View Source: Design Patterns in the Wild

Caching Strategies

Ashwin J Mathew
School of Information, UC Berkeley

Spring 2008

Today

- Memory Caching
- Distributed Caching
- Layered Cache Architectures
- Abstract Factory Pattern

Why Cache?

- Store data closest to where it's needed
- Depending on where data is stored, trade-off between:
  - CPU cycles
  - Memory consumption
  - Disk access
  - Network utilization

Cache Storage

- Increase locality of data
- Balance resource usage for data retrieval and storage
- Remember – it's a temporary store

Memory

Disk

Network

Caching Strategies

- Cost-based
  - Fixed size
  - Responsive to item state
- Memory sensitive
  - Variable size
  - Responsive to system state
- Time sensitive
  - Often combined with other strategies

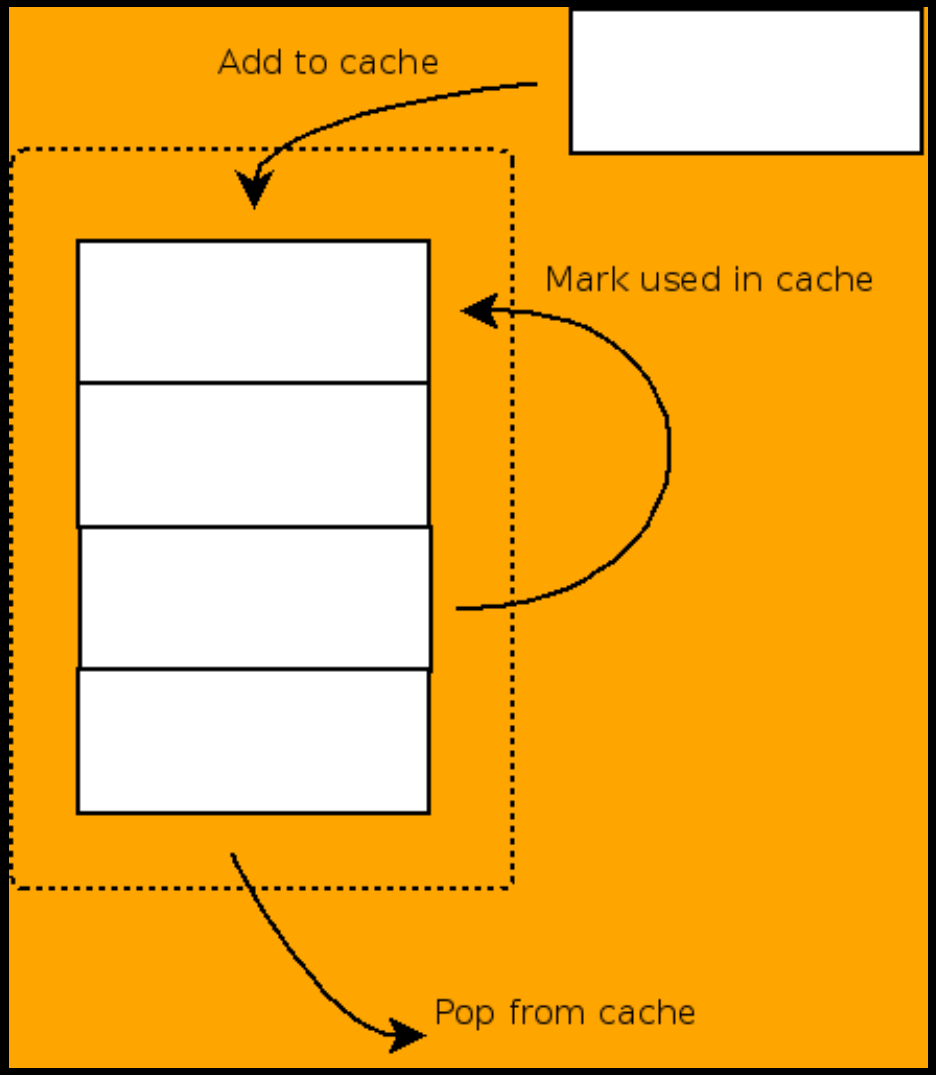
Hit Ratio

$$= \frac{\text{Hits}}{\text{Total Requests}}$$

Hits - Requests which find items in cache

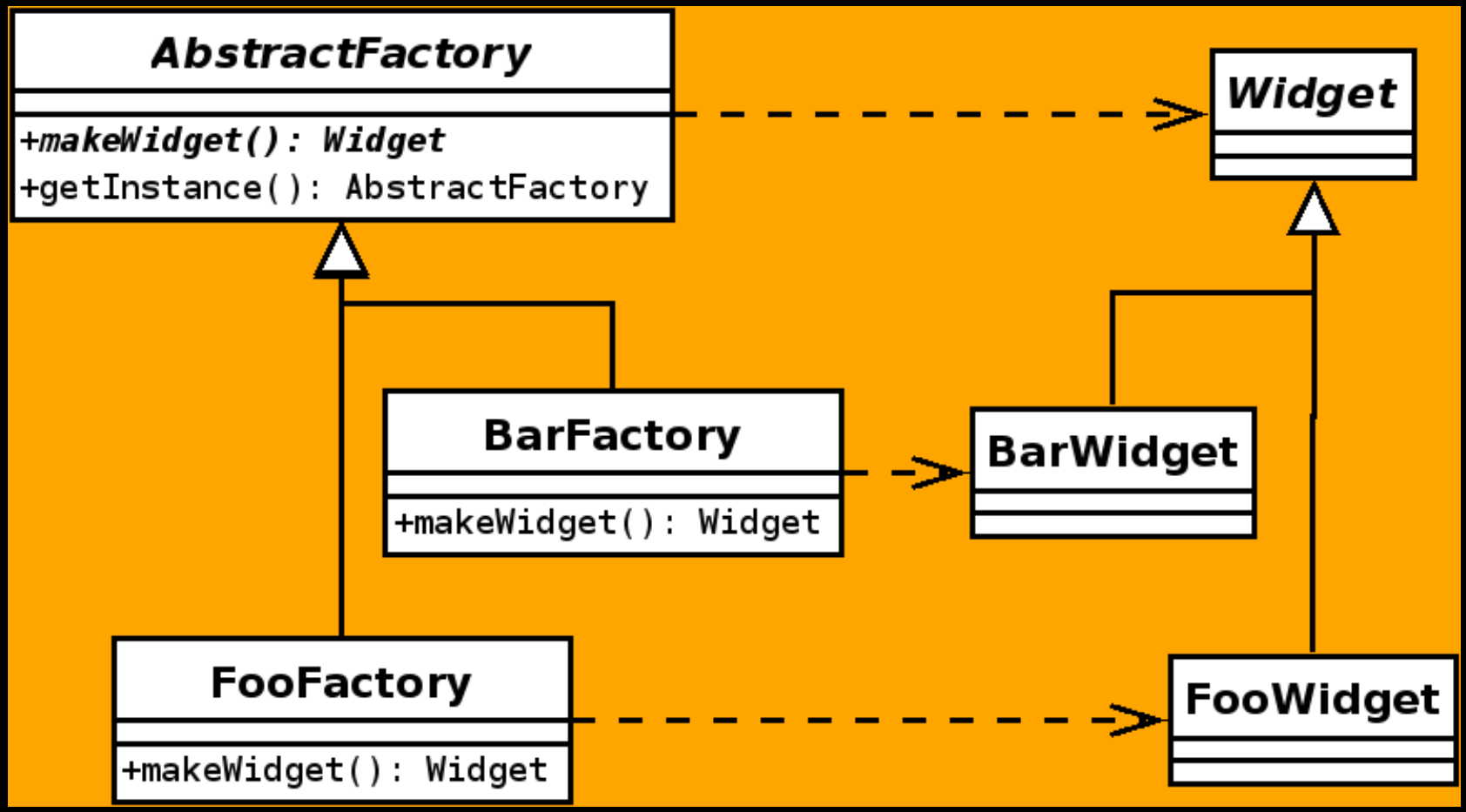
Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

Cost-based Caching



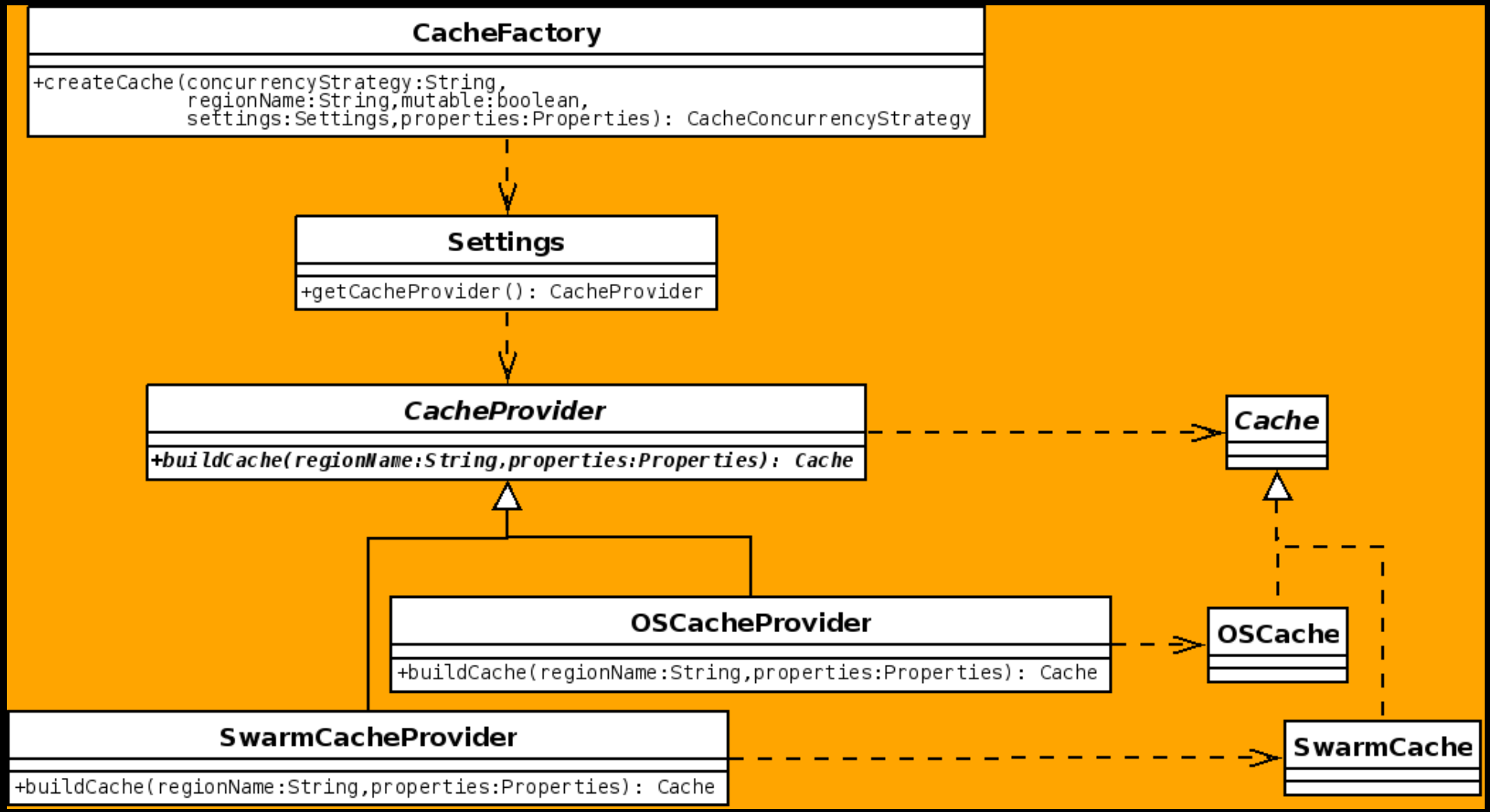
- Fixed cache size
- Cost-based algorithm to maintain size
  - LRU, LFU, etc.
- Cache size critical to performance

Abstract Factory



Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

Abstract Factory in Hibernate

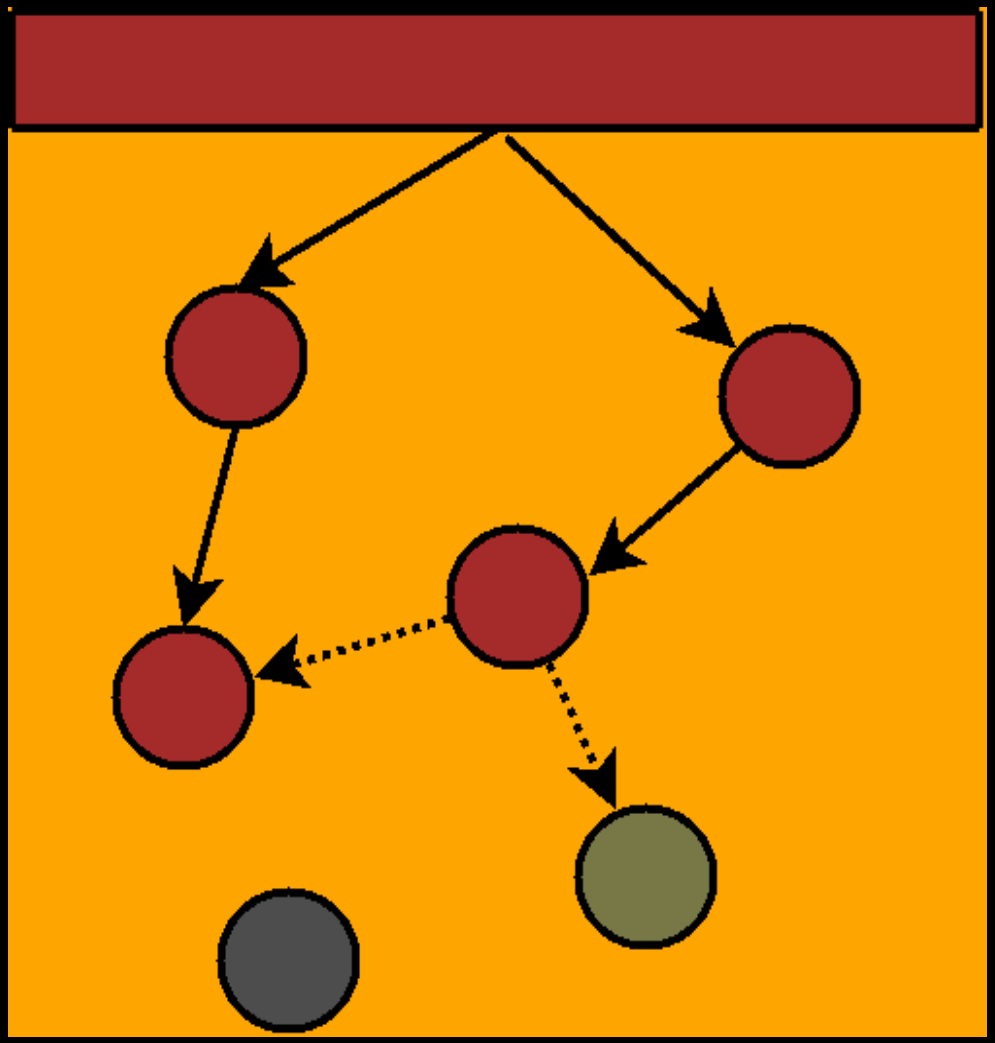


Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

Using Abstract Factory

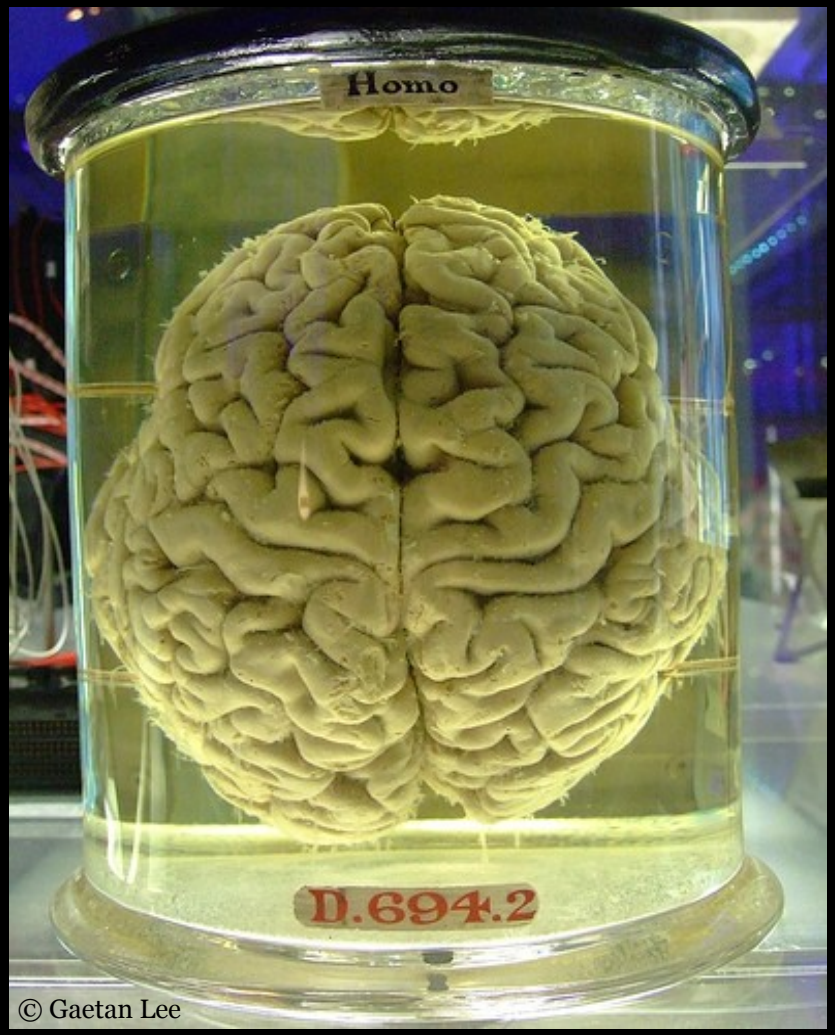
- Many products, many factories
- Delegate construction to different factory method implementations
- Flexibility to configure in new products and associated factory implementations

Soft References



- Java's Garbage Collector frees up memory by removing unreferenced objects
- Soft References are maintained as long as the GC is not seeking memory

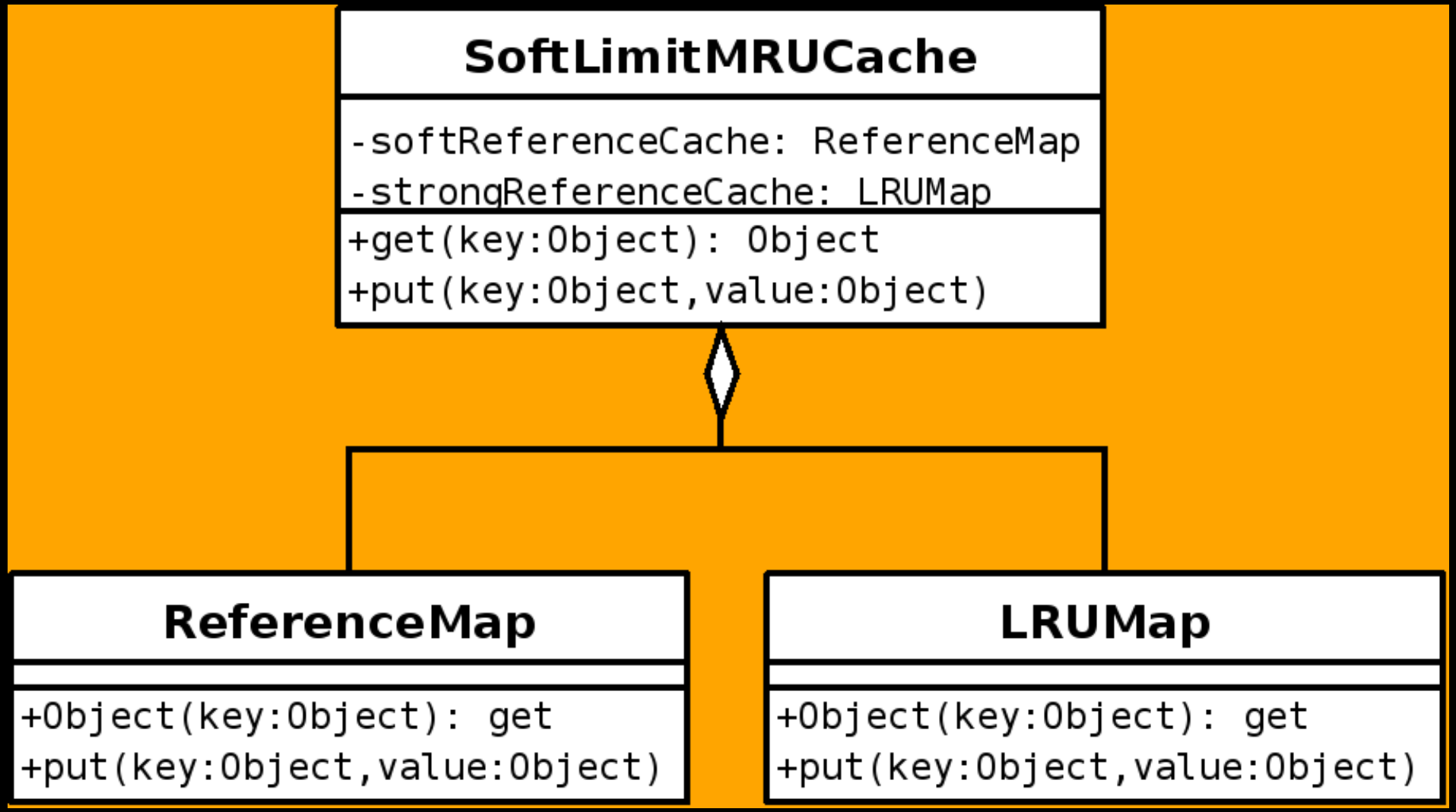
Memory Sensitive Caching



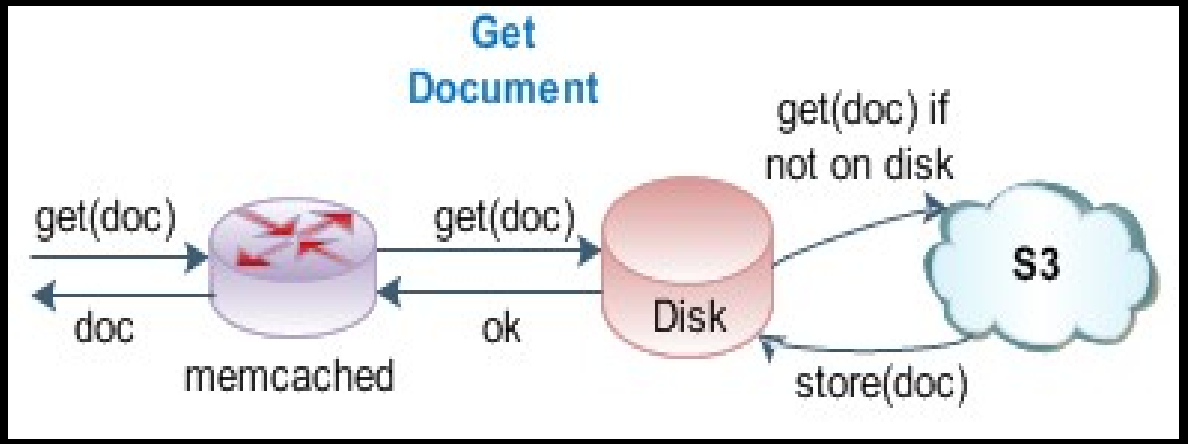
- Variable size
- Responsive to system constraints
- Potential object churn under load

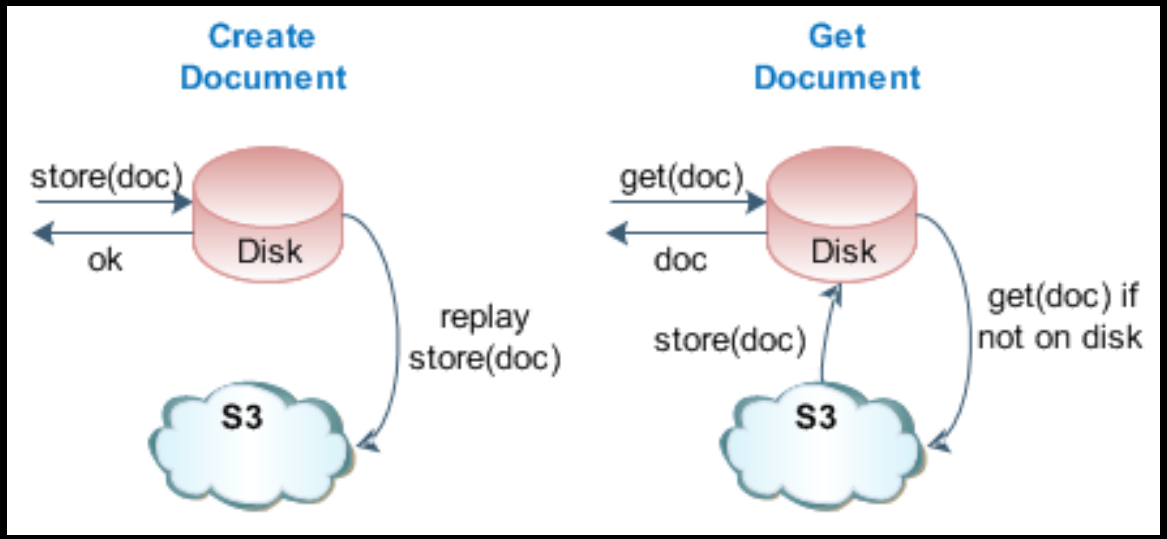
Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

Stacking Strategies



Stacking with Thrudb





Images from http://www.igvita.com/2007/12/28/thrudb-faster-and-cheaper-than-simpledb/

Using Caching

- Improve performance
  - Increase locality of data
  - Pre-process data
- Tuning is critical
  - Hit Ratio
  - Overall system performance
- Warning: Caches may sync amongst themselves – but yet be out of sync with underlying data stores