NYOUG

TANGOSOL

Improving Scalable Performance Using Clustered Caching

Cameron Purdy
President
Tangosol, Inc.

NYOUG, 9 June 2005

Copyright 2000-2005 by Tangosol, Inc.

NYOUG

TANGOSOL

Introduction

- **Cameron Purdy**
  - President and founder of Tangosol
  - Contributor to Java and XML specifications
  - JCache (JSR107) specification lead
- **Tangosol Coherence**
  - Leading clustering and grid computing software for Java and J2EE, with hundreds of successful production deployments.
  - Coherence enables in-memory data management for clustered J2EE applications and application servers, and makes sharing, managing and caching data in a cluster as simple as on a single server.

Copyright 2000-2005 by Tangosol, Inc.

NYOUG

TANGOSOL

Predictable Scalable Performance

- **Defining "Scalable Performance"**
  - **It is *NOT* just focused on making an application faster!**
  - **Performance: Overall response times for an application are within defined tolerances for normal use and remain within those tolerances up to the expected peak user load.**
  - **Scalable: There is an understanding of the required resources to support additional load without exceeding those tolerances.**
  - **Predictable: There can be certainty that additional resources will handle the load.**

Copyright 2000-2005 by Tangosol, Inc.

NYOUG

TANGOSOL

Predicting Scalable Performance

- **Measuring Scalable Performance**
  - **Resource Usage: Quantifying the effect of each user**
  - **Concurrency: Determining how additional users affect the scalability of an application**
  - **Complexity: Understanding the number of tiers involved in servicing a request, and how often each is involved**
  - **Cost: The effects of each of the above on cost**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Obstacles to Scale

- **Resource Usage: Later Tiers**
  - **There is a cost when a tier invokes a later tier**
  - **Collocation of tiers reduces inter-tier communication**
  - **Applications that have to talk to the database on each request will suffer from scalability problems**
  - **The Database tier is *difficult* and *expensive* to scale; it is difficult to scale a database server to more than a single host, and it becomes exponentially more expensive to add CPUs**
  - **Database servers scale sub-linearly *at best* with additional CPUs, and there is a CPU limit**

Copyright 2000-2005 by Tangosol, Inc.

Obstacles to Scale

- **Summary**
  - **Architect so that the application is CPU- or memory-bound, and that the bottleneck is in the application tier at the latest**
  - **For high-scale applications, make sure that the bottleneck will never be the data source (mainframe service, database)**
  - **Benefit: You can use server farms and server clusters to scale an application almost linearly and with a predictable cost per user**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Clustering for Scale

- **Clustering enables multiple servers or server processes to work together**
- **Clustering can be used to horizontally scale a tier, i.e. scale by adding servers**
- **Clustering usually costs much less than buying a bigger server (vertical scaling)**
- ***Clustering also typically provide failover and other reliability benefits***

Copyright 2000-2005 by Tangosol, Inc.

Clustering for Scale

- **Clustering Categories**
  - **Master/Slave: For availability**
  - **Parallel: For scalability, e.g. stateless web server farms**
  - **Centralized: Single server for coordination *(can represent bottleneck and/or SPOF)***
  - **Hierarchical: Multi-tiered centralized model**
  - **Peer-to-Peer: Servers work independently, but have knowledge of and direct access to the entire cluster *(cooperative worker model)***

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Clustering for Scale

- **Primary benefits of Clustering:**
  - **If the application has been built correctly, it supports a predictable scaling model**
  - **Clustering allows relatively inexpensive CPU and memory resources to be added to a production application in order to handle more concurrent users and/or more data**
    - **Increase application throughput**
    - **Increase the in-memory data capacity of the application**
  - **Uses redundancy to improve availability**
    - **Simple (n+1) model**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Clustering for Scale

- **The Potential for Negative Scale**
  - **Single server mode often permits unrestricted caching ..**
  - **.. since clustering may imply the disabling of caching ..**
  - **.. two servers often handle less load than one!**
- **Data Challenges in a Clustered Environment**
  - **Maintaining the data in sync in the cluster is the biggest challenge for applications that have to cache read/write data**
  - **If multiple application update the data source, then the app caches need a way to stay in sync with the persistence tier**
    - **Triggers, Polling and custom App-To-App integration**
  - **How to failover servers without losing in-memory data?**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Clustered Caching for Scale

- **Common uses for Clustered Caching**
  - **HTTP Session Caching for stateful applications**
  - **Page, Document and Segment Caching**
  - **Application Data Caching: Your Own Java Objects (YOJOs ;-)**
  - **Load Balancing of Data Operations**
    - **Information Fabrics**
    - **Compute Farms**
    - **Offloading XML Transformations**
  - **Dramatically reduce database load by using read-through, write-through and write-behind caching**

There is no better way to increase scalability than to use caching to unload later tiers!

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL™

Clustered Caching for Scale

- **Cache Coherency**
  - **A cache that is "coherent" shows the same contents at every location within a distributed or clustered environment**
  - **Caches of read-only data are automatically coherent!**
  - **The choice for clustered caching of read/write data:**
    - **Accept a certain amount of data staleness**
    - **Maintain cache data coherency across the cluster**
  - **Clustered data coherency implies a means to *synchronize*:**
    - **Clustered concurrency control (like Java "synchronized")**
    - **Distributed Transactional Caching**
  - **Interposing the data caches *between* the application logic and the data source prevents loss of consistency**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

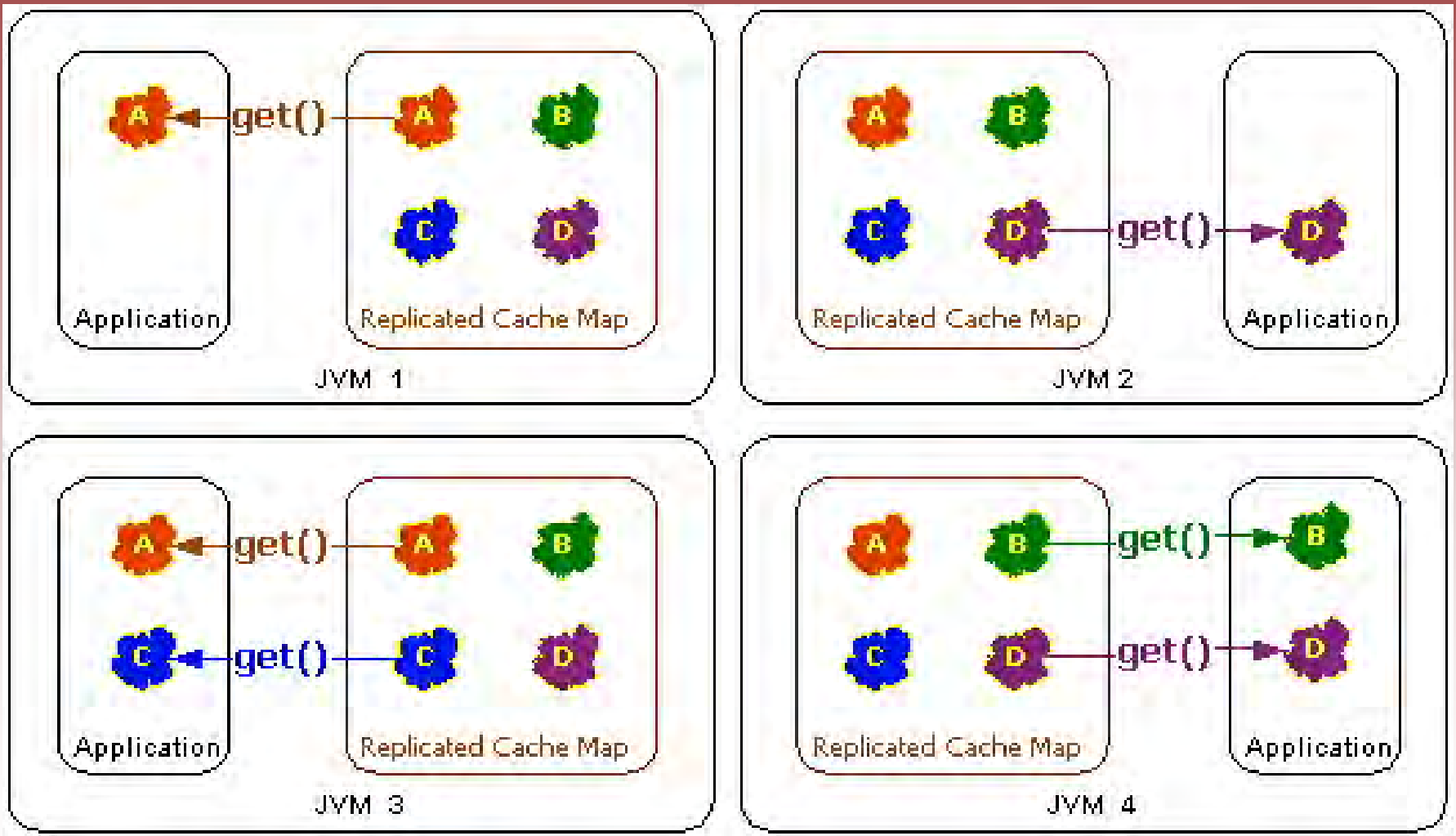
Replicated Caching

- ***Challenge*** **: Extreme Performance.**
- ***Solution*** **: Cache Data is Replicated to all members of the cluster.**
- ***Zero Latency Access*** **: Since the data is replicated to each cluster member, it is available for use without any waiting. This provides the highest possible speed for data access. Each member accesses the data from its own memory.**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Replicated Caching



Access to a Replicated Cache

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

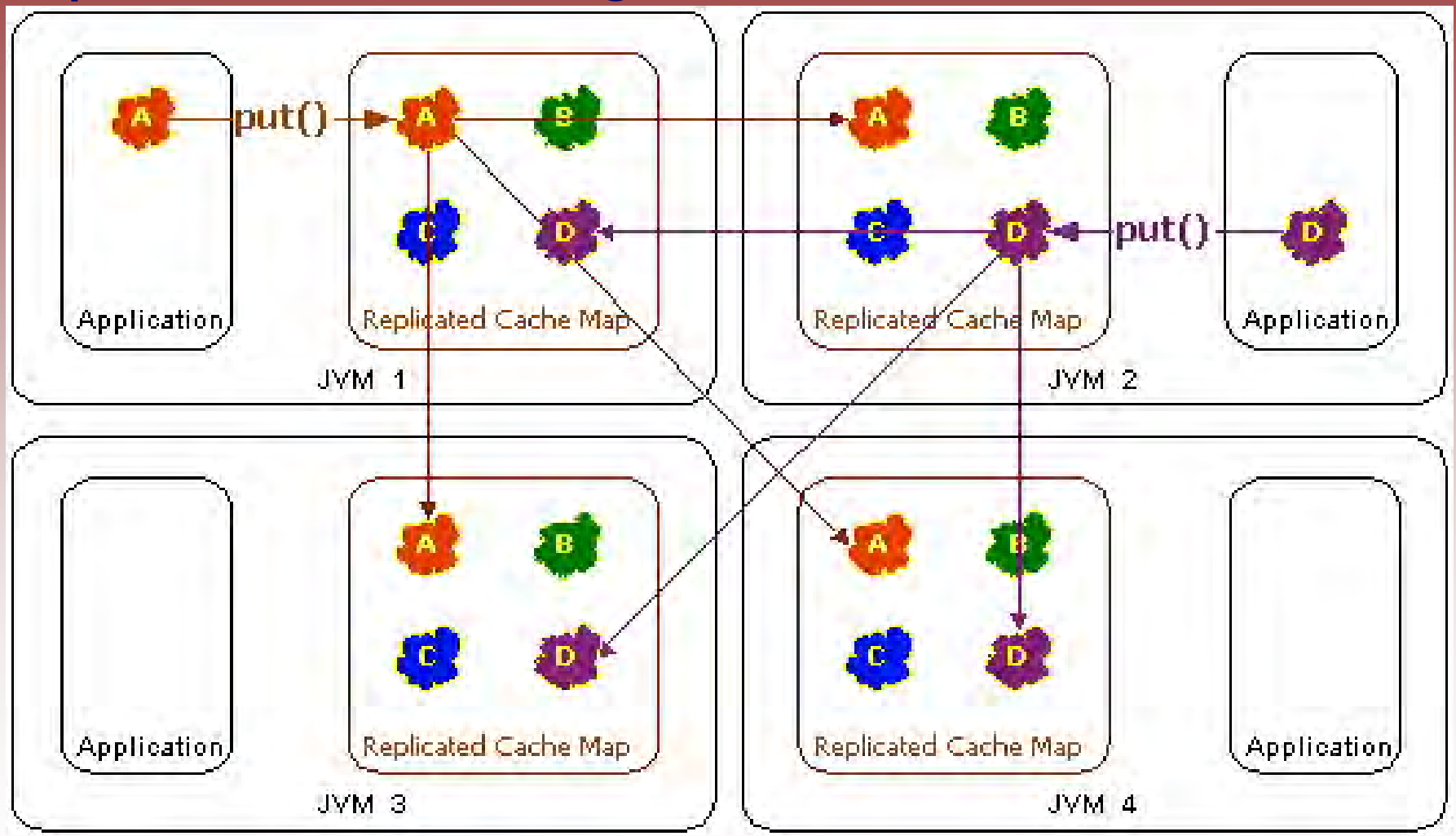
Replicated Caching

So, what's the catch?

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Replicated Caching



Update to a Replicated Cache

Copyright 2000-2005 by Tangosol, Inc.

Replicated Caching

- ***Scalability Limits with Replicated Caching***
  - ***Cost Per Update*** **: Updating a replicated cache requires pushing the new version of the data to all other cluster members, which will limit scalability if there are a high frequency of updates per member.**
  - ***Cost Per Entry*** **: The data is replicated to every cluster member, so Java heap space is used on each member, which will impact performance for large caches.**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Replicated Caching

So, how to solve the scalability issue?

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL™

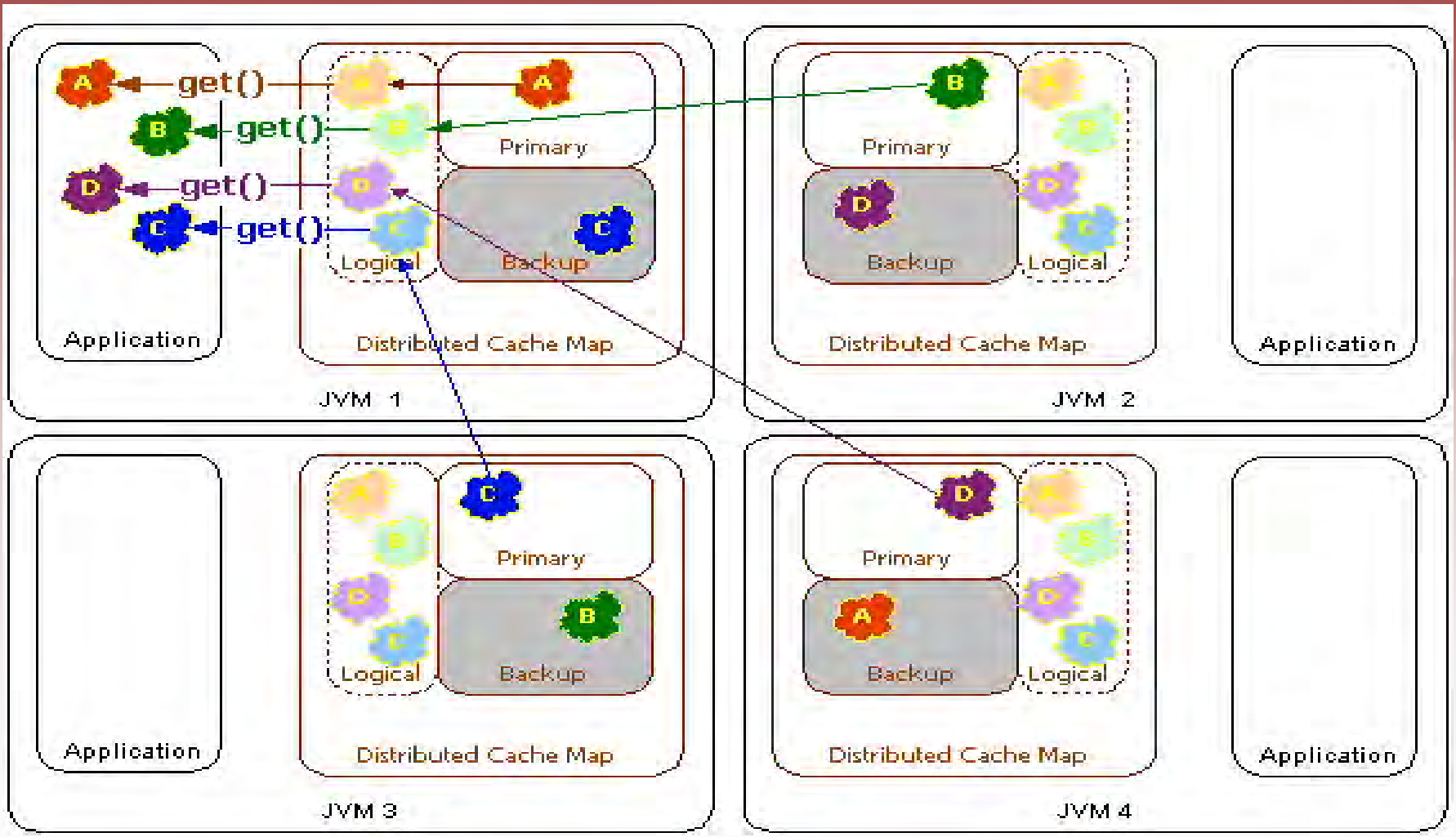
Partitioned Caching

- ***Challenge*** **: Extreme Scalability.**
- ***Solution*** **: Transparently partition the Cache Data to distribute the load across all cluster members.**
- ***Linear Scalability*** **: By partitioning the data evenly, the per-port throughput (the amount of work being performed by each server) remains constant.**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Partitioned Caching



Access to a Distributed Cache

Copyright 2000-2005 by Tangosol, Inc.

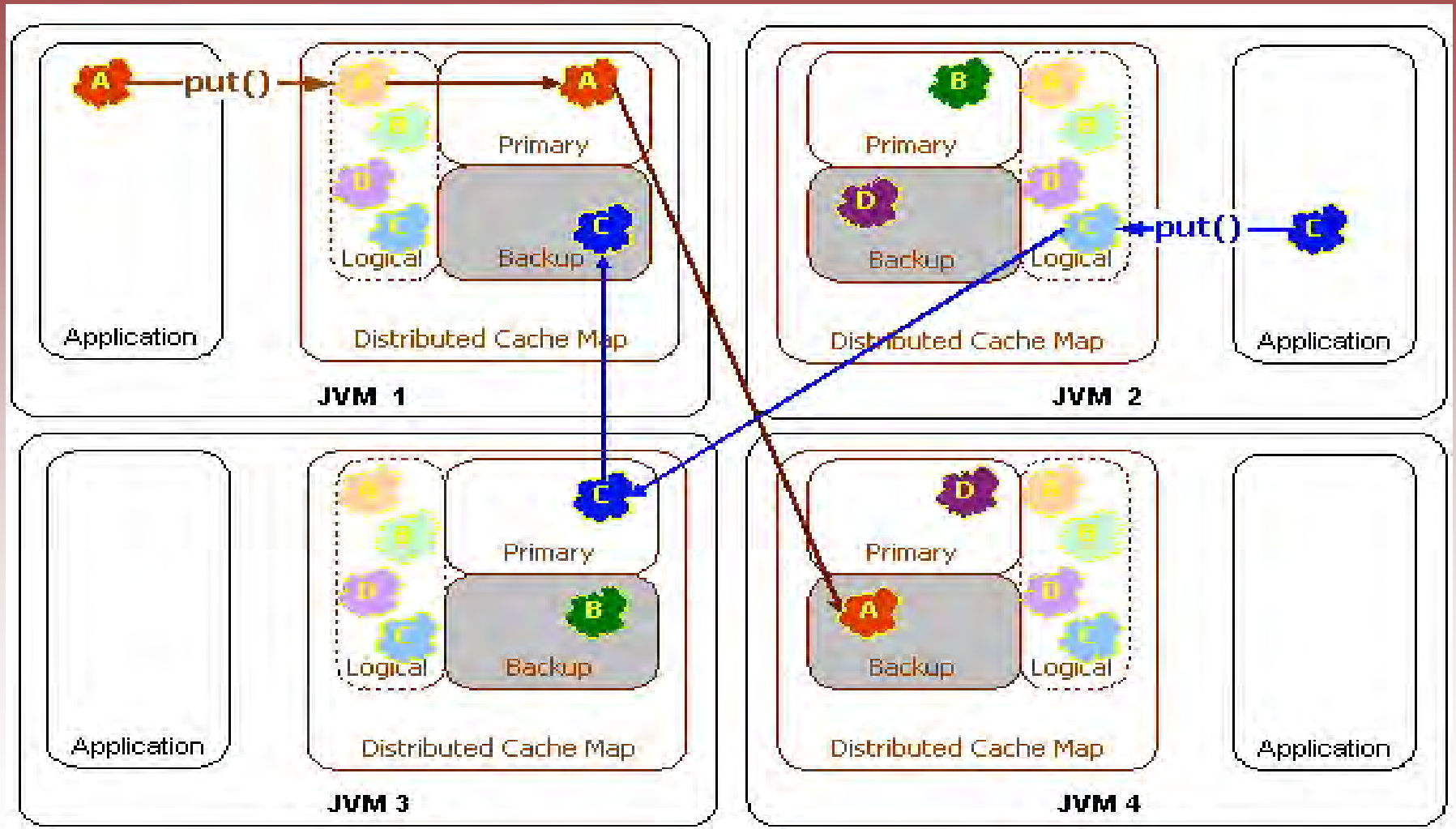
Partitioned Caching

- **Benefits of partitioning**
  - ***Scalability*** **: The size of the cache and the processing power available grow linearly with the size of the cluster.**
  - ***Load-Balanced*** **: The responsibility for managing the data is automatically load-balanced across the cluster.**
  - ***Ownership*** **: Exactly one node in the cluster is responsible for each piece of data in the cache.**
  - ***Point-To-Point*** **: The communication for the distributed cache is all point-to-point, enabling linear scalability.**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL™

Partitioned Caching



Update to a Distributed Cache

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

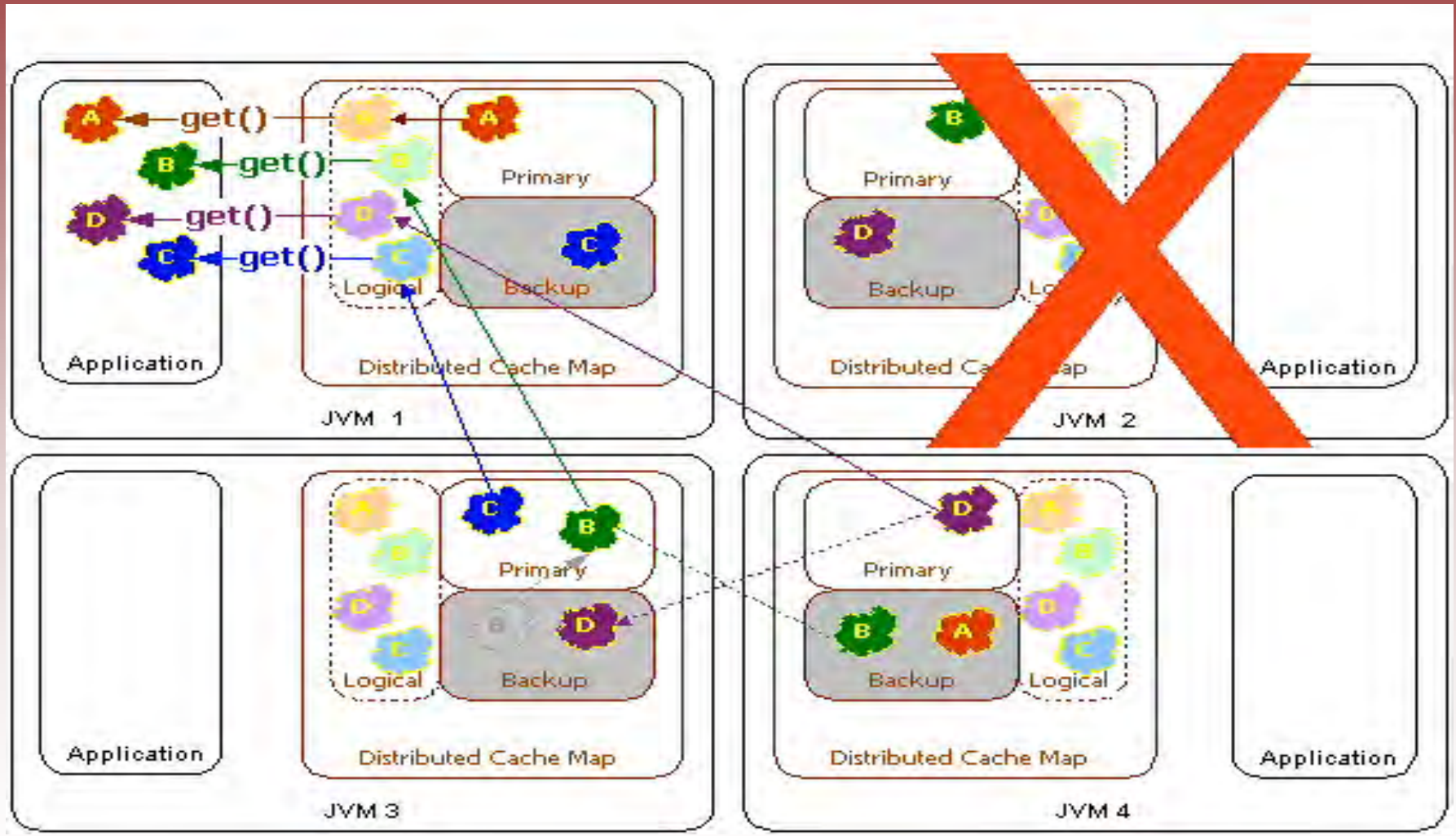
Partitioned Caching

- ***Failover*** **: Cache services can provide failover and failback without any data loss, and that includes partitioned caches:**
  - **Configurable level of redundancy (backups)**
  - **Any cluster node can fail without the loss of data.**
  - **Data is explicitly backed up on different physical servers**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Partitioned Caching



Failover of a Distributed Cache

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

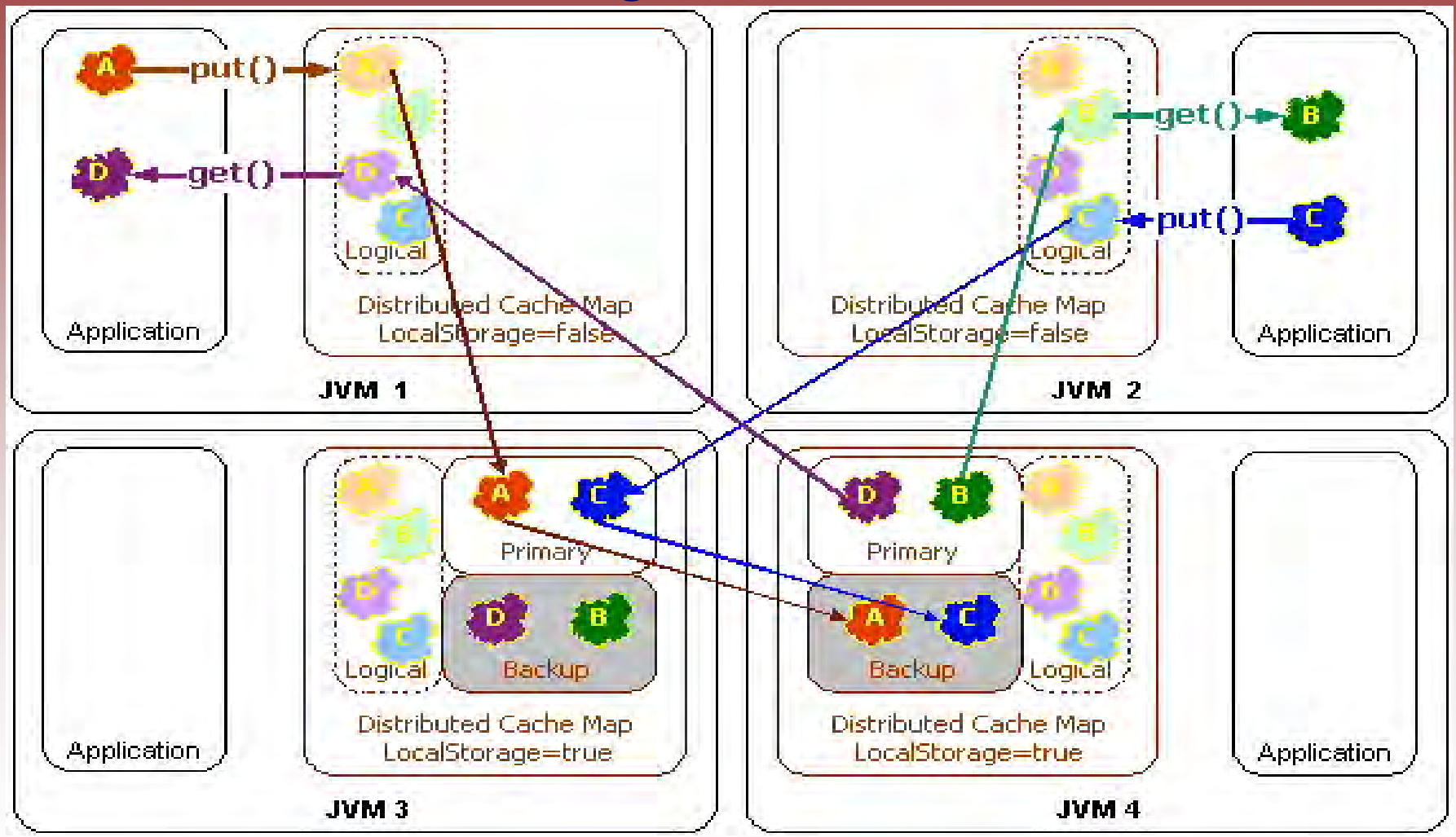
Partitioned Caching

- ***Local Storage*** **: Cluster nodes with local storage enabled will provide the cache and backup storage for the distributed cache. Cluster nodes with local storage disabled will still have the same exact view of the data, even though they are not actually managing any of the data.**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Partitioned Caching



Storage configuration of a Distributed Cache

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Partitioned Caching

So, how to solve the latency issue?

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

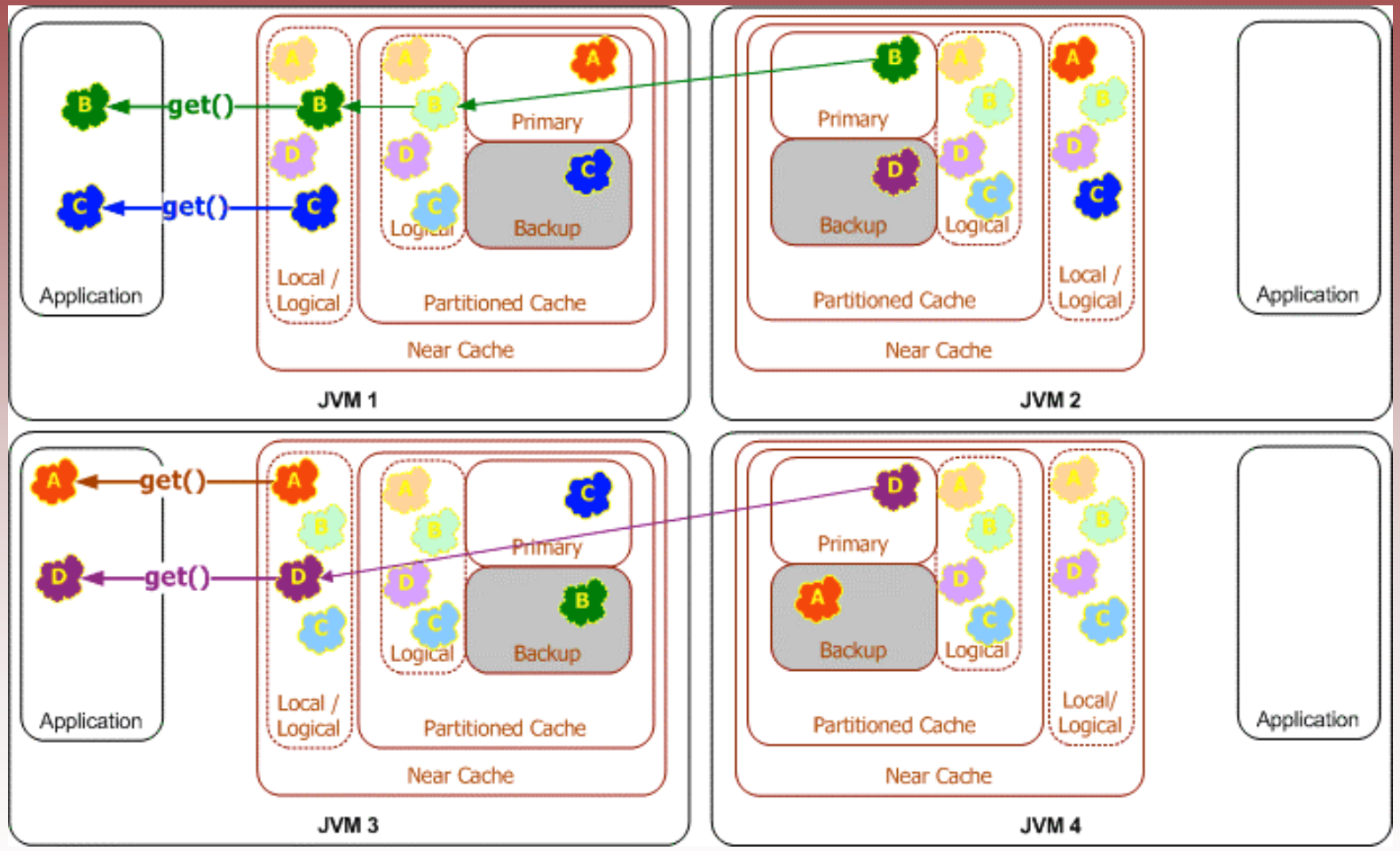
Near Caching

- ***Challenge*** **: Best Scalable Performance.**
- ***Solution*** **: Add in-memory performance to distributed cache scalability.**
- ***Coherency*** **: Provides a number of cache-invalidation strategies, including simple expiry and event-based invalidation.**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Near Caching



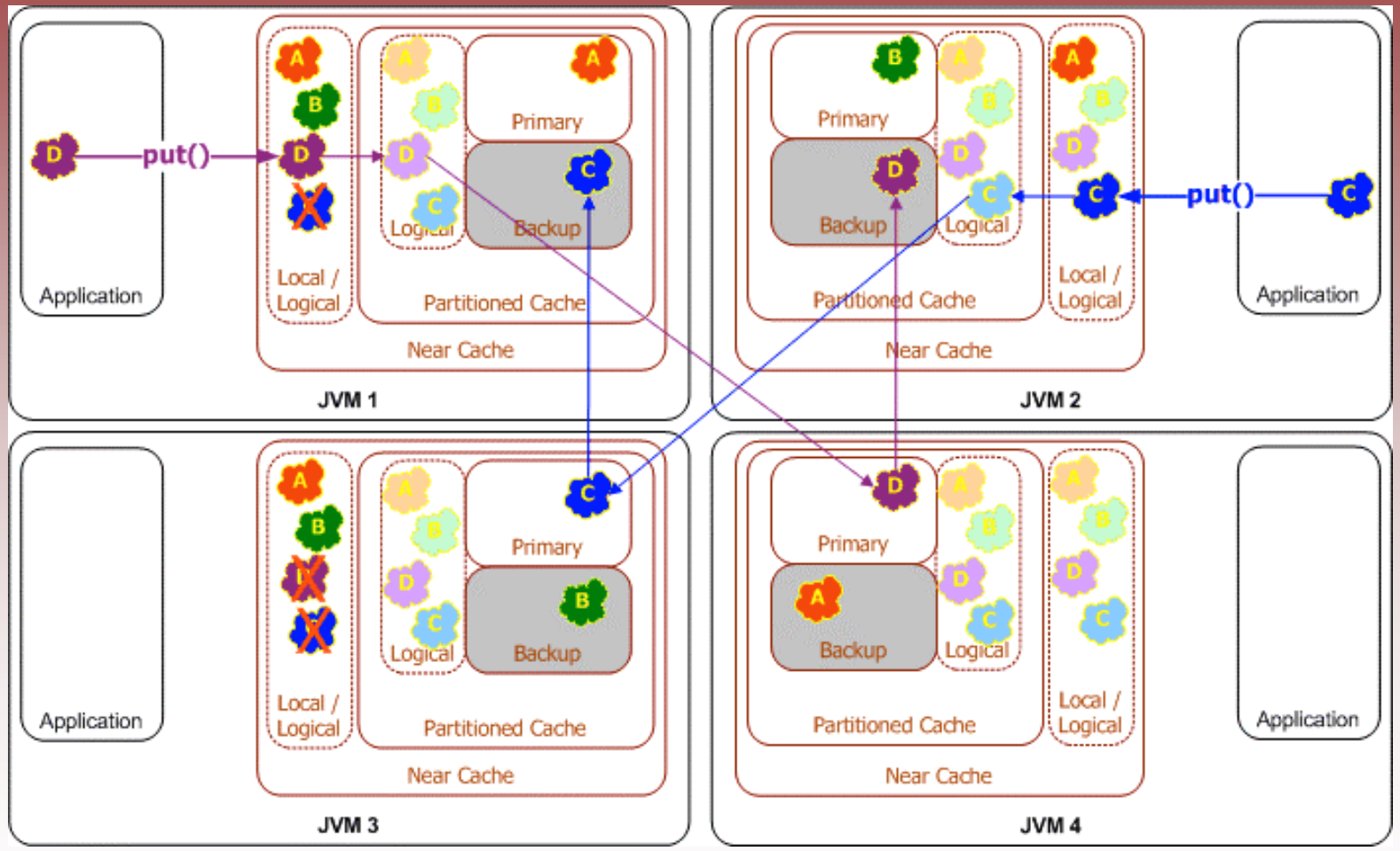
Access to a Near Cache

JVM1 needs to load values to the 'front local cache', JVM3 already has them loaded

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Near Caching

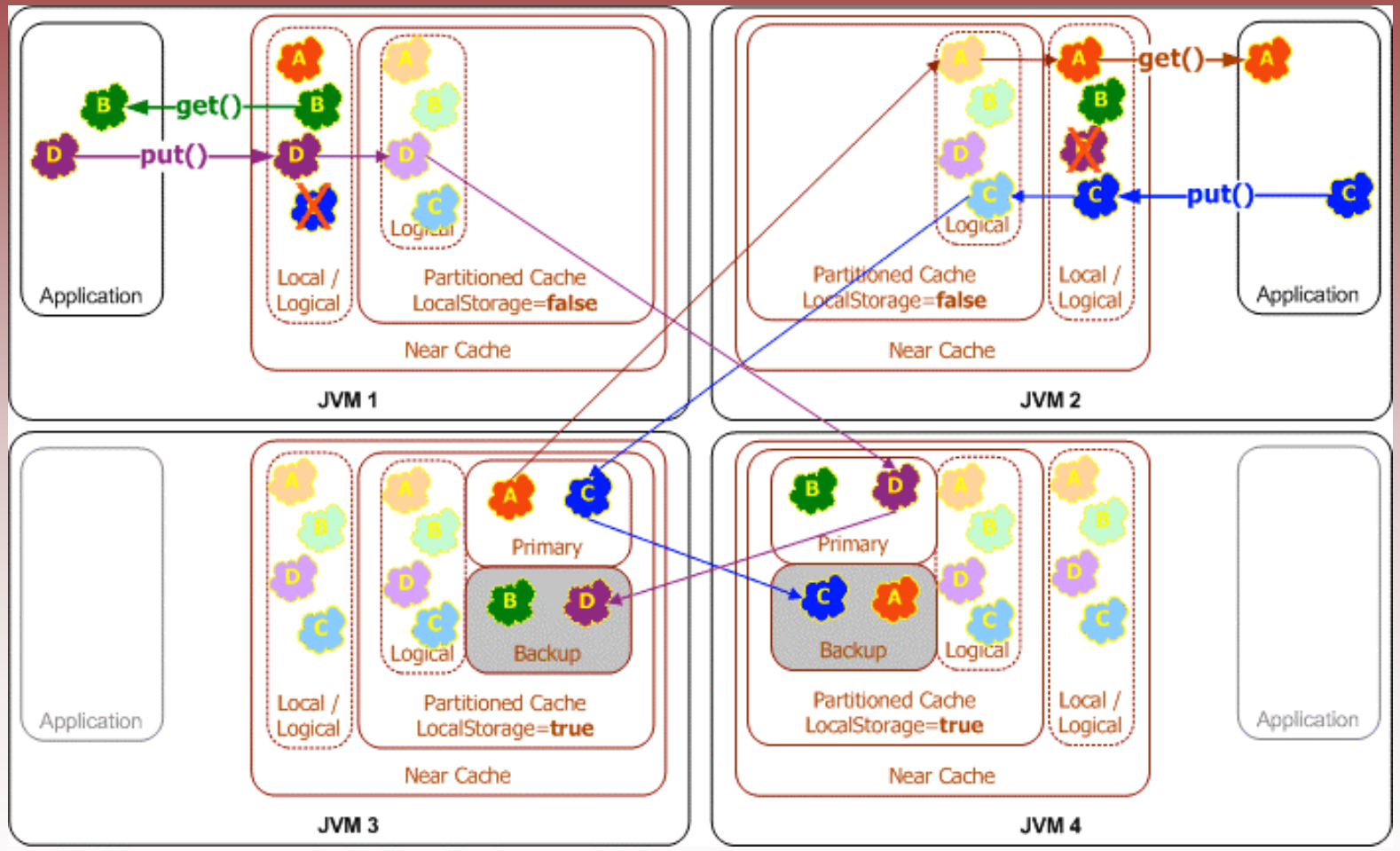


Update to a Near Cache

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Near Caching + Storage Enabled Option



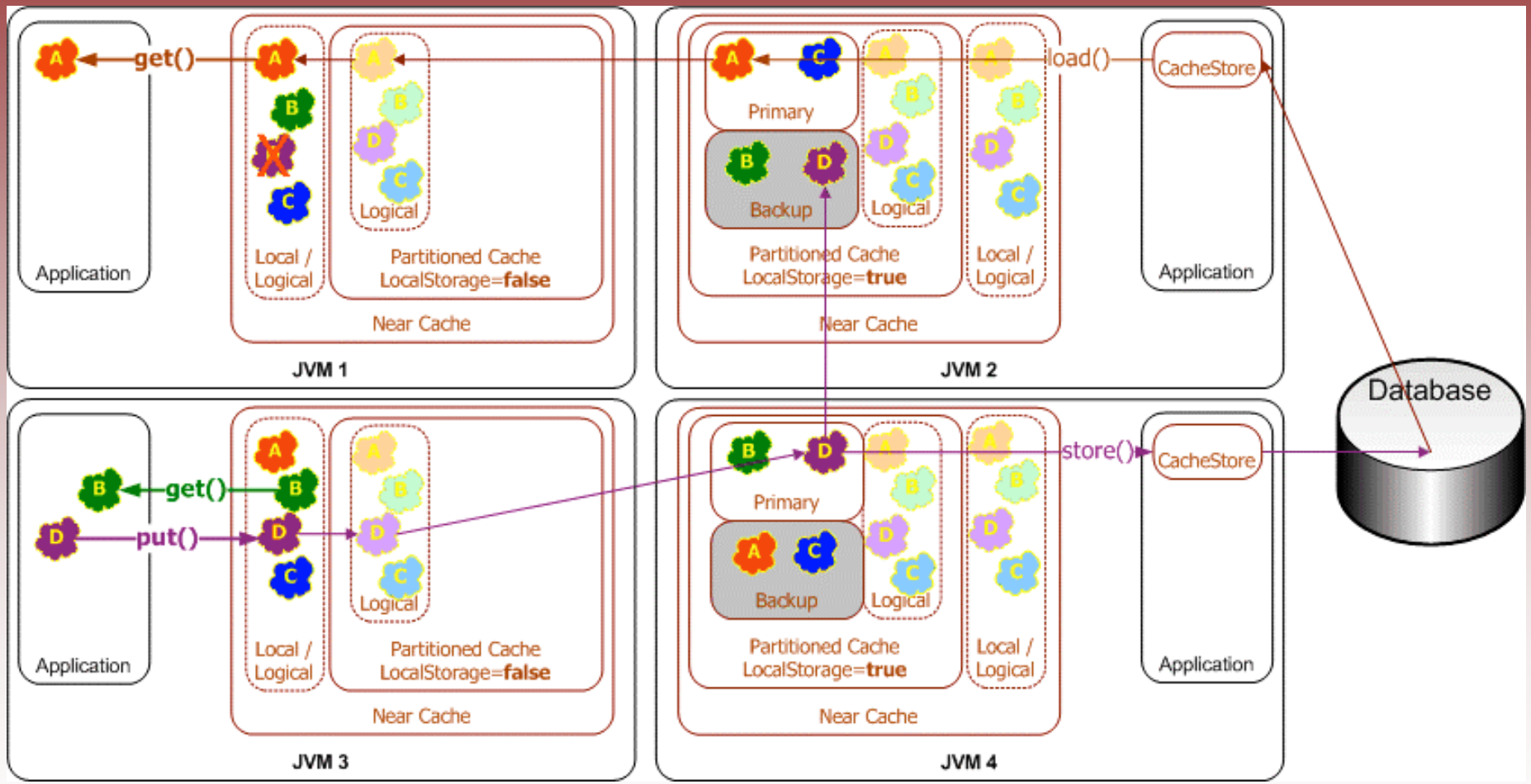
Update to a Near Cache

Copyright 2000-2005 by Tangosol, Inc.

NYOUG

TANGOSOL

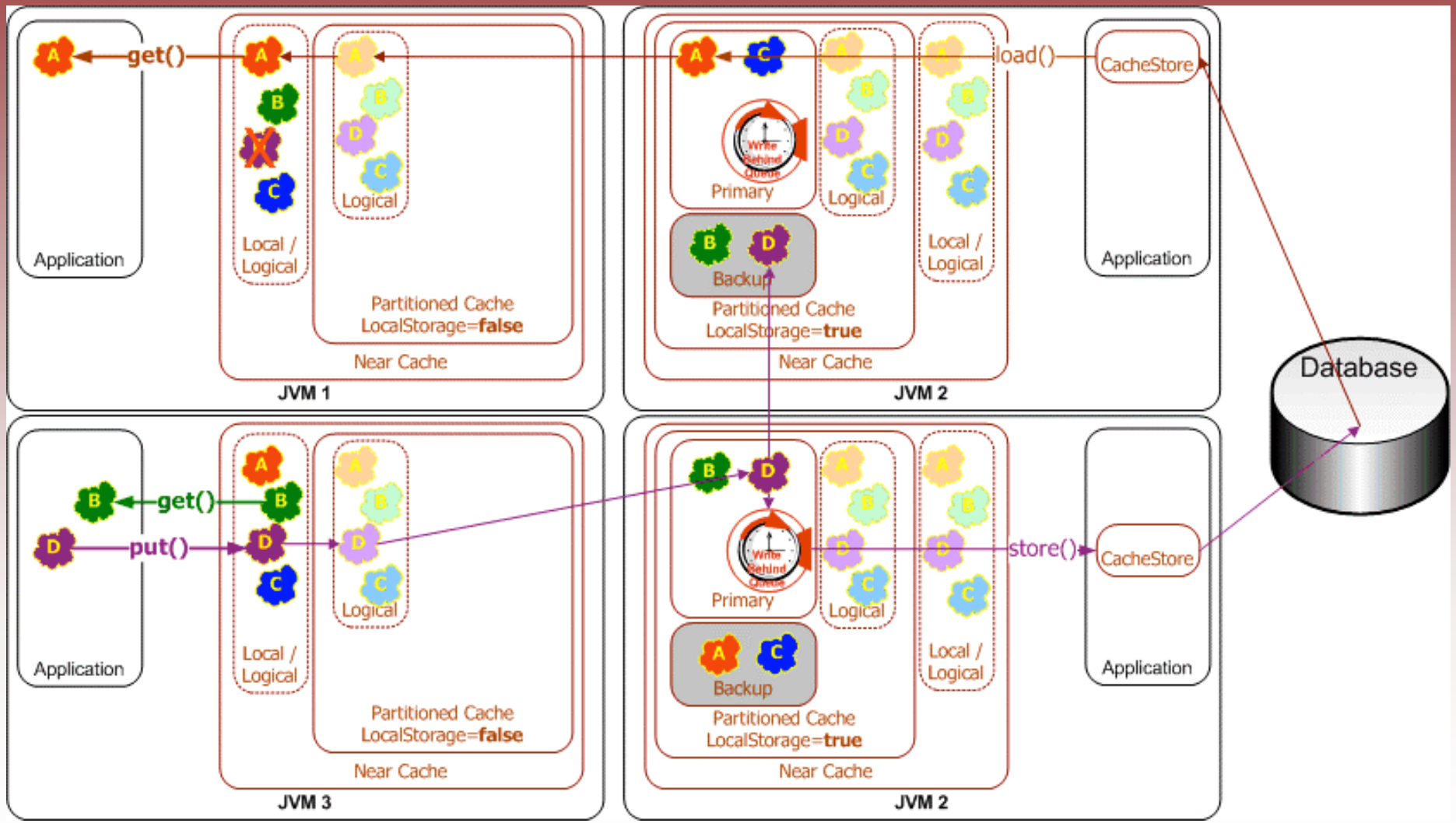
Read-Through / Write-Through Caching



Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Write Behind Caching



Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Applications to WS Infrastructure

- **WS requires HA and Scalability, just like any other line-of-business process**
- **Stateful WS conversations require scalable, reliable state management**
- **WS often require large extents of data (lots of reads, lots of joins), meaning that WS requests have many opportunities for cache optimizations**
- **WS can use significant transform cycles, which is a task that begs for a transform farm**

Copyright 2000-2005 by Tangosol, Inc.

TANGOSOL

Clustered Caching Summary

- **Clustering provides reliability through redundancy, and scalability by horizontal scale**
- **Applications that delegate all state management to the database will not scale well**
- **Clustered caching can significantly reduce the back-end load, resulting in scalable performance**
- **Decoupling the application from the back end (using caching, clustered data, write-behind and JMS) can help make applications Highly Available**

Copyright 2000-2005 by Tangosol, Inc.

NYOUG

TANGOSOL

Improving Scalable Performance Using Clustered Caching

Cameron Purdy
President
Tangosol, Inc.

NYOUG, 9 June 2005

Copyright 2000-2005 by Tangosol, Inc.