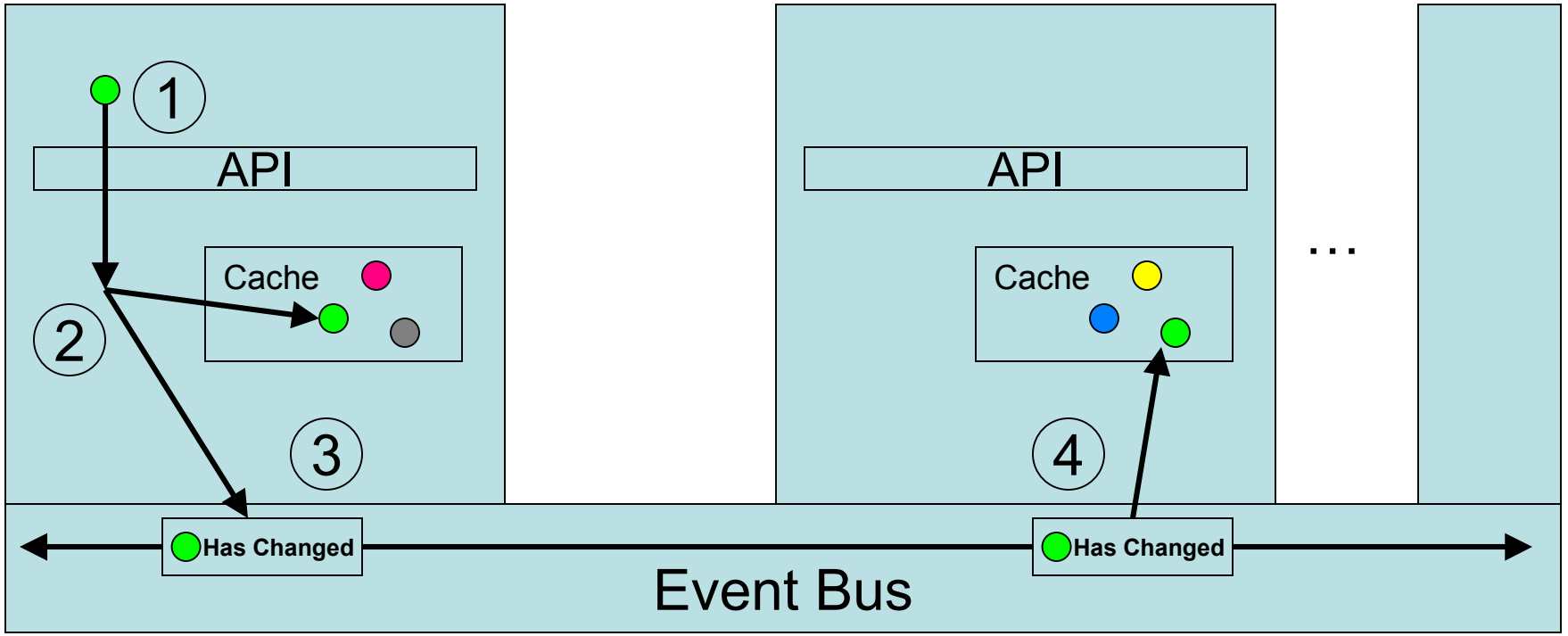
Caching in Sakai

Charles Severance
Notes - 5/20/2006
csev@umich.edu

Clustered Caching Issues

- Problem is that bad caching in clustered environments = bug reports
- Length of cache
  - Within one (exactly one) request-response cycle (0.5 - 2 sec) - Very safe
  - Across multiple request-response cycles (10 sec - 5 minutes) - Source of “bugs”
- This covers the multiple request scenario
  - App-aware caching / event bus caching
  - Hibernate / EHCache style

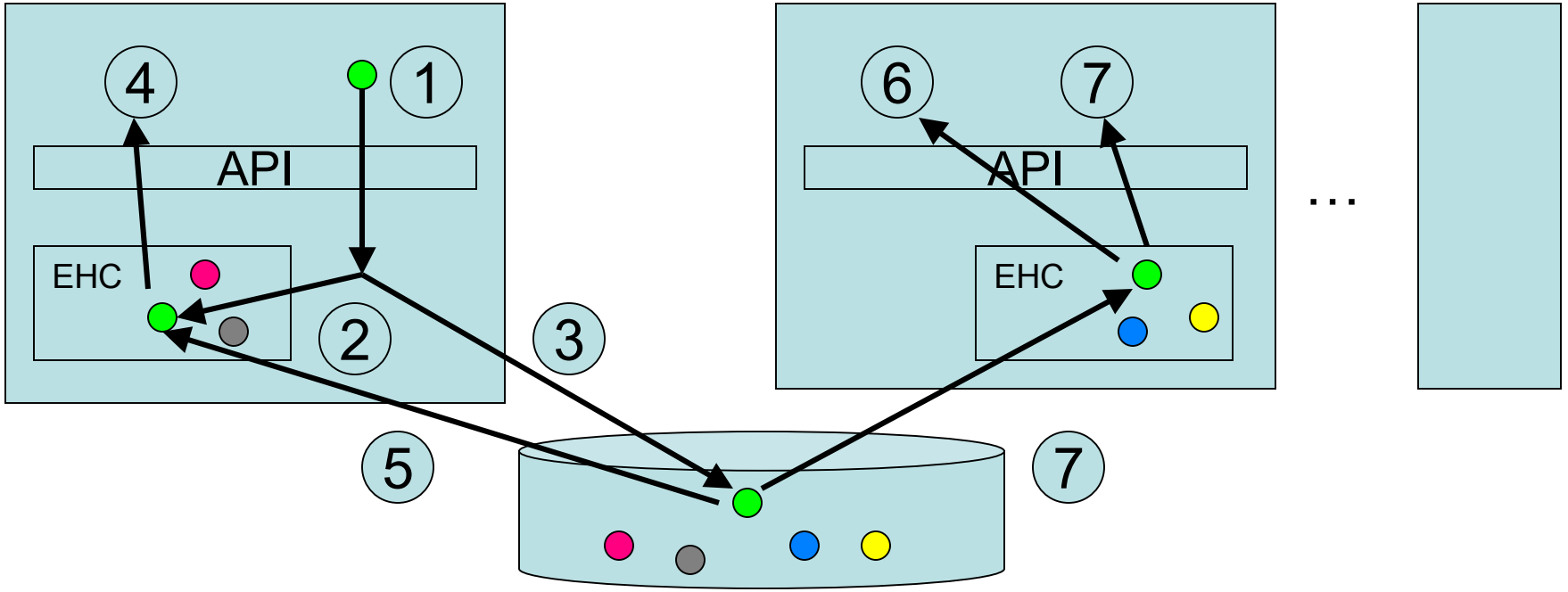
Sakai App-Aware Cache



1) Some tool modifies an object and saves it via the API
2) The local cache is invalidated
3) An event is generated indicating which object was modified
4) The event is delivered across cluster causing invalidation as necessary

Note: TTL is still on the order of 3 minutes to keep caches small and fresh

Hibernate / EHCache



1) Object is modified in a node
2) It is invalidated in the local cache
3) It is updated in the DB
4) When a new request happens,
5) It is properly re-retrieved
6) Until TTL expires, requests come from cache on other nodes and get bad data
7) When TTL expires and a request is made for the object
8) It is properly re-retrieved from disk

Possible Future Directions

- Improve Event Implementation
  - Use JGroups instead of Database
    - Early experiments worked well
    - Must solve the issues in dynamic cluster configuration
    - Likely to improve scaling nicely
- Have Hibernate use Cluster-aware caching
  - Will need careful performance analysis - coordination in large clusters could easily overwhelm benefits
  - oscache, jbosscache, swarmcache (From Josh)

Recommendations

- TTL of < 10 seconds in EHCache or any other non-cluster aware cache - Even things that seem “read-mostly” are dangerous to cache - because when these change often they are broad settings like “test start time” that are important to propogate quickly.
- A short cache TTL approximates “request scope caching” as well as “app-aware caching”
- Developers may want a TTL of zero to insure that they truly are encouraged to look at generated SQL.

Summary

- Increasing EHCache TTL as a performance tuning approach is very bad
  - Developers on single systems will *never* encounter bugs
  - Synthetic tests in clustered environments may not catch bugs
  - Users will catch bugs when making changes because of user-support or student request - user behavior causes worst-case need for synchronization
- Sakai's app-aware caching is not *easy*
  - Requires the use of events
- Improve performance of applications which use Hibernate by looking at the data model and SQL and improving it - not just using EHCache