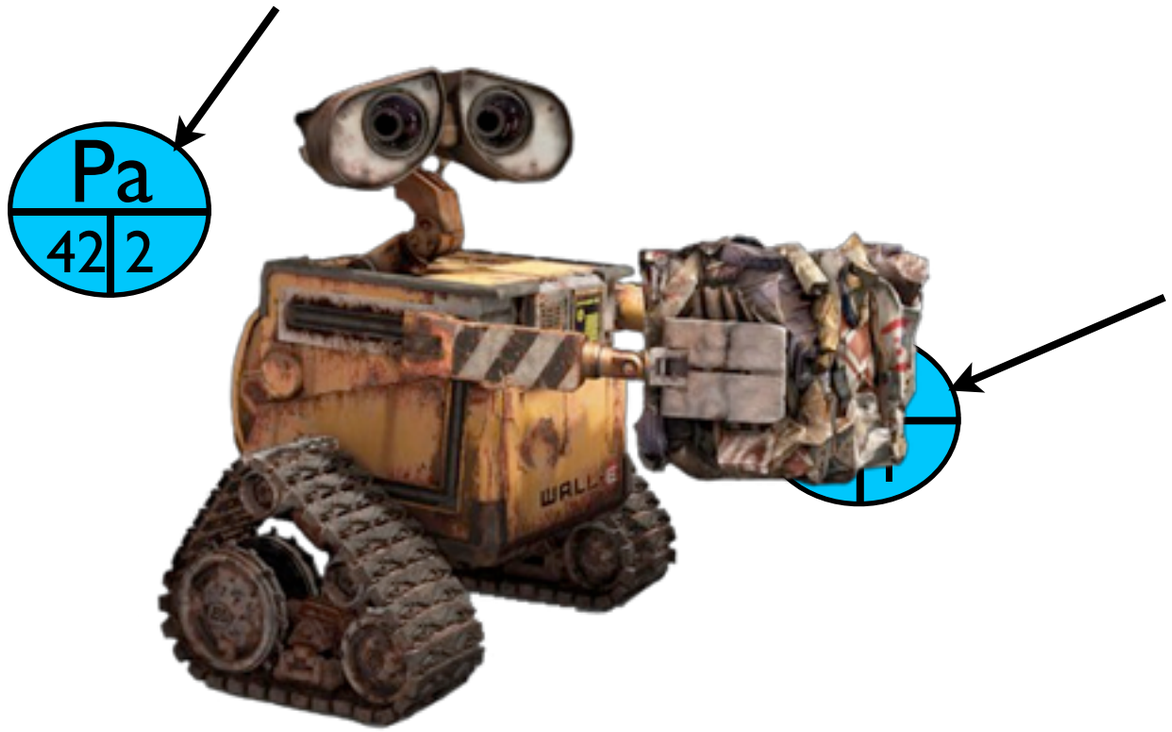
Secondary memory

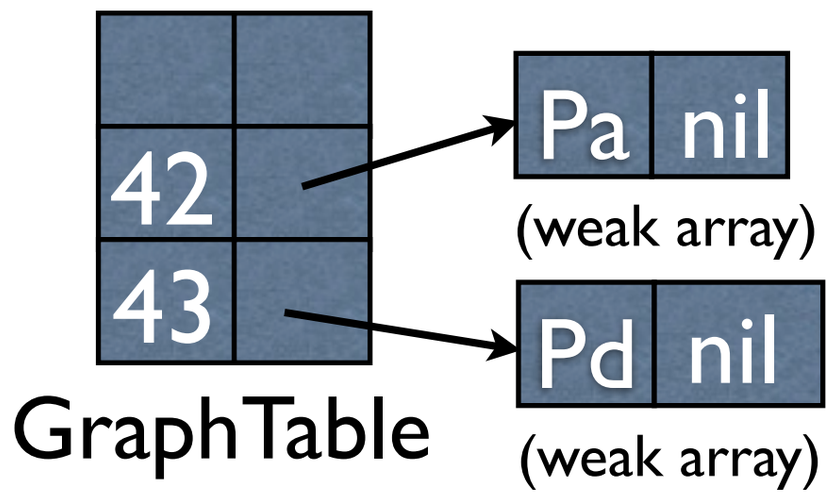


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs

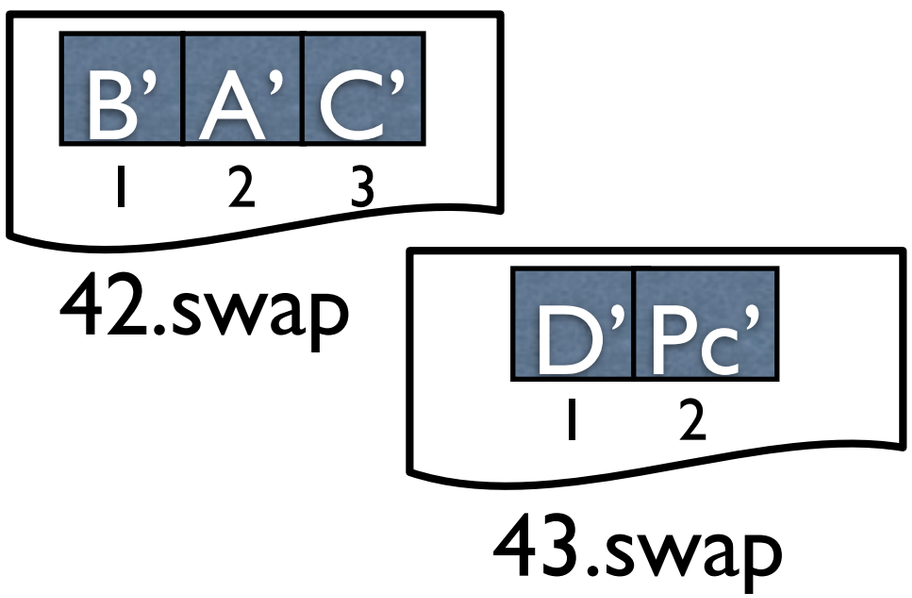
Intersections...

Primary memory





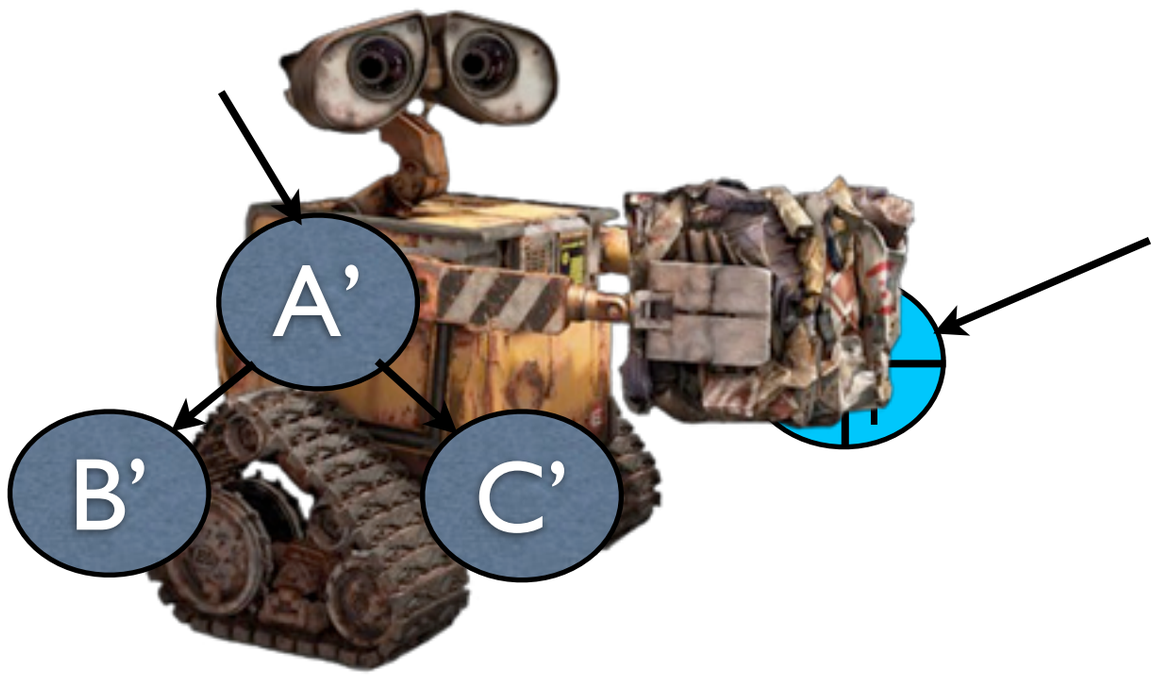
Secondary memory

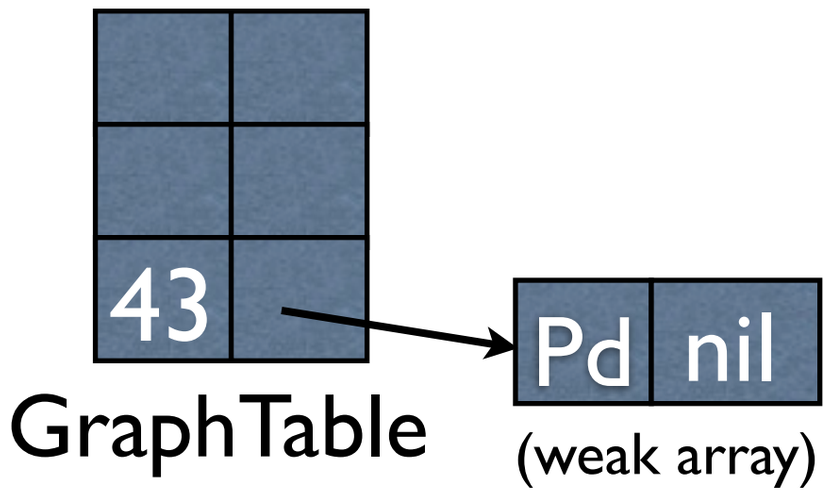


1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Swap in graph 42

Intersections...

Primary memory





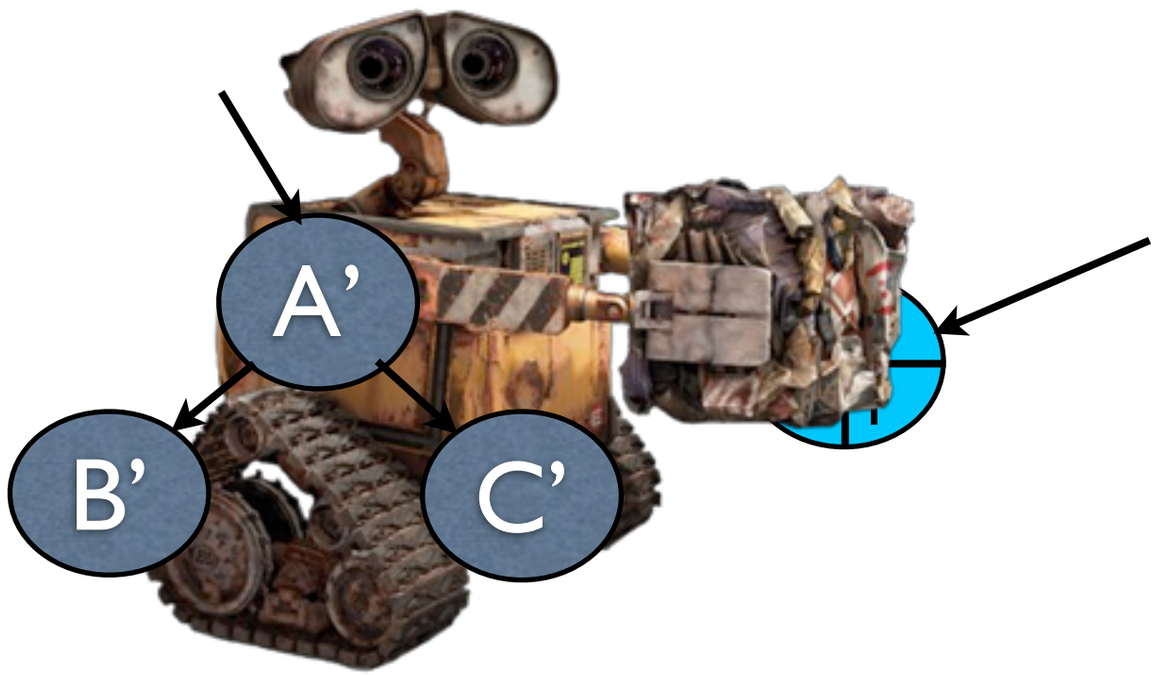
Secondary memory

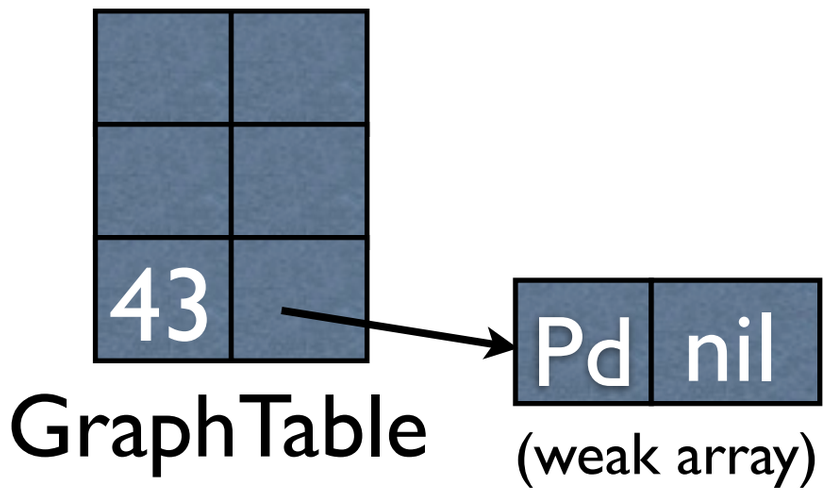
43.swap

1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Swap in graph 42

Intersections...

Primary memory





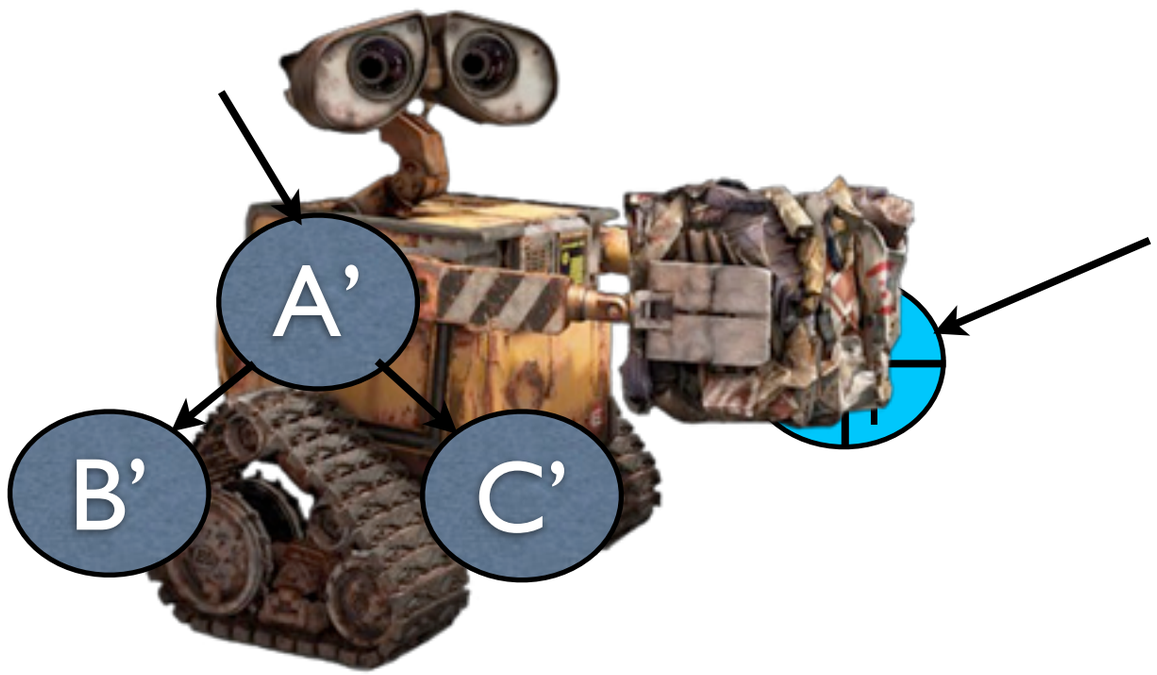
Secondary memory

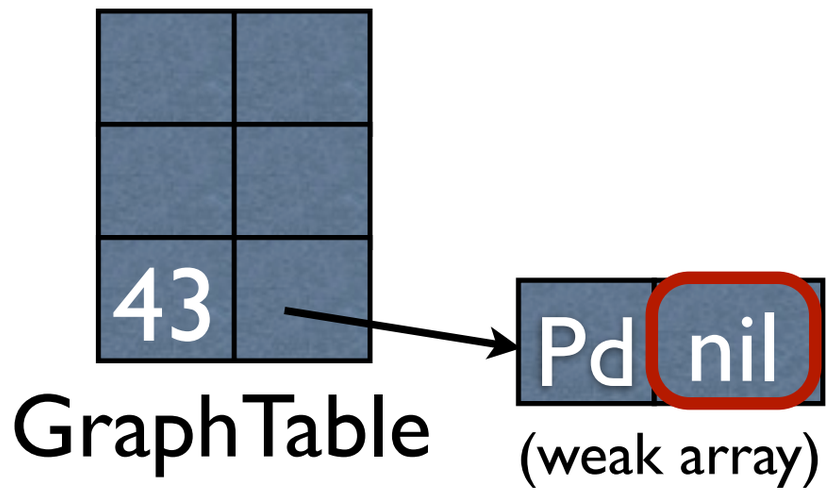
43.swap

1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Swap in graph 42
8) Swap in graph 43...

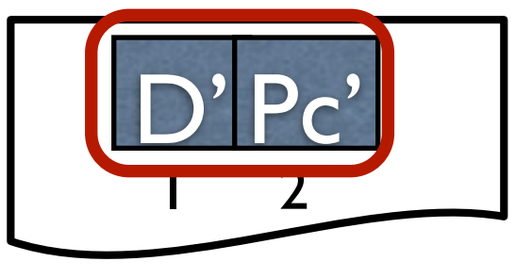
Intersections...

Primary memory





Secondary memory

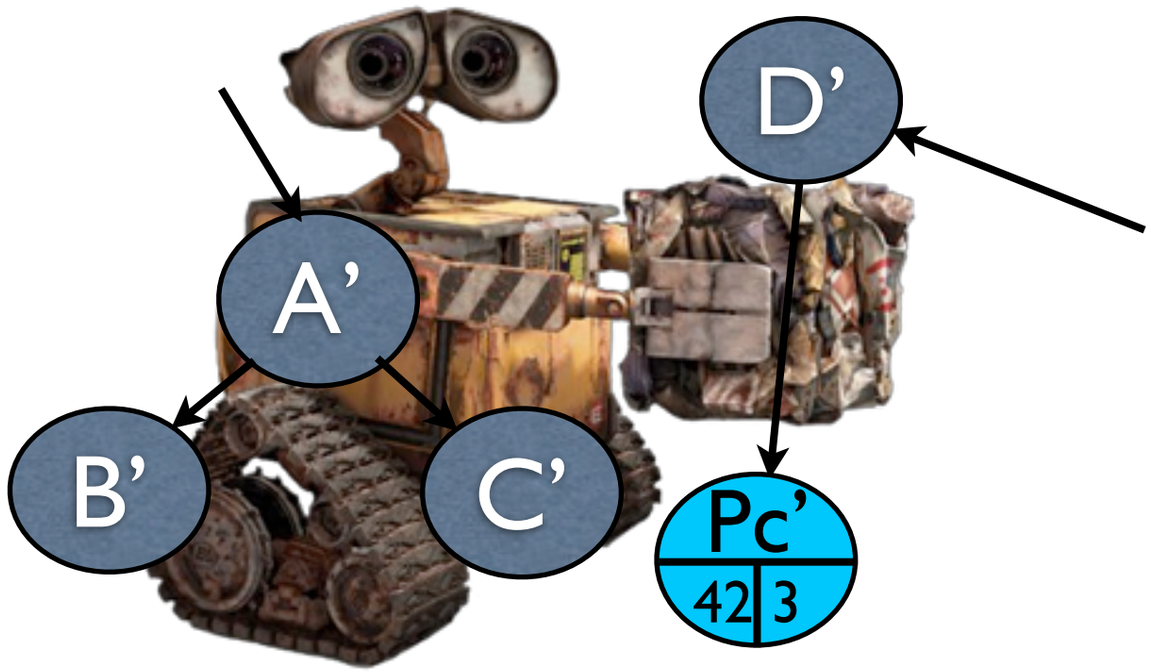


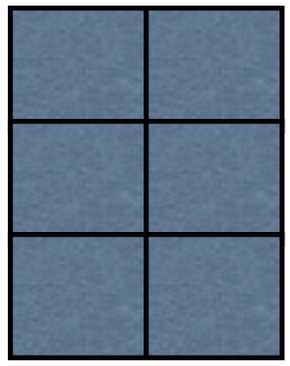
43.swap

1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Swap in graph 42
8) Swap in graph 43...

Primary memory

Intersections...





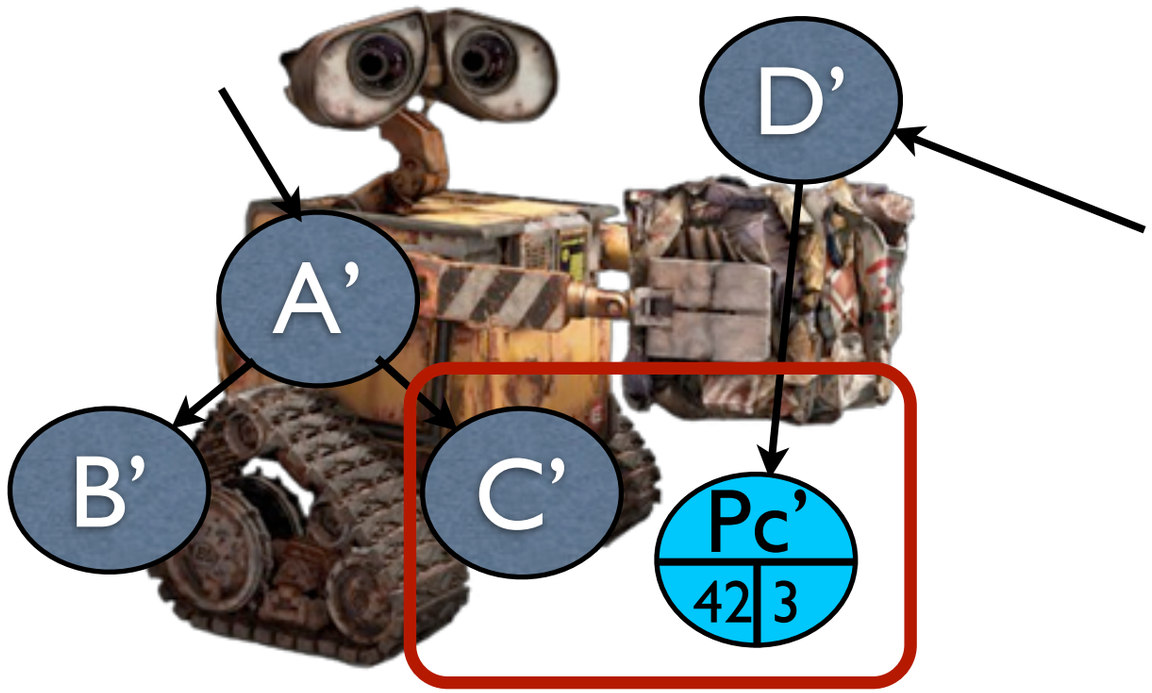
GraphTable

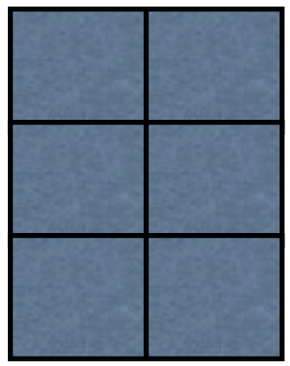
Secondary memory

1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Swap in graph 42
8) Swap in graph 43...

Intersections...

Primary memory





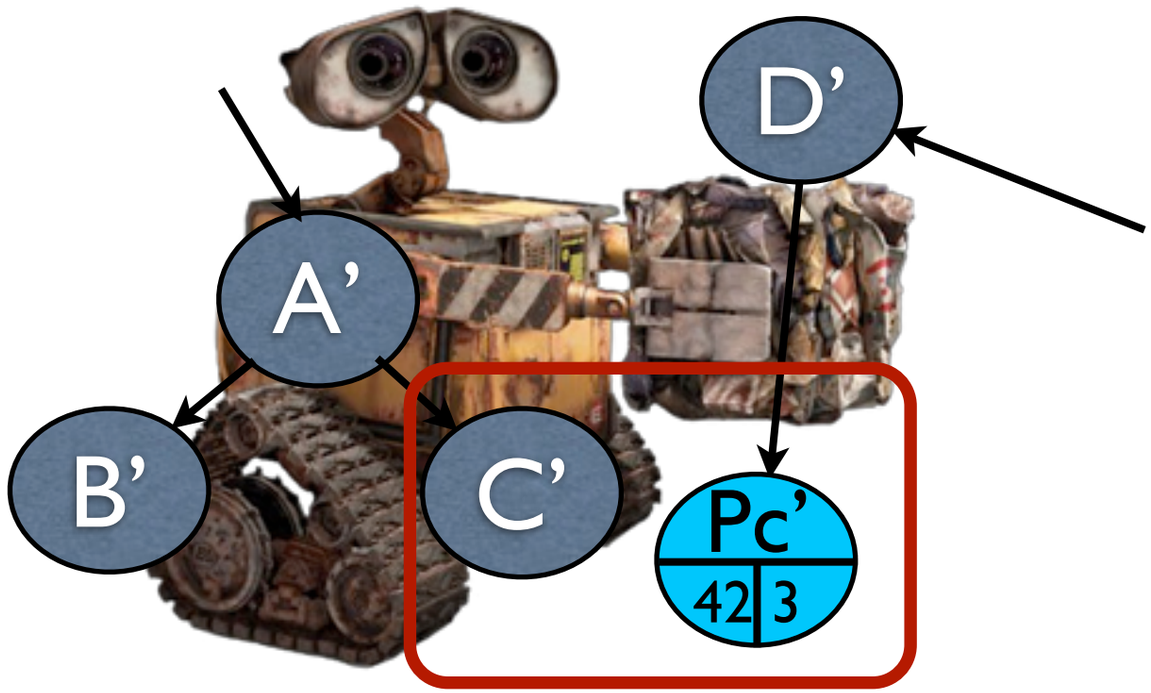
GraphTable

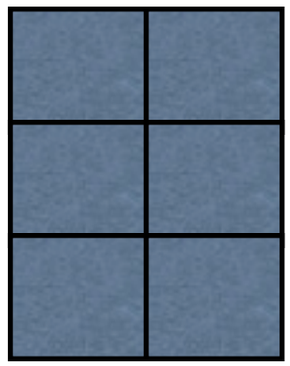
Secondary memory

1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies
5) Update GraphTable
6) GC runs
7) Swap in graph 42
8) Swap in graph 43...

Intersections...

Primary memory





GraphTable

Secondary memory

1) Assign graph ID
2) Serialize the object graph (customize serializer)
3) Create and associate proxies
4) Replace original objects with proxies

INCORRECT!!!

Graph intersections

- Avoiding proxies for proxies (Ppc) does not solve the problem either.
- Several scenarios for swapping in swapped intersecting graphs.

Graph intersections

- Avoiding proxies for proxies (Ppc) does not solve the problem either.
- Several scenarios for swapping in swapped intersecting graphs.

Marea solves all these scenarios and challenges.

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

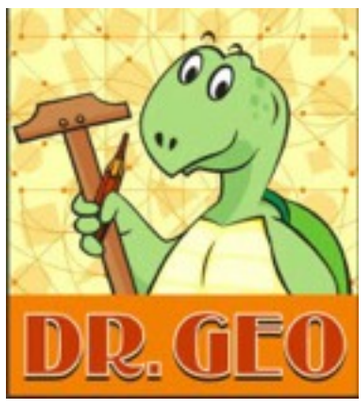
Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Experiment

(web app + CMS)

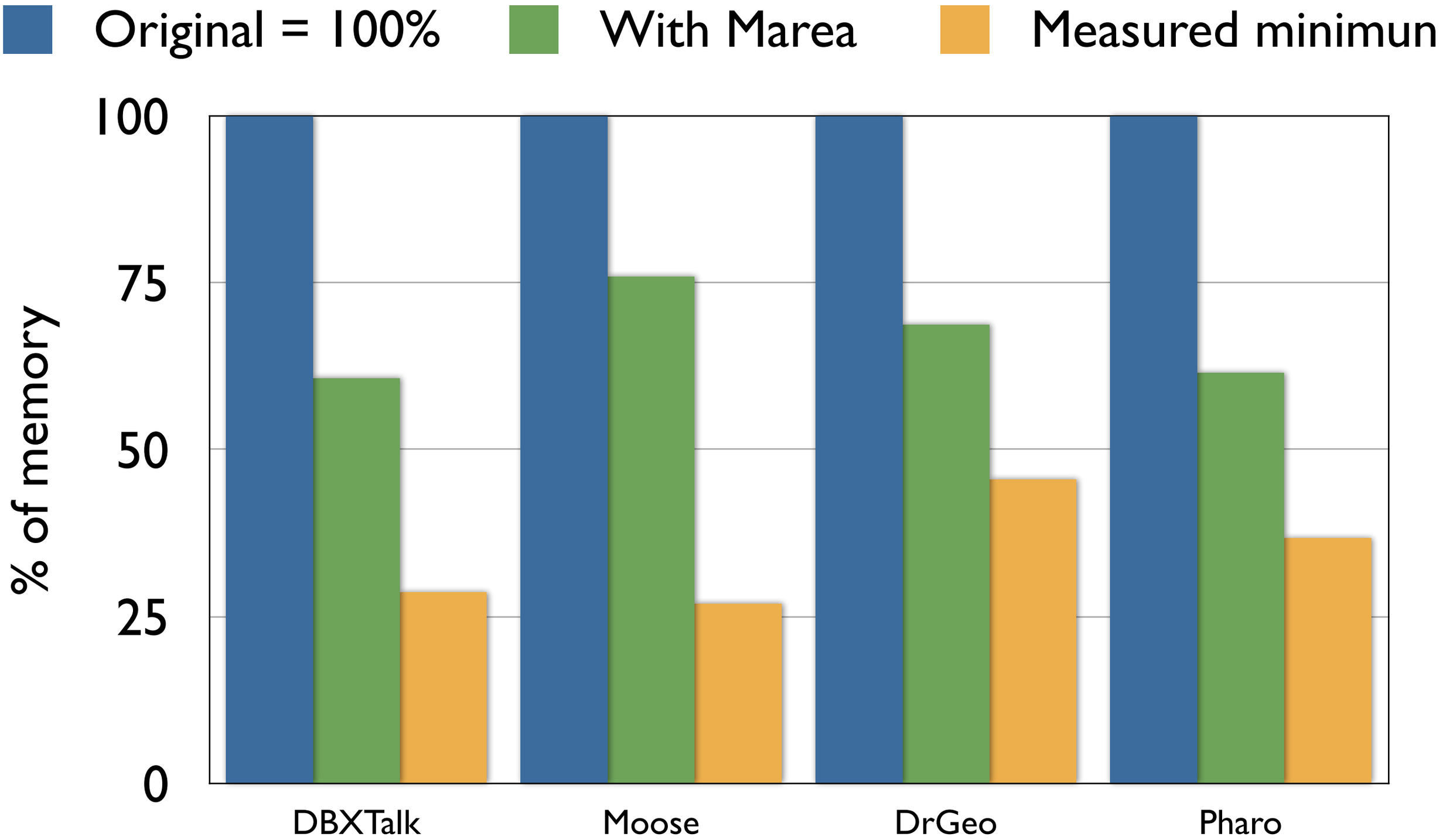
(standalone + lots of tools + large)
(174 pack. 1364 class. 121010 LOC)

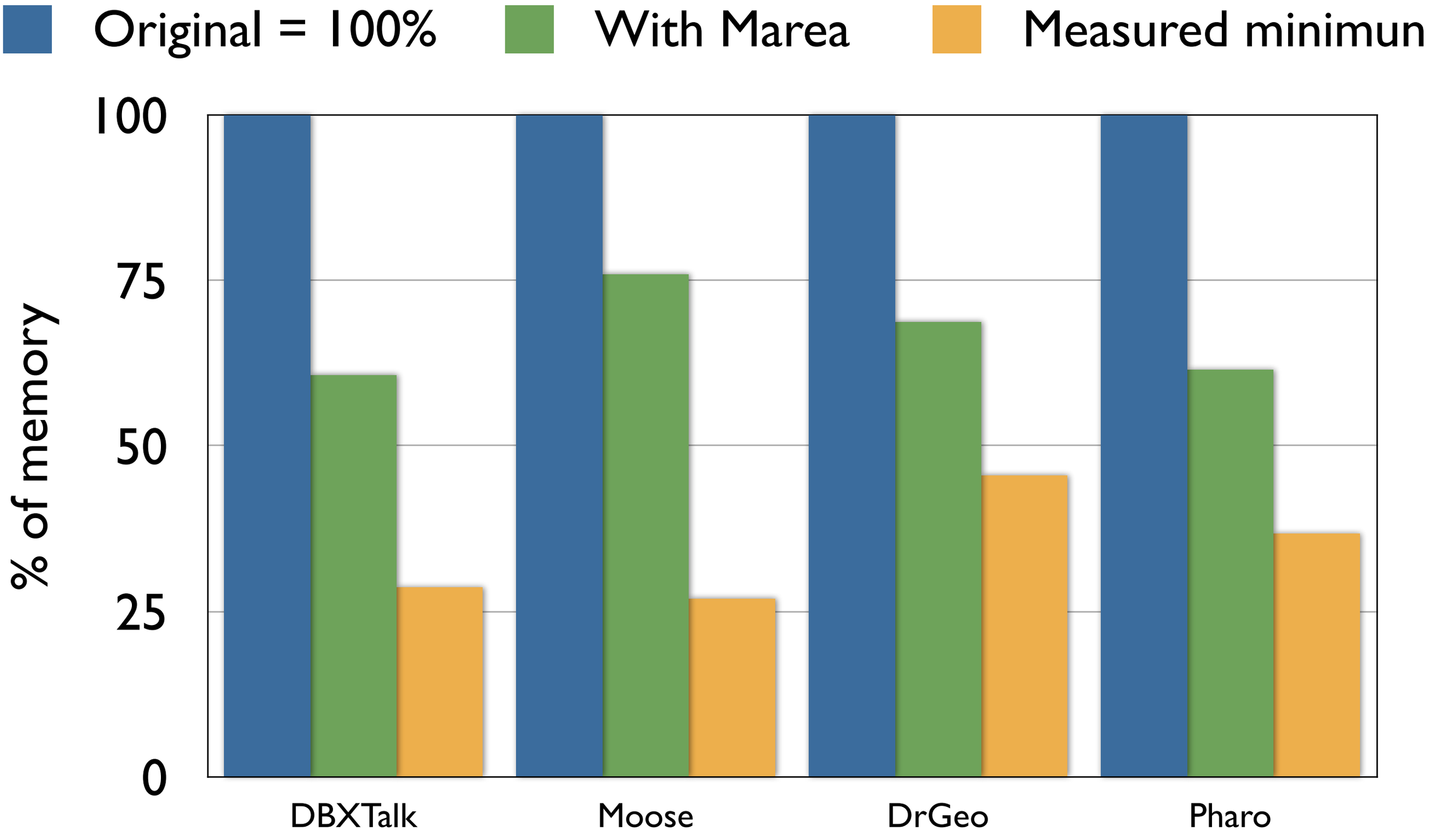


(mobile app)

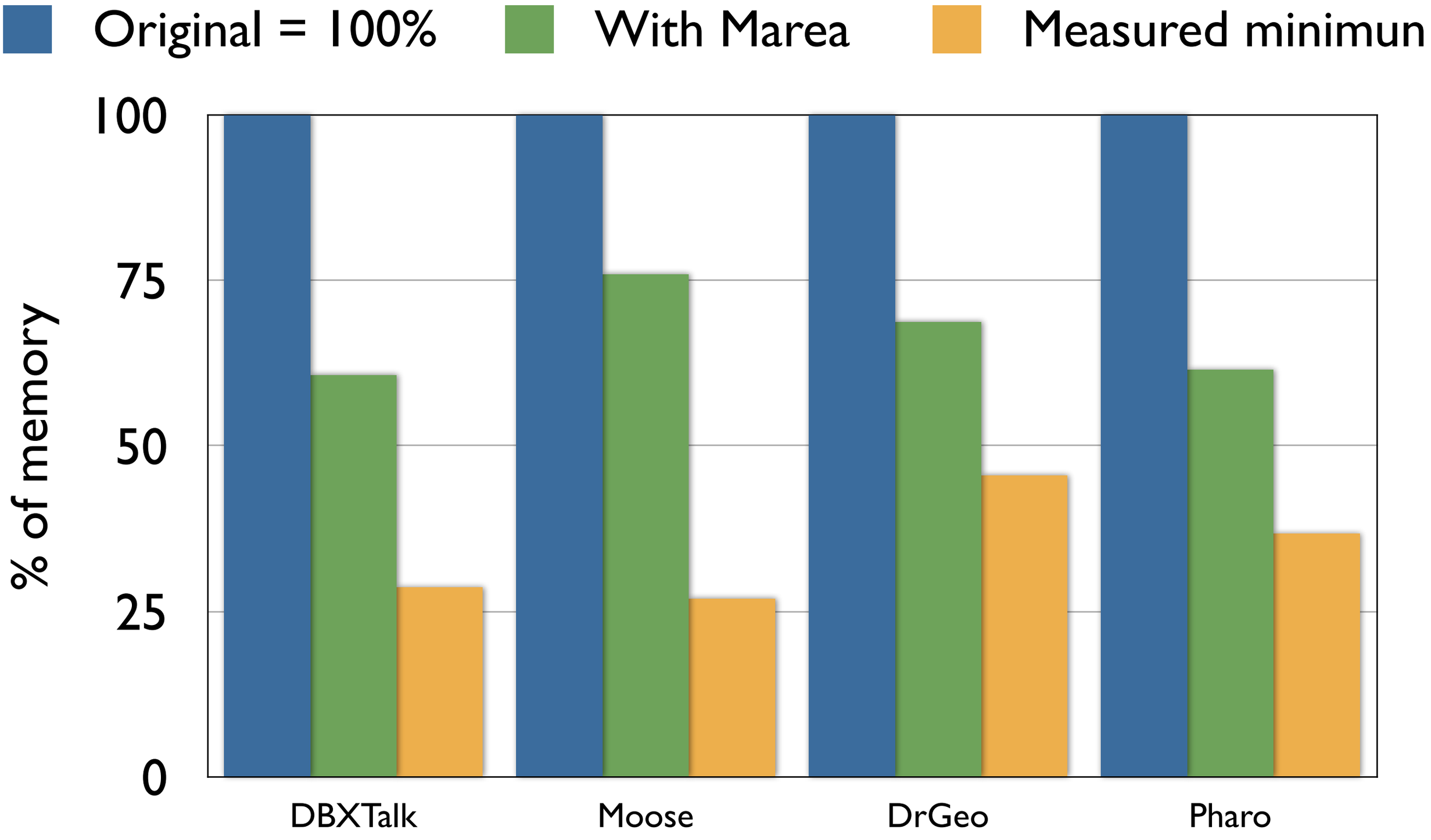
(infrastructure)

1. We swapped out most classes with their instances.
2. We navigated, used and tested each application
    1. Swapped in all needed graphs for typical uses.
    2. Tests also for correctness.
3. We measured differences.


Original = 100%
With Marea
Measured minimun
100
75
50
25
0
% of memory
DBXTalk
Moose
DrGeo
Pharo



Marea released between 25%
and 40% of the used memory.



Marea released between 25% and 40% of the used memory.

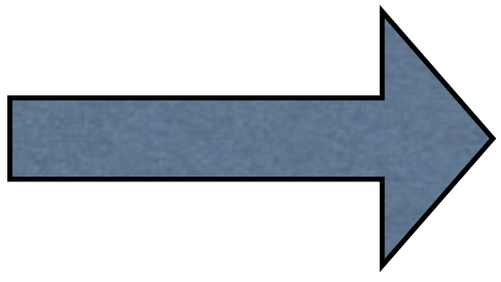
There is still room for improvement.

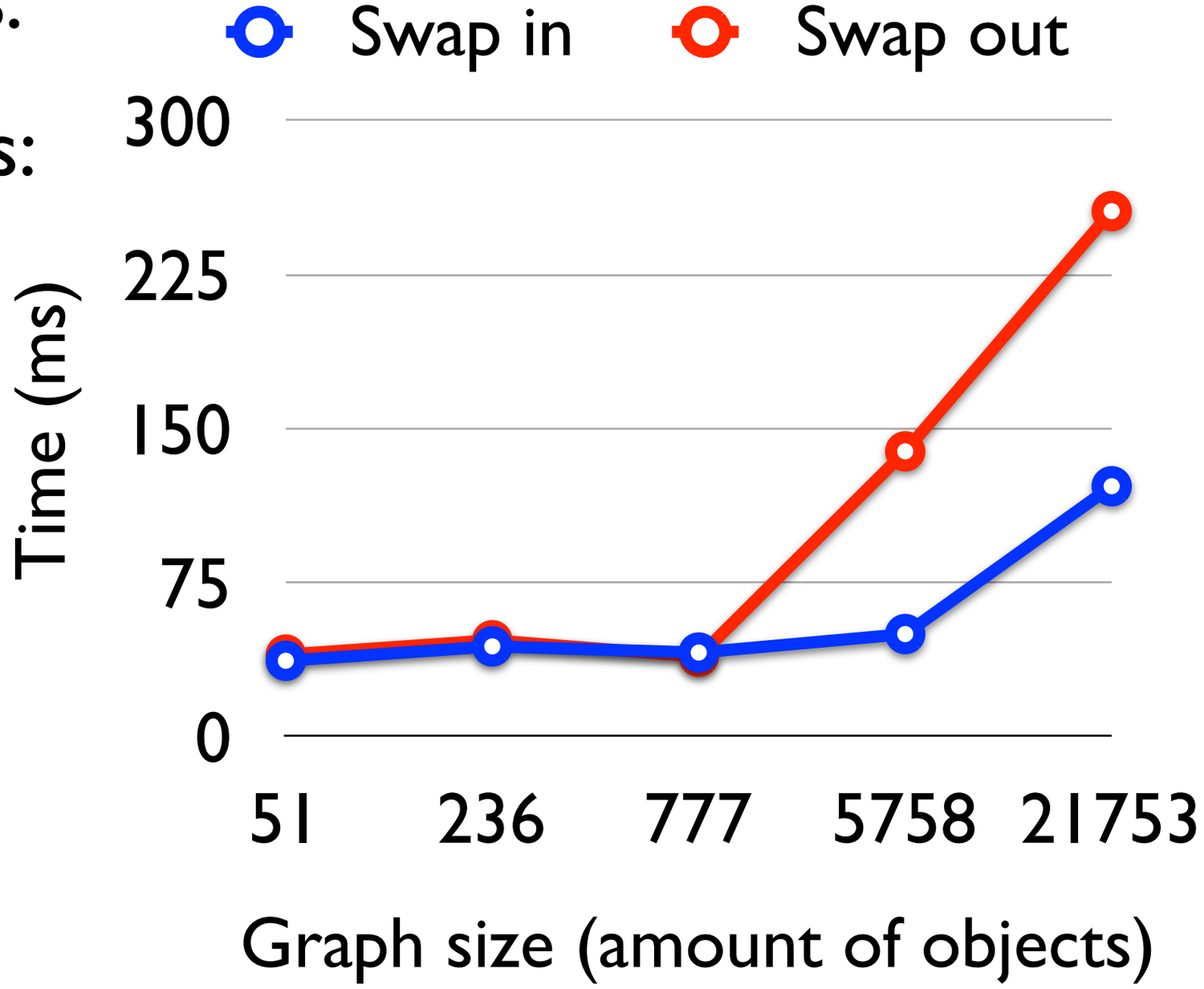
Speed overhead

- For typical uses is 0%.
- For non covered uses:

Speed overhead

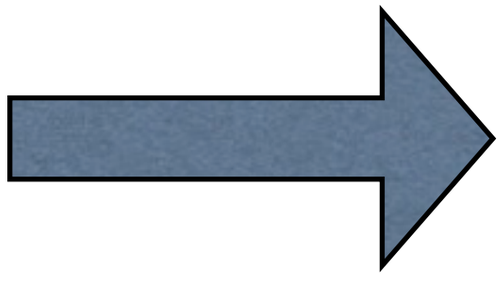
- For typical uses is 0%.
- For non covered uses:

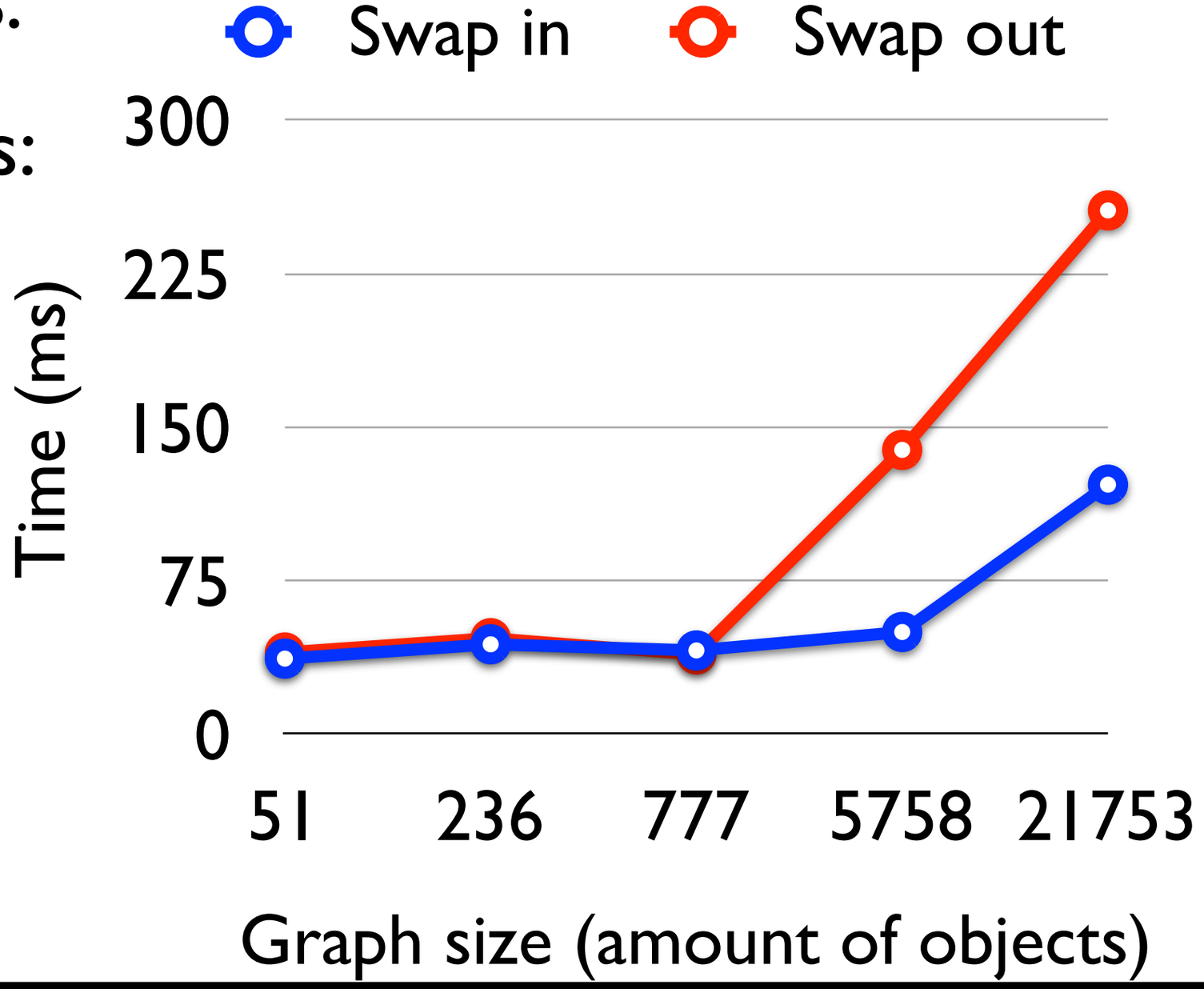




Speed overhead

- For typical uses is 0%.
- For non covered uses:

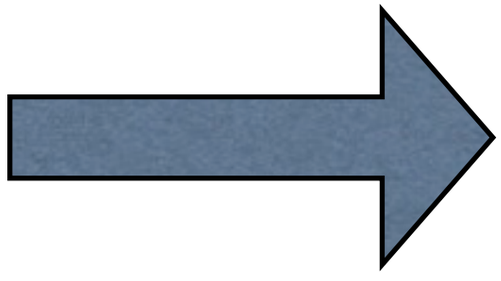


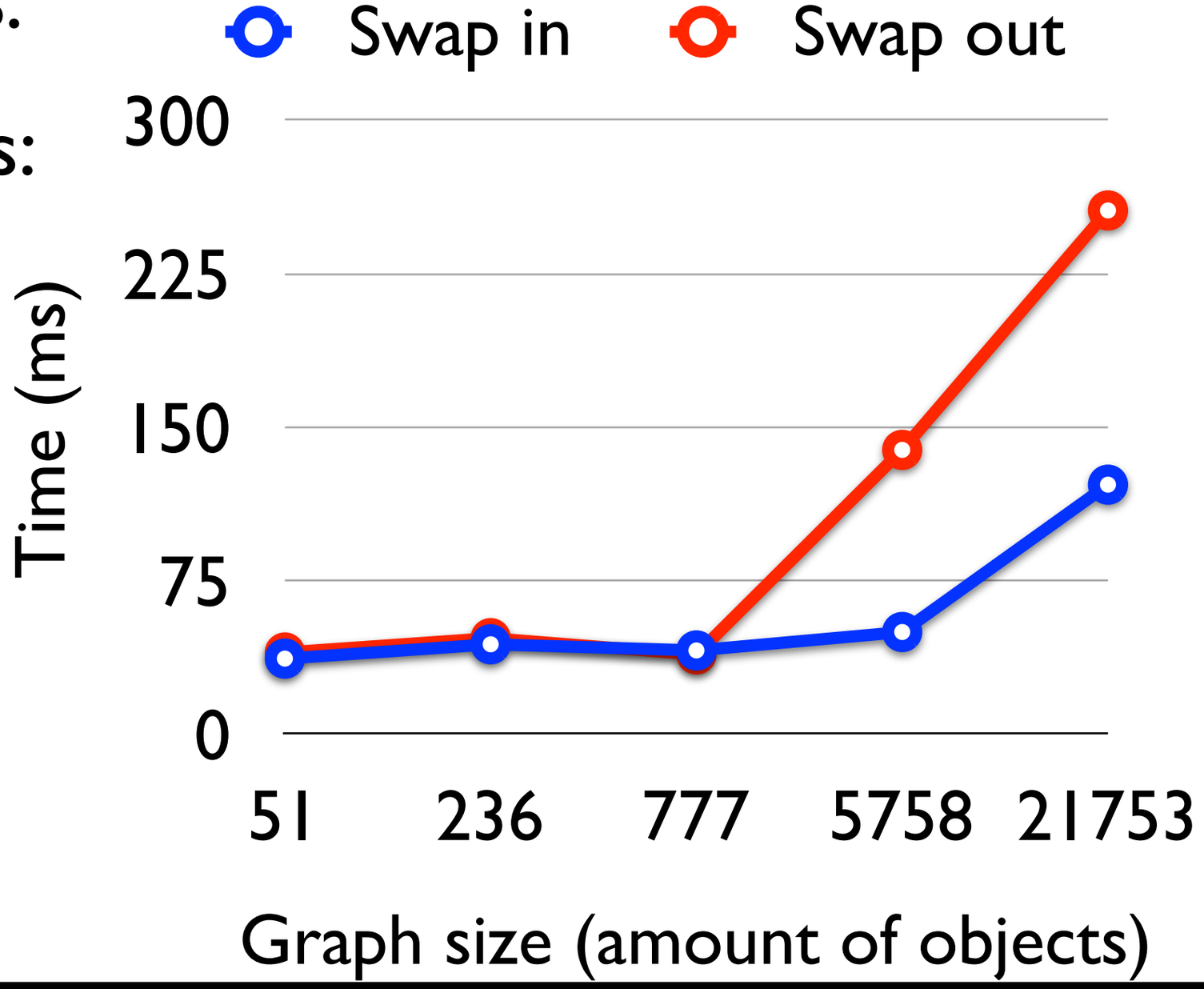


Swapping in is faster than swapping out.

Speed overhead

- For typical uses is 0%.
- For non covered uses:





Swapping in is faster than swapping out.

The graph size has a small impact in small/medium graphs.

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Comparison criteria

- Efficient object-based swapping unit.
- Uniformity.
- Reversibility.
- Automatic swapping in.
- Automatic swapping out.
- Transparency.
- Controllability at the application level.
- Portability.

Language level and libraries

Runtimes

OS related

	OS Virtual Memory	Mach, V++, Apertos	Krueger et al.	Exokernel	Reduced and Specialized Runtimes	Custom and Specific Runtimes	Orthogonal Persistence and ODBMS	LOOM - Large Object-Oriented Memory	Melt - Tolerating Memory Leaks	ImageSegment
Efficient Object-Based Swapping Unit	○	○	○	○	—	—	◐	◐	◐	●
Uniformity	●	●	●	●	○	○	○	◐	◐	○
Reversibility	●	●	●	●	○	○	●	●	●	●
Automatic Swapping In	●	●	●	●	—	—	◐	●	●	●
Automatic Swapping Out	●	●	●	●	—	—	○	●	●	○
Transparency	●	●	●	●	○	○	◐	●	●	◐
Controllability at the Application Level	○	◐	●	●	○	●	●	○	—	●
Portability	○	○	○	○	●	●	●	○	○	○

Language level and libraries

Runtimes

OS related

	OS Virtual Memory	Mach, V++, Apertos	Krueger et al.	Exokernel	Reduced and Specialized Runtimes	Custom and Specific Runtimes	Orthogonal Persistence and ODBMS	LOOM - Large Object-Oriented Memory	Melt - Tolerating Memory Leaks	ImageSegment
Efficient Object-Based Swapping Unit	○	○	○	○	—	—	◐	◐	◐	●
Uniformity	●	●	●	●	○	○	○	◐	◐	○
Reversibility	●	●	●	●	○	○	●	●	●	●
Automatic Swapping In	●	●	●	●	—	—	◐	●	●	●
Automatic Swapping Out	●	●	●	●	—	—	○	●	●	○
Transparency	●	●	●	●	○	○	◐	●	●	◐
Controllability at the Application Level	○	◐	●	●	○	●	●	○	—	●
Portability	○	○	○	○	●	●	●	○	○	○

Language level and libraries

Runtimes

OS related

	OS Virtual Memory	Mach, V++, Apertos	Krueger et al.	Exokernel	Reduced and Specialized Runtimes	Custom and Specific Runtimes	Orthogonal Persistence and ODBMS	LOOM - Large Object-Oriented Memory	Melt - Tolerating Memory Leaks	ImageSegment
Efficient Object-Based Swapping Unit	○	○	○	○	—	—	◐	◐	◐	●
Uniformity	●	●	●	●	○	○	○	◐	◐	○
Reversibility	●	●	●	●	○	○	●	●	●	●
Automatic Swapping In	●	●	●	●	—	—	◐	●	●	●
Automatic Swapping Out	●	●	●	●	—	—	○	●	●	○
Transparency	●	●	●	●	○	○	◐	●	●	◐
Controllability at the Application Level	○	◐	●	●	○	●	●	○	—	●
Portability	○	○	○	○	●	●	●	○	○	○

Language level and libraries

Runtimes

OS related

	OS Virtual Memory	Mach, V++, Apertos	Krueger et al.	Exokernel	Reduced and Specialized Runtimes	Custom and Specific Runtimes	Orthogonal Persistence and ODBMS	LOOM - Large Object-Oriented Memory	Melt - Tolerating Memory Leaks	ImageSegment
Efficient Object-Based Swapping Unit	○	○	○	○	—	—	◐	◐	◐	●
Uniformity	●	●	●	●	○	○	○	◐	◐	○
Reversibility	●	●	●	●	○	○	●	●	●	●
Automatic Swapping In	●	●	●	●	—	—	◐	●	●	●
Automatic Swapping Out	●	●	●	●	—	—	○	●	●	○
Transparency	●	●	●	●	○	○	◐	●	●	◐
Controllability at the Application Level	○	◐	●	●	○	●	●	○	—	●
Portability	○	○	○	○	●	●	●	○	○	○

Language level and libraries

Runtimes

OS related

	OS Virtual Memory	Mach, V++, Apertos	Krueger et al.	Exokernel	Reduced and Specialized Runtimes	Custom and Specific Runtimes	Orthogonal Persistence and ODBMS	LOOM - Large Object-Oriented Memory	Melt - Tolerating Memory Leaks	ImageSegment	Marea
Efficient Object-Based Swapping Unit	○	○	○	○	—	—	◐	◐	◐	●	●
Uniformity	●	●	●	●	○	○	○	◐	◐	○	●
Reversibility	●	●	●	●	○	○	●	●	●	●	●
Automatic Swapping In	●	●	●	●	—	—	◐	●	●	●	●
Automatic Swapping Out	●	●	●	●	—	—	○	●	●	○	◌
Transparency	●	●	●	●	○	○	◐	●	●	◐	●
Controllability at the Application Level	○	◐	●	●	○	●	●	○	—	●	●
Portability	○	○	○	○	●	●	●	○	○	○	●

Language level and libraries

Runtimes

OS related

	OS Virtual Memory	Mach, V++, Apertos	Krueger et al.	Exokernel	Reduced and Specialized Runtimes	Custom and Specific Runtimes	Orthogonal Persistence and ODBMS	LOOM - Large Object-Oriented Memory	Melt - Tolerating Memory Leaks	ImageSegment	Marea
Efficient Object-Based Swapping Unit	○	○	○	○	—	—	◐	◐	◐	●	●
Uniformity	●	●	●	●	○	○	○	◐	◐	○	●
Reversibility	●	●	●	●	○	○	●	●	●	●	●
Automatic Swapping In	●	●	●	●	—	—	◐	●	●	●	●
Automatic Swapping Out	●	●	●	●	—					○	●
Transparency	●	●	●	●	○					◐	●
Controllability at the Application Level	○	◐	●	●	○	●	●	○	—	●	●
Portability	○	○	○	○	●	●	●	○	○	○	●

In progress

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

in

Object Graph Swapper	Proxy Toolbox 
Object Graph Serializer 	Object Graph Storage  

- An extensible, uniform and fast serializer.
- General-purpose.
- Used in research and industry.
- Integrated in Pharo 2.0 and ported to Squeak, VisualWorks and Newspeak.
- ESUG Innovation Technology Awards 2011.

- An extensible, uniform and fast serializer.
- General-purpose.
- Used in research and industry.

- Integrated in Pharo 2.0 and ported to Squeak, VisualWorks and Newspeak.
- ESUG Innovation Technology Awards 2011.

Martin Dias, **Mariano Martinez Peck**, Stéphane Ducasse and Gabriela Arévalo. ***Fuel: A Fast General Purpose Object Graph Serializer***. Software: Practice and Experience, 2012.

- Proxify all kind of objects.
- Low memory footprint.
- Intercepting all messages.
- Clear division between proxies and handlers.
- General-purpose.

- Proxify all kind of objects.
- Low memory footprint.
- Intercepting all messages.
- Clear division between proxies and handlers.
- General-purpose.

[Martinez Peck 2012a] **Mariano Martinez Peck**, Noury Bouraqadi, Stéphane Ducasse, Luc Fabresse and Marcus Denker. ***Ghost: A Uniform and Lightweight Proxy Implementation***. Science of Computer Programming, 2012 **(submitted + passed first review round).**

Object replacement

- Provided by regular VM (#become:)
  - Specific to Smalltalk.
- Different implementations
  - Full memory scan in Pharo VM.
  - Object header swapping in VisualWorks VM.
  - Object tables' entries swapping in Gemstone VM.

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

Outline

1. Context, problem and introduction.
2. My proposal.
3. Validation.
4. Related work.
5. Implementation.
6. Conclusion and future work

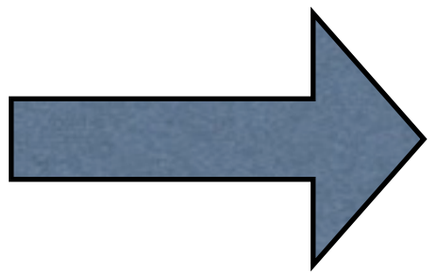
Conclusion

For an improved automatic memory management we need:

GC

+

An application-level virtual memory



We propose Marea

Results

☑ Correct

- Facade objects.
- Graph intersection.
- Serializer and proxies for all type of objects.
- Proxies intercept all messages.
- Support proxies in primitives.

Results

☑ Maximizes memory released

- Small proxies.
- Object graphs as swapping unit.
- Proxies only for facade objects.

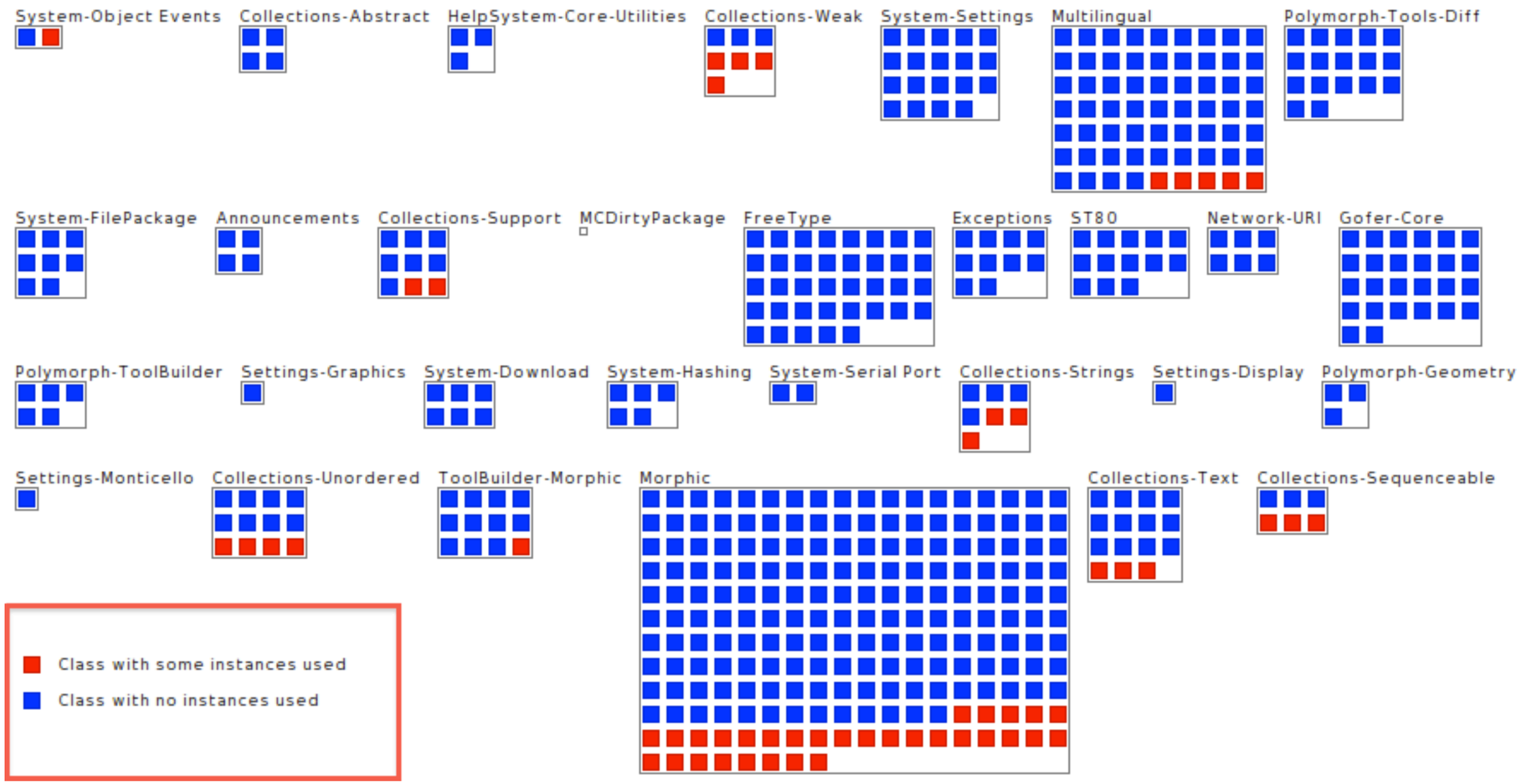
☑ Minimizes speed overhead

- Fast serialization.
- Fast detection of facade objects.
- Avoid unnecessary swap in.

Future Work

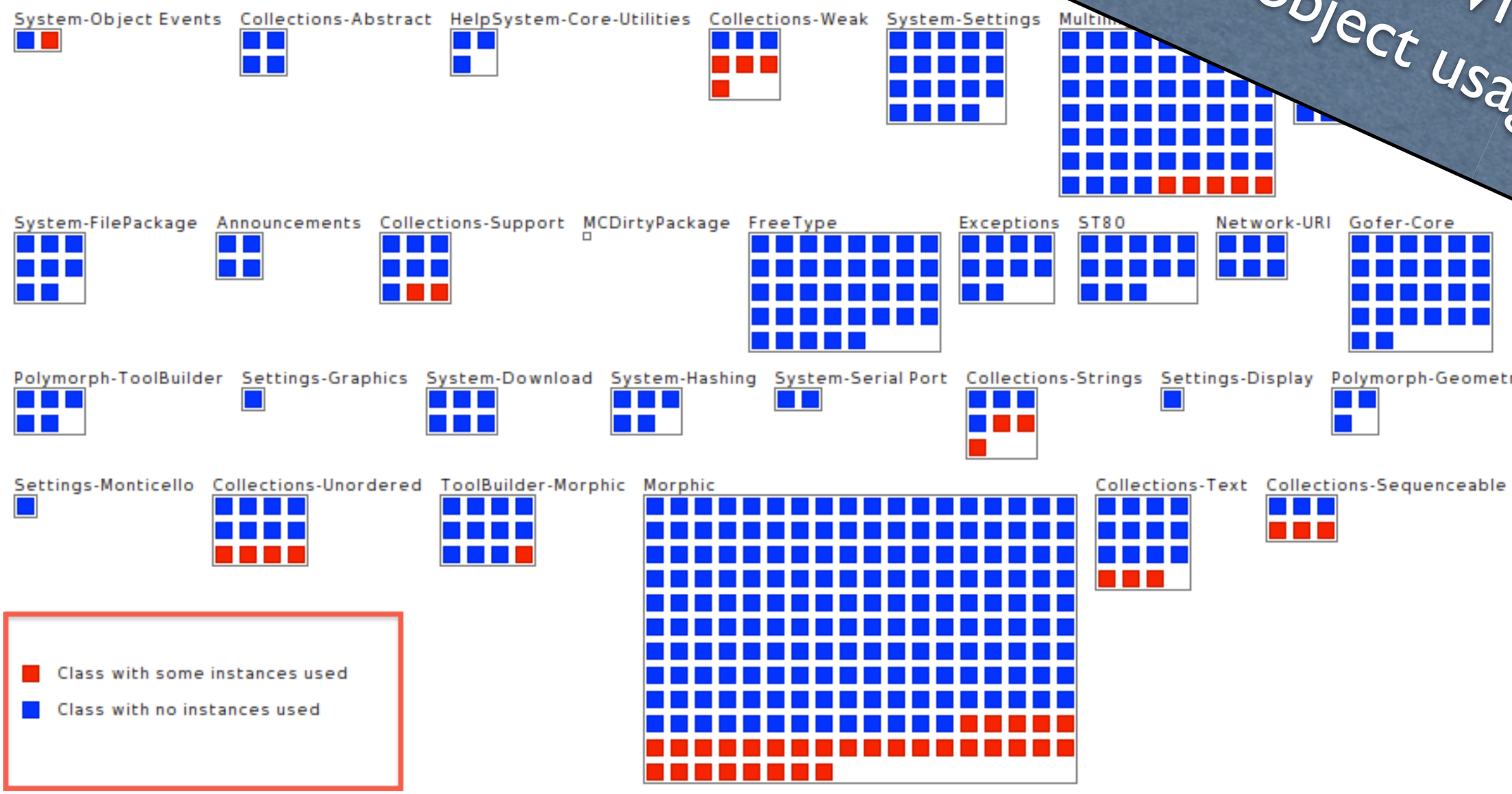
- Automatic graphs detection.

Visualizing Objects Usage



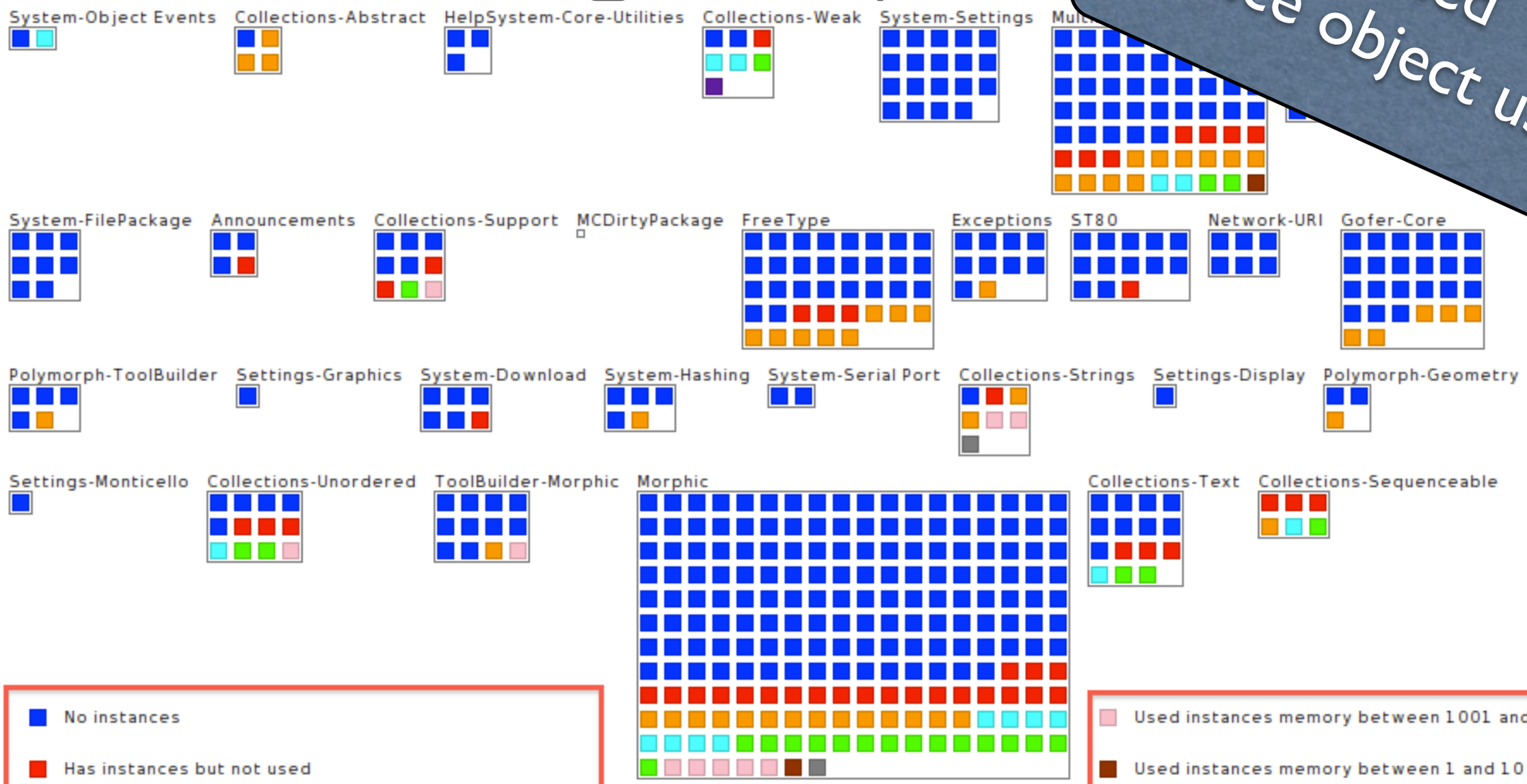
Visualizing Objects

Customized VM to trace object usage



Visualizing Objects

Customized VM to trace object usage



Future Work

- Automatic graphs detection.
- Study possible integration with GC.
- Partial swap in.

List of publications

Conferences

[Dias 2011] Martin Dias, **Mariano Martinez Peck**, Stéphane Ducasse and Gabriela Arévalo. ***Clustered Serialization with Fuel***. In Proceedings of ESUG International Workshop on Smalltalk Technologies (IWST 2011), Edinburgh, Scotland, 2011.

[Martinez Peck 2010a] **Mariano Martinez Peck**, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse and Luc Fabresse. ***Experiments with a Fast Object Swapper***. In Smalltalks 2010, Concepción del Uruguay, Argentina, 2010.

[Martinez Peck 2010b] **Mariano Martinez Peck**, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse and Luc Fabresse. ***Visualizing Objects and Memory Usage***. In Smalltalks 2010, Concepción del Uruguay, Argentina, 2010.

[Martinez Peck 2011a] **Mariano Martinez Peck**, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse and Luc Fabresse. ***Efficient Proxies in Smalltalk***. In Proceedings of ESUG International Workshop on Smalltalk Technologies (IWST 2011), Edinburgh, Scotland, 2011.

[Martinez Peck 2011b] **Mariano Martinez Peck**, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse and Luc Fabresse. ***Problems and Challenges when Building a Manager for Unused Objects***. In Proceedings of Smalltalks 2011 International Workshop, Bernal, Buenos Aires, Argentina, 2011.

List of publications

Journals

[Martinez Peck 2011c] **Mariano Martinez Peck**, Noury Bouraqadi, Stéphane Ducasse and Luc Fabresse. ***Object Swapping Challenges: an Evaluation of ImageSegment***. Journal of Computer Languages, Systems and Structures, vol. 38, no. 1, pages 1–15, nov 2011.

[Dias 2012] Martin Dias, **Mariano Martinez Peck**, Stéphane Ducasse and Gabriela Arévalo. ***Fuel: A Fast General Purpose Object Graph Serializer***. Software: Practice and Experience, 2012.

[Martinez Peck 2012a] **Mariano Martinez Peck**, Noury Bouraqadi, Stéphane Ducasse, Luc Fabresse and Marcus Denker. ***Ghost: A Uniform and Lightweight Proxy Implementation***. Science of Computer Programming, 2012 **(submitted + passed first review round).**

[Martinez Peck 2012b] **Mariano Martinez Peck**, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse and Luc Fabresse. ***Object-Based Virtual Memory Brought To The Application Level***. Journal of Object Technology, 2012 **(submitted).**

In summary...

- We implemented a novel and efficient application-level virtual memory for object-oriented systems.
- Almost no need from the VM and fully implemented in the language side.
- Users can decide and influence what and when to swap.

Thanks!

Mariano Martinez Peck

Université Lille Nord de France
Pôle de Recherche et d'Enseignement Supérieur

Mines Douai
LILLE EURORÉGION

Inria

RMod

Université Lille1
Sciences et Technologies