Controlling Items in the Solution Cache

Elton Stoneman
elton@sixeyed.com

pluralsight
hardcore developer training

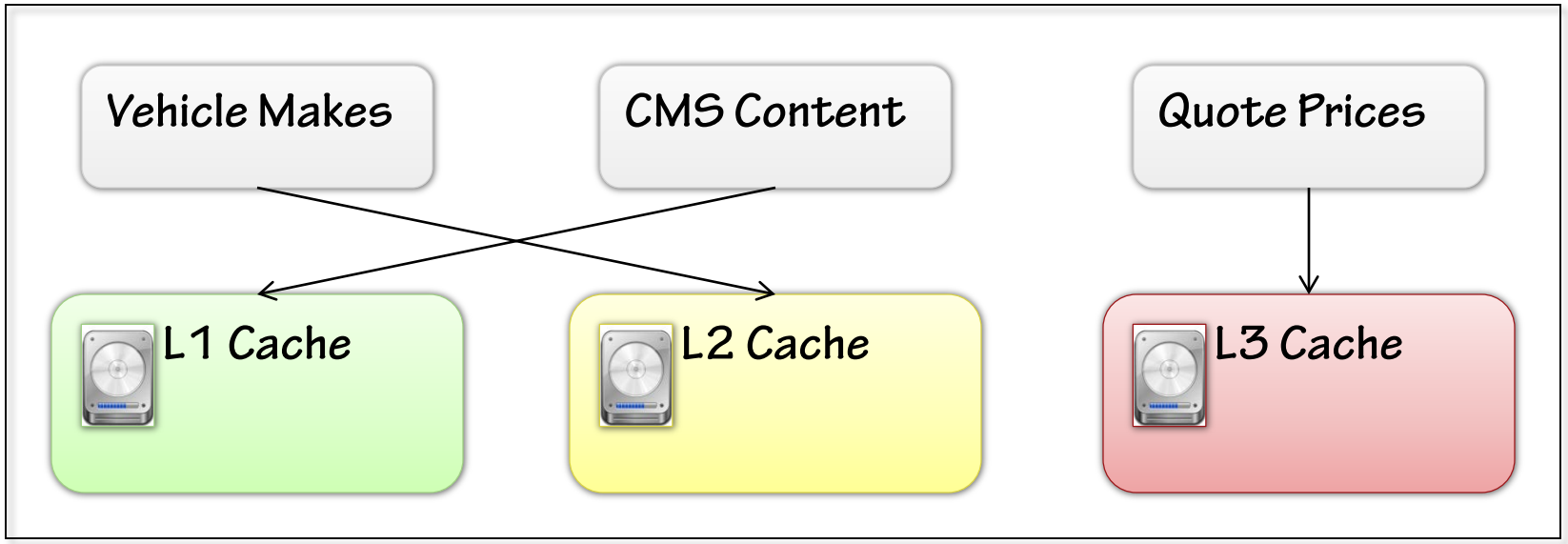
Outline

- **Adding items to the cache**
  - Specifying and changing the cache store
- **Removing items from the cache**
  - Manually
    - Cache reset
    - Invalidation
  - Automatically
    - Expiration
    - Eviction
- **Disabling the cache**
  - Per-item
  - Globally
- **Preloading the cache**

Adding items to the cache

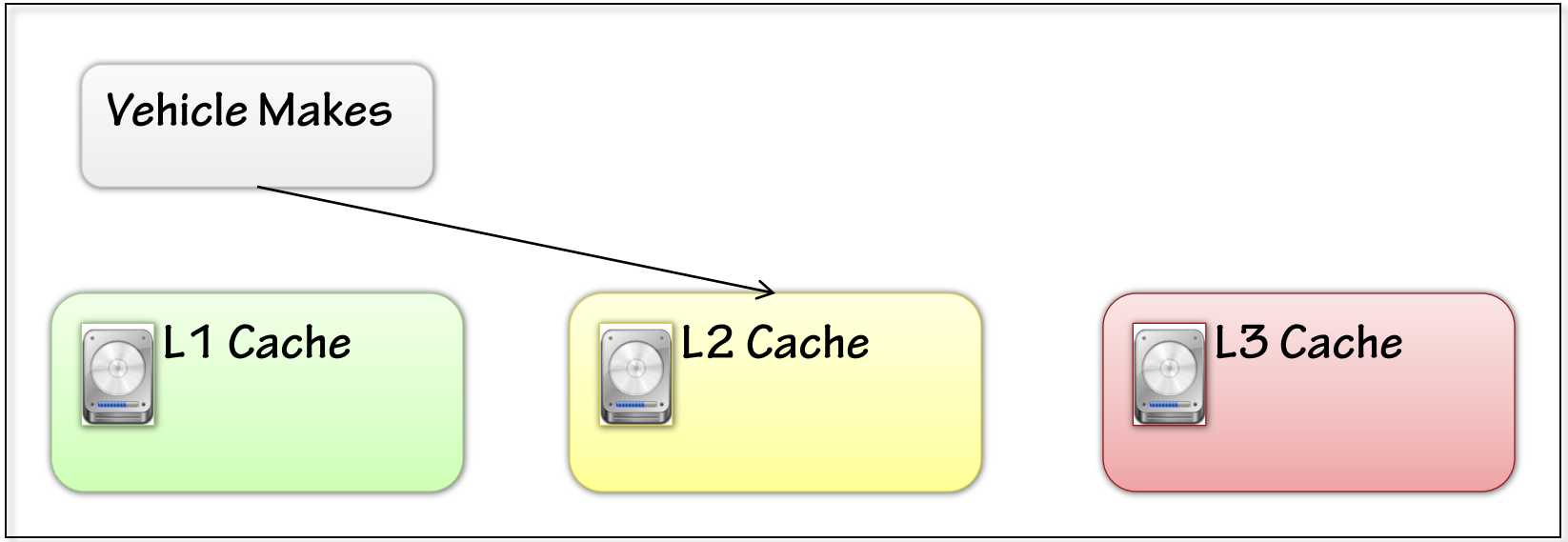
- Specify the cache store at design time

```
[Cache(CacheType=CacheType.Disk)]
public virtual IEnumerable<Make> GetMakes()
```



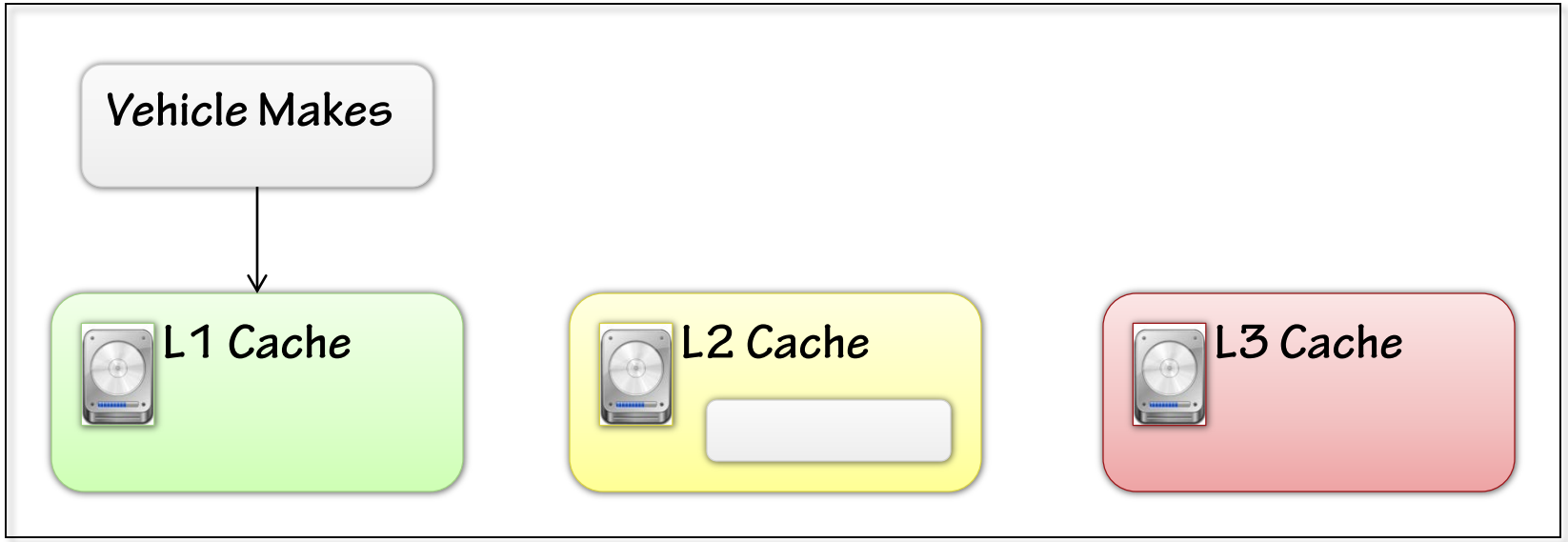
Adding items to the cache

- At run-time, in app config



Adding items to the cache

- At run-time, in app config



Adding items to the cache

- Demo

Adding items to the cache

- At run-time, in app config
- Keyed by cache key prefix
  - ContentClient.GetItemsInternal_c8d31f05-1c00-4284-374f-c3422e1f27aa
  - ContentClient.GetItemsInternal
- Cache target element

```
<sixeyed.core.caching>
  <targets>
    <target keyPrefix="ContentClient.GetItemsInternal"
            cacheType="AppFabric"/>
```

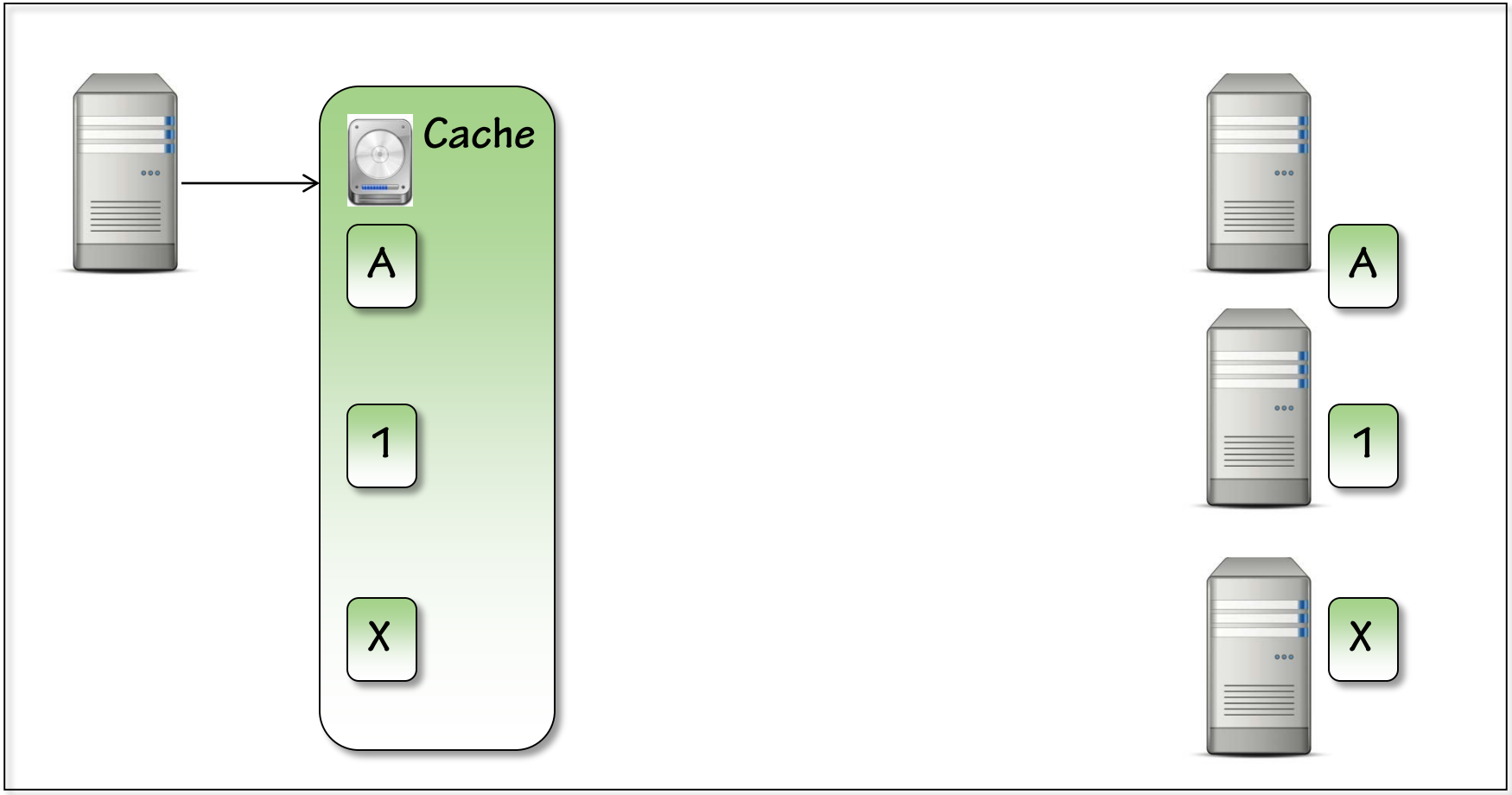
- Process recycle on config change

Removing items from the cache

- **Manually – cache reset**
  - Clear all items
- **In-memory**
  - Recycle process(es)
- **Persistent**
  - Delete items
- **Blunt approach**
  - But useful
  - Documented process, e.g.
    - Recycle app pools (IIS)
    - Restart cache cluster (PowerShell)
    - Delete the files in the cache library (Explorer)

Removing items from the cache

- Manually – cache reset

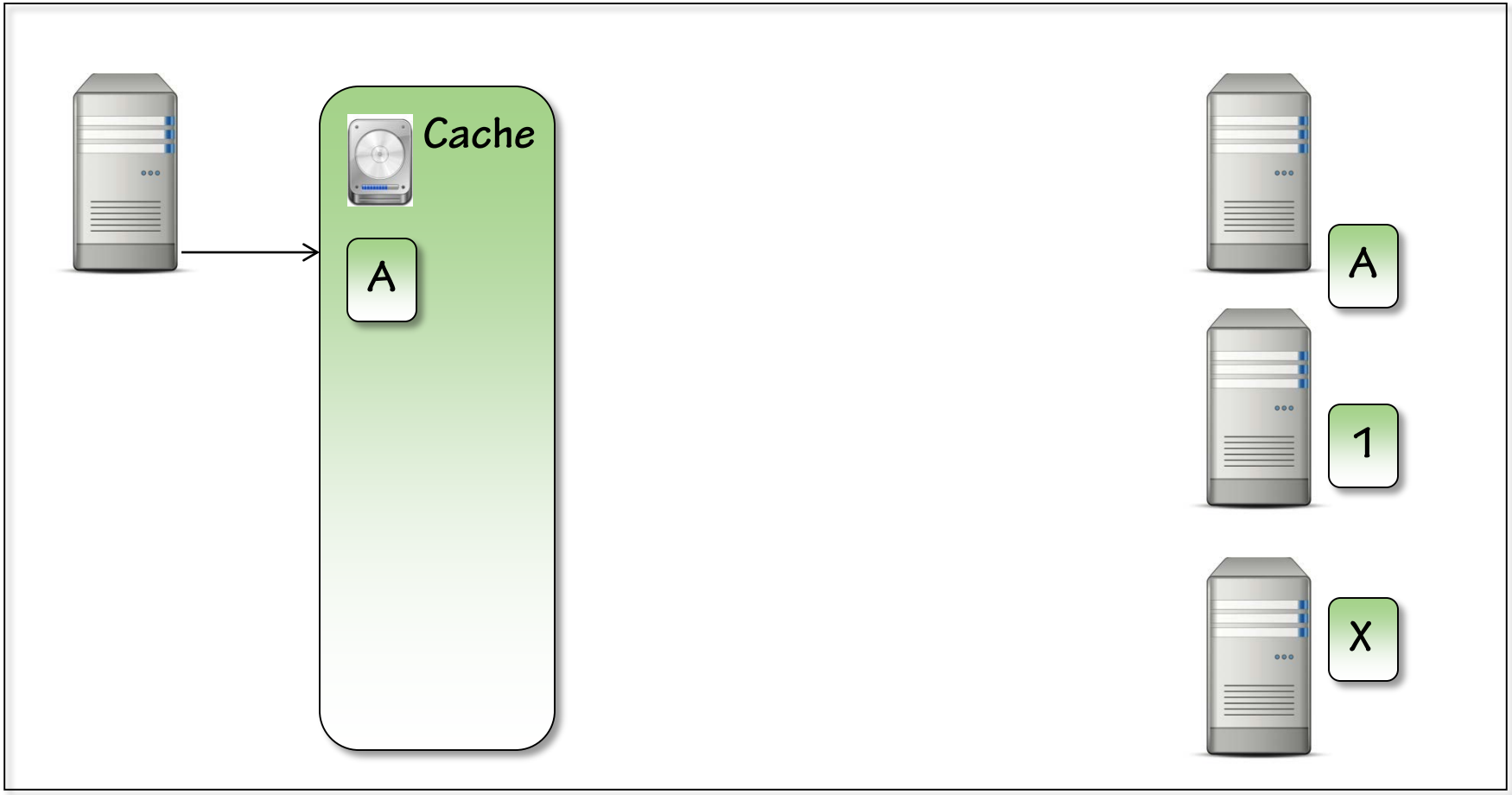


Removing items from the cache

- Manually – cache reset

Removing items from the cache

- Manually – cache reset

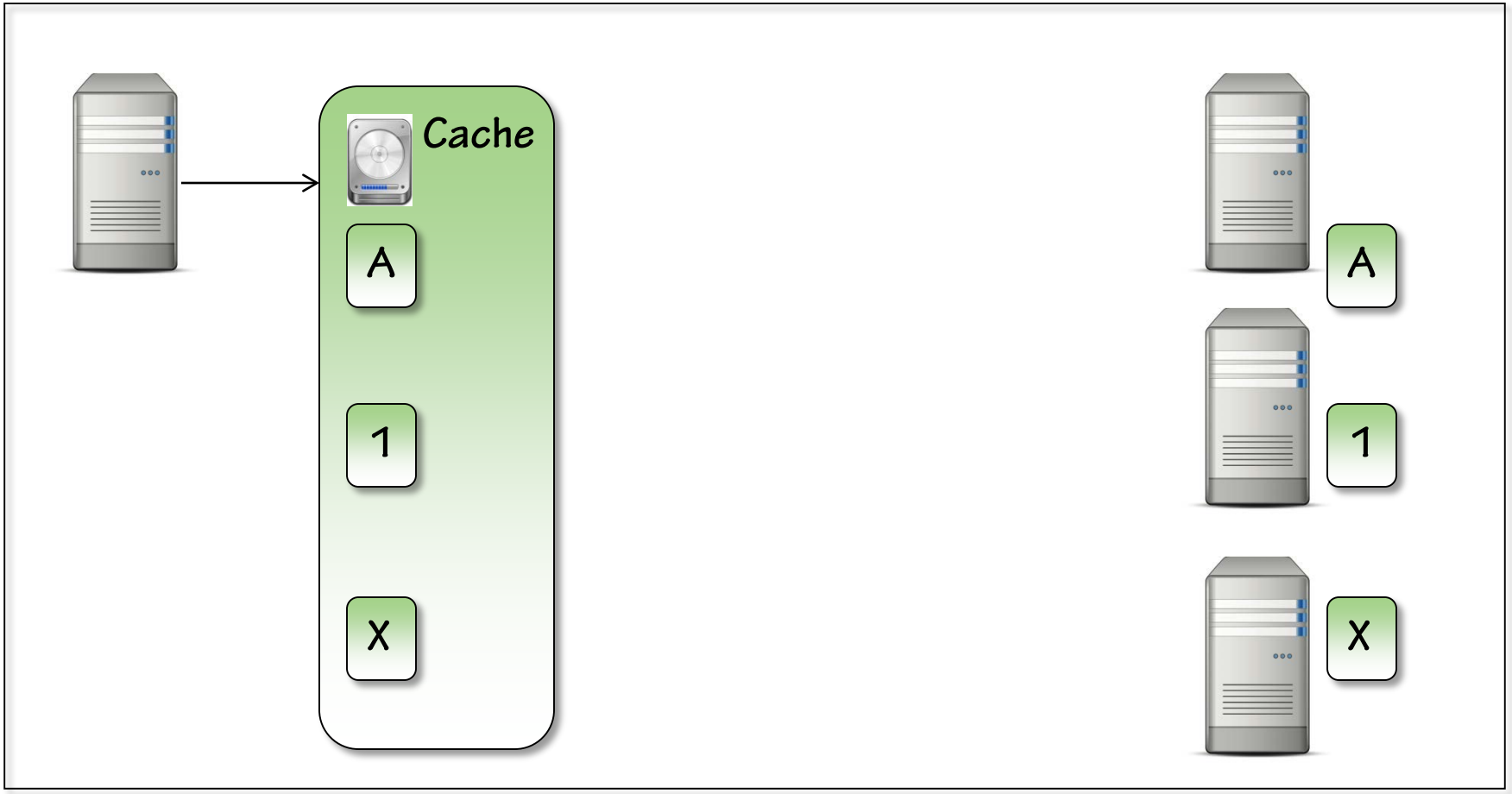


Removing items from the cache

- **Manually – item invalidation**
  - Remove individual items
  - Control object updates
- **ICache.Remove**
  - Cache key

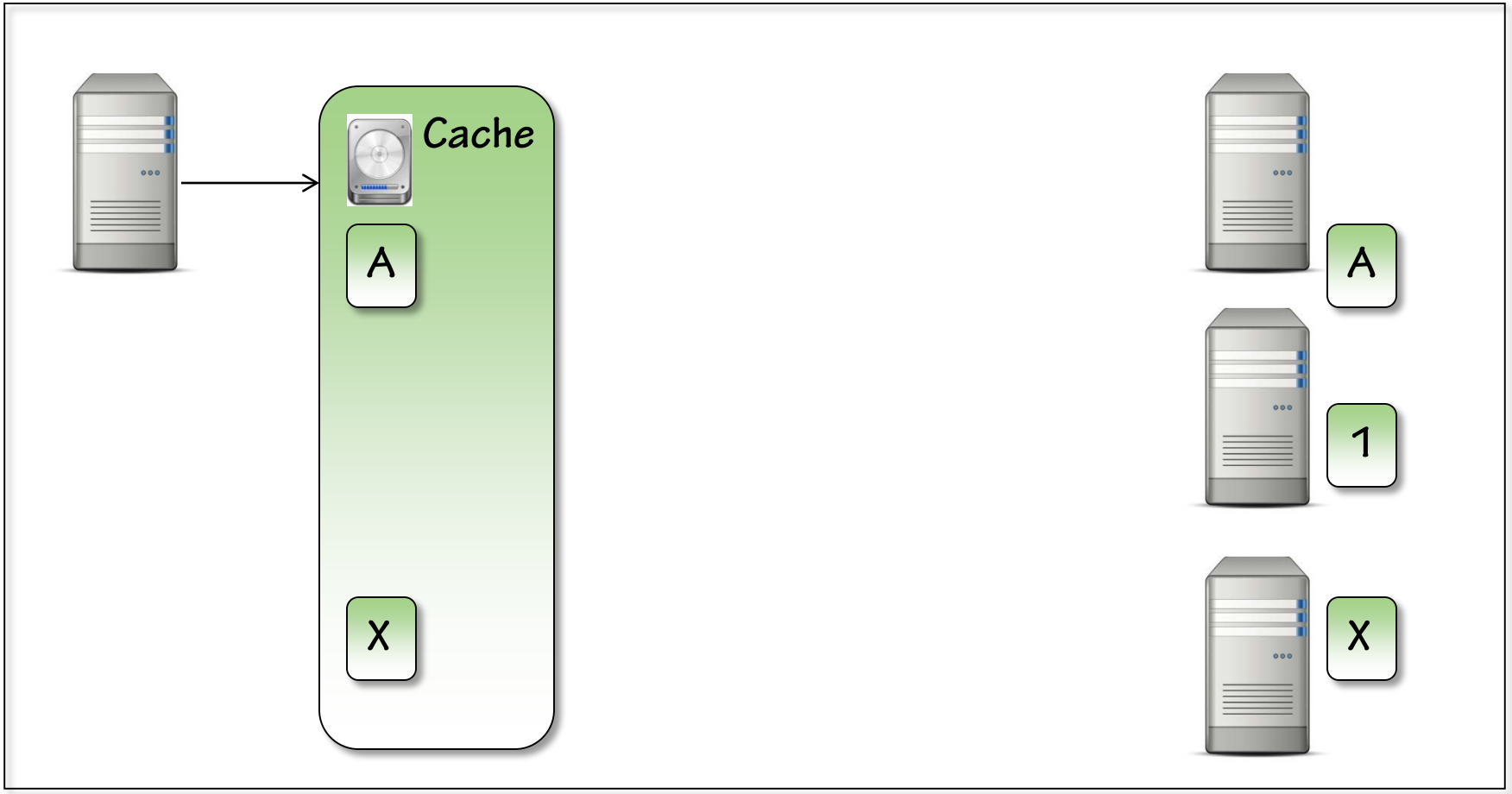
Removing items from the cache

- Manually – item invalidation



Removing items from the cache

- Manually – item invalidation



Removing items from the cache

- **Not generic**
  - May not control object updates
  - Different triggers and trigger types
    - Limited support in cache stores
    - Overhead
  - Multiple items cached per-method
- **By exception**
  - Manual implementation
  - Cost to build

Removing items from the cache

- **Automatically – expiration**
  - Sliding and absolute
- **Sliding**
  - Set with timespan
  - Item removed if not accessed
  - Useful for keeping cache fresh
  - Not universally supported
- **Absolute**
  - Set with expiry time
  - Item removed at expiry, accessed or not
  - Useful for keeping data fresh
  - Well supported
- **Expiry runs at cleanup interval**

Removing items from the cache

- **Automatically – eviction**
  - Some caches provide
- **Keeping the cache fresh**
  - Within configured size
  - Remove items by policy
    - Least Recently Used
    - FIFO
- **Eviction runs at cleanup interval**
- **Item removal transparent to consumers**
  - App follows cache miss path

Removing items from the cache

- Demo

Removing items from the cache

- By absolute expiration
- Cache target element

```
<sixeyed.core.caching>
  <targets>
    <target keyPrefix="ContentClient.GetItemsInternal"
            cacheType="NCacheExpress"
            seconds="20"/>
```

- Days, hours, minutes, seconds
  - Cumulative
  - TimeSpan
- Enforced expiry
  - Metadata for persistent store
  - Cleanup routine for in-memory

Disabling the cache

- **Cache as a non-functional component**
  - Improve performance and scalability
  - Risk serving stale or invalid data
  - Business requirements change
- **Disable the cache**
  - Selectively or globally
  - Without changing the application
- **Application configuration**

Disabling the cache

- Demo

Disabling the cache

- Globally

```
<sixeyed.core.caching enabled="false" />
```

- Per-item

```
<sixeyed.core.caching>
  <targets>
    <target keyPrefix="ContentClient.GetItemsInternal"
            cacheType="NCacheExpress"
            seconds="20"
            enabled="false"/>
```

- Enabled overrides lifespan and cache type
- Global overrides target

Preloading the cache

- **Proactively populate the cache**
  - Bulk load rather than incremental
  - Keep cache items available and fresh
- **Selective appeal**
  - Entry point to load the cache
  - Access to source variable collection
  - Defined trigger
- **Quote Prices**
  - GetQuotes WebService method
  - Database stores vehicle ID and postal code
  - IIS – global.asax on startup; or external tool

Preloading the cache

- Demo

Preloading the cache

- **Made GetQuotes cacheable**
- **Dedicated preloader**
  - Iterates through vehicle IDs and postal codes
  - Spawns a new task for each
  - Calls into GetQuotes
- **Pre-loads quote prices into DiskCache**
  - 1.5M items, 15,000 items per hour
  - Four days to load the full set
  - Targeted subset / distributed process
- **Trigger**
  - Separate process, manual trigger
  - Global application start, app pool scheduled recycle

Summary

- **Adding items - choosing the cache store**
- **Removing items from the cache**
  - Manually
  - Automatically
  - Absolute expiration
- **Disabling the cache**
  - Graceful degradation
- **Preloading the cache**
  - Proactive performance boost