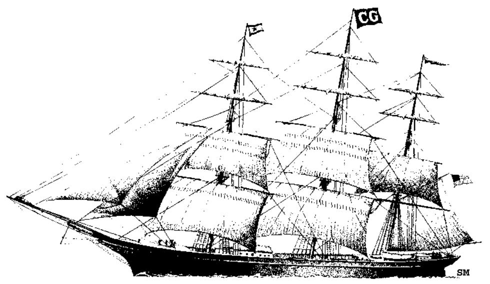
THE CLIPPER GROUP
Navigator™

Navigating Information Technology Horizons℠

Published Since 1993 Report #TCG2004098 December 17, 2004



Real-Time Enterprise IT Building Blocks — Tangosol's *Coherence* Clustered Data Caching

Analyst: Anne MacFarland

Management Summary

For many enterprises, adding self-service capabilities to business-process applications, such as e-commerce and logistics, have made interaction with customers, partners, and employees more convenient. It has reduced the need for data entry and the expense of staffing positions for which demand is uneven. However, adding self-service also has distributed the points of demand on the databases that support these enterprise applications, and made the volume of these requests less predictable. **All of a sudden, access to these databases has become a bottleneck – one that is hard to manage by traditional means because the nature and extent of the bottleneck is unpredictable.** Caching at the database, while sometimes helpful, is insufficient. The huge growth of data contained in these databases supporting these self-service endowed applications exacerbates the problem**.** When you add in the need for real-time synchronization, the parallelization of SMP, scale out, and grid architectures can become part of the problem, not the solution. There is a need to have various pieces of data more usefully at hand.

The answer lies in memory – memory that, happily, is cheaper than it used to be. More specifically, the answer lies in cache, but not any old cache. Memory cache has been used as a ready-reference scratch pad for applications. It has been optimized to flush writes to disk at the first sign of system failure to prevent data loss. However, **in today's integrated, transaction-oriented environments, we need cache to be even more intelligent and fault tolerant, and we need to cluster cache into a globally coherent resource that can be used in increasingly important scale-out environments.** While we need to support the ACID properties of good transactions (see box, next page), we cannot afford the cost of supporting them by involving disks in the process. **More has to be done in cache.**

While creating caches within the context of an application is a familiar task to system administrators, this wider use of cache is something perhaps best left to those who have obsessed about the possibilities and ramifications full-time. Tangosol is one of a small handful of companies who focus on J2EE distributed data caching that underlies application clustering and grid computing. If your J2EE environments are not performing as well as you need, Tangosol, Inc. of Somerville, MA, has the clustered cache tools in its Coherence product to solve many of these problems with a quick ROI in enterprise environments. For more details about their Coherence product line, read on.

IN THIS ISSUE

The Expansion of Technology's Domain

Enterprise computing has moved from supporting straightforward, computationally intense business processes (such as financial reporting and payroll), to supporting less linear business processes and workflows. Because there are many distributed sources of transaction demand, enterprises cannot always control the rate of transactions they must support. In e-commerce environments, they also may need to undo transaction commitments to critical systems like inventory, while still maintaining the integrity of the transactional environment.

Many enterprises have turned to scale-out J2EE environments, where multiple servers run a particular application, to give resilience needed to address the ebb and flow of supply and demand. Together with the Web Services and XML (and its variants) that tie processes together, J2EE offers a way to integrate business process components, and to reuse both these components and the information that supports them, in ways somewhat similar to how business operations worked in the days of paper-based information used by a capable team of intelligent people. However, J2EE runtimes have little tolerance for protracted sessions, and supporting multi-stage transactions can generate a performance-sapping volume of messages. And, in J2EE environments, as in human environments, adding more nodes can make the messaging overhead worse rather than better, due to the distance between the nodes and the need to coordinate them.

Expanding the Functionality of Cache

The concept of cache has been around for as long as there has been something on which to write a note for use later. The concept of *intelligent cache*, in the form of omniscient assistants and secretaries, has been a key part of business strategies for centuries. When you ask your spouse where some household item is located, you deem that spouse your cache. Cache is a familiar concept, but, until now, caches have received little respect and marginal management.

In its most basic form, an electronic data cache is a ready-reference instance of data convenient to whatever is using it. There are "lower level" caches on disk (and even on tape), when only that degree of convenience is required, but the cache that is most often thought of is a section of CPU memory assigned for holding information that may become needed by a process, for many processes are apt to use a piece of information repeatedly.

Exhibit 1 - ACID Basic Transaction Properties

Atomicity - The transaction must be fully committed or not. There can be no middle ground.

Consistency - A transaction must render valid data, either by producing new valid data or by returning data to its previous state.

Isolation - A transaction must be isolated from other transactions while it is in process. (See Exhibit 3 on Page 4 for details.)

Durability - In order for a transaction to be durable and binding, the data must be flushed from cache to some more durable form of data storage, where system failure will not destroy it, producing an ambiguous situation.

Read caches[1] optimize access to information. Here, synchronization is important to assure that the content is current and correct. *Write caches*[2] can optimize transaction throughput by aggregating writes to disk[3]. Here, flushing algorithms assure that data has been written safely to disk, both routinely and in the case of a system failure. *Write-through* synchronously updates disk, while asynchronous *write-behind* allow the application to keep processing when the database is temporarily unavailable. In applications where the nature of what information will be needed can be predicted, read-ahead cache algorithms can speed processing throughput.

Caches can be used to decouple access to data sources, to reduce access times, to evade the computational cost of going through another server,[4] and to avoid duplicating calculations that are often repeated. Cache can also be used as a transmission buffer, but that use is beyond the scope of this bulletin.

1 *Reads* are called *Gets* in cache-speak.

2 *Writes* are known to cache cognoscenti as *Puts*.

3 Except in the case of two-stage transactions, which are not complete until the data has been written to disk.

4 This is important when the data source has processor-cycle-based utility pricing, like mainframe MIPS.

Exhibit 2 - Types of Cache

- ***Local Cache*** - Local cache is the on-heap, non-clustered convenience of a ready reference shelf. It has instant read and write but no fault tolerance. It may require more garbage collection.
- ***Replicated Cache*** - Data is stored in every node, fully synchronized. This is a kind of pro-active cache used for small, often-accessed information. It gives the fastest read, but it uses a lot of memory on each machine.
- ***Optimistic Cache*** - Optimistic cache is like replicated cache, but with no locking (coherence is preserved by queuing) to improve performance. It is good for read-intensive situations.
- ***Distributed Cache*** - Data is partitioned. For fault tolerance, data can be kept in the memory of multiple machines (a la RAID). This form of cache also scales linearly and uses less memory acreage than replicated cache.
- ***Near Cache*** - This is a hybrid cache that fronts a fault-tolerant distributed cache. The Near Cache asynchronously invalidates cache entries, providing very fast (but not quite real-time) cache synchronization. This is used in combination with another form of cache.

Cache Management

Management of data in memory is similar to management of data on disk or other media, but at another order of magnitude. There are topologies and cache policies[5] to determine how the cache will work. (See Exhibit 2, at the right.) There are structures for redundancy and procedures for locking that ensure the transaction isolation that underlies data integrity. (See Exhibit 3, on the next page.) There are configuration options to specify transaction atomicity.

Cache Trade-Offs

The world of cache is a world of trade-offs. The size of cache extends capability, but also impedes cache performance. Intelligence enables more and different kinds of things to be done, but also slows process. So, building real-time responsive IT environments is more than a masonry exercise in completeness, for each application will have its own profile of acceptable trade-offs that must be tailored to. Design-it-yourself may not be the best strategy.

Tangosol *Coherence*

Tangosol's Coherence is a tool to build and manage distributed cache in ways that have been shown to improve transaction performance in hundreds of large J2EE environments. It is implemented as an API that exposes a set of commonly used caching and cache management functionalities as simple XML options[6]. There are three editions.

The *Clustered Edition* offers basic cache construction and cache management. The *Enterprise Edition* adds:

- Support for distributed queries, write-through, and write-behind caching for database applications.
- Clustered invocation services to bring the application process to the data in data-center grid computing architectures. (See the section *Uses for Coherence* on the next page.)
- Web-performance sticky-session load-balancing.[7] This cache intelligence offloads the need to load balance from the Apache Web server layer, a streamlining that is important in environments with massive transaction loads.
- J2CA Transactional Caching support enables two-phase commit to as well Tangosol's fault-tolerant redundancy in the cache cluster. The data becomes *cluster durable* (the D in ACID).

The free one-year license to the *Development Edition* gives full access to all Coherence features for development purposes only. Developers often find that the Coherence API lets them postpone the choice of cache structures until deployment, which allows their software to be more flexible.

[5] These include cache size, fullness ramifications, object longevity, replication (including a snapshot, for in cache, as on disk, recovery is faster using the index of an image than in a classic rebuild through slow pipes.), distribution of data updates, and notification of any updates to servers that request them.

[6] A CLI option is also available.

[7] In sticky load balancing, each node takes responsibility for certain accounts (as opposed to the random assignations of usage-based load balancing).

Exhibit 3 -
Transaction Isolation Levels

- ***Read/Get Uncommitted*** allows access to data from an uncommitted transaction. These are called dirty reads. (*Coherence does not support this loosest level. It is included for completeness of the spectrum.*)
- ***Read/Get Committed*** involves shared read locks. It prevents dirty reads, but does not insure that the data will not be changed by another transaction.
- ***Repeatable Read/Get*** locks the data used, so another transaction cannot update it, generating a non-repeatable write. This prevents both dirty reads and non-repeatable reads, but not phantom reads of new information added by other transactions in the meantime.
- ***Serializable:*** This most restrictive level of isolation executes a transaction as a series, no matter what order the transaction elements are received, preventing phantom reads. It is usually accomplished by locking the data set until the transaction is complete. Another approach is to track the assumptions of each transaction element, and commit at the end, rather like the coherence of optimistic cache. (See Exhibit 2). With either approach, the effort of maintaining concurrency in environments of serializable transactions is less. This is important where the rate of transactions is high.

Coherence's Ecosystem

Coherence is supported on BEA *WebLogic* and IBM *WebSphere* application platforms. It can be used in *Solaris, Linux*, *AIX, z/OS, AS400, HP-UX*, *OS/X,* and *Windows* environments. Coherence can be accessed by *C++* and *.Net* applications via local proxy objects.

Coherence runs on any Java application server (including mixed environments), as well as outside an application server. The cluster can reside on one machine, in a number of Java Virtual Machines, or spread across several machines attached to a single switch, or used in a grid architecture.

Coherence is supported, as a plug-in, by SolarMetric *Kodo*, Hemtech *JDO Genie*, Riflexo *Jcredo*, Hibernate, and Isocra *Livestore*. Tangosol collaborates for integration with Codemesh and JNBridge, as well as offering its own professional services. It has strategic partnerships with BEA, HP, IBM, and Sun.

Coherence Clustering Model

Coherence uses clustering based on the conference room model. All server caches can speak to all other caches, hear all, and detect entry and departure. Because this model is a peer-to-peer rather than a tiered architecture, nodes can also speak to a single node or subset of nodes.

Membership is open to both clients (application servers) and back-end database servers. Subject to security, joining is automatic. Coherence automatically maintains cluster membership and information about damaged cache nodes. Cluster membership uses few resources, so hundreds of clients can be supported (the current architectural limit is over 8000). Since all members are known, the cluster can be made redundant and fault tolerant.

Because Coherence cache is clustered, there is no single point of failure. If a server should fail, failover and a redistribution of cluster management services are automatic, as is the discovery and fail back when a new server is added to the system. Coherence offers a transparent soft restart somewhat akin to the sense-and-respond of autonomic systems. Coherence's TCMP clustering protocol is designed for the unreliable high-latency low-bandwidth of WAN links. Its distributed locking avoids single server bottlenecks. It supports tiered cache. It supports compression and Tangosol will add fault-tolerant WAN routing in the 3.0 release, due out at the end of the first quarter of 2005.

Uses for Coherence

Database Querying

Tangosol Coherence can simply cache data from database tables as Java objects, which can be queried. With a replicated cache (see Exhibit 2), queries are performed locally, and do not use indexes. In situations with distributed caching, the caches themselves can be indexed. The advantage of distributed caching is that the cluster gives more cache capacity and more CPU processing as it grows, speeding throughput. For example, queries in a distributed cache are executed in parallel across the cluster.

Developers can add custom filters for queries, structures for query parallelization, or index-aware filters. Coherence includes a built-in, cost-based query optimizer.

HTTP Session management

As we go from using the Web as a presentation tool to using it as the basis for interactions to satisfy a variety of needs, response time, that rigid parameter of customer satisfaction, will be an even greater challenge. The *Coherence*Web* module uses *NearCache* (as described in Exhibit 2) to offer three options of HTTP session cache management:

- **Traditional**, for use with small HTTP objects
- **Monolithic**[8] for situations where multiple session attributes refer to the same session object (think of mainframes and virtual server-consolidated environments)
- **Split**, where larger attributes are split out and managed individually.

With Coherence, application servers of different hardware platforms and operating systems can share common user session data. This makes amenities like Single Sign On (SSO) a lot easier.

Grid Agent Invocation

Coherence's clustered invocation services (available in the Enterprise Edition) can be used in a data center environment with data-rich applications expecting parallelization. In a flip of the traditional use of cache, instead of caching data near the processor, the invocation services request the application process to run near the cached data. Moving the process to data cached in distributed caches (as shown in Exhibit 2) is much more efficient than supplying data to hundreds of needy, CPU-anchored application clones. Because the data is already partitioned across the cluster, it is, in effect, already load-balanced, so when you send out the processes, they are naturally load-balanced.

The Benefits of Coherence

By managing data across the clustered cache in the application tier, Coherence-assisted J2EE applications can scale higher without a bottleneck developing in the I/O to shared resources such as a database. Because Coherence is a separate entity, rather than part of the application, administrators can manage and modify caches and cache policies without touching the application code. This limits the risk of evolving cache as needed, and lets J2EE environments be optimized more fully and as needed.

While its structures and management is familiar, Tangosol *Coherence* clustered cache does challenge traditional server-storage relationships. Used as part of traditional architectures, Coherence reduces the role of storage as an active participant in transactional environments. Used in data grid environments, the processor becomes a motel-of-convenience for the application. By removing the need to optimize these hardware elements, the use of commodity hardware is facilitated and higher levels of utilization can be more easily supported.

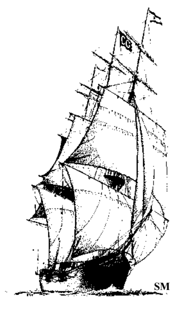
Tangosol Go to Market

Tangosol sees the market for its product as an amalgam of software developers - for whom Coherence shortens time to market, system integrators - for whom Coherence is a fabulous tool, and platform technology vendors in application security, application management and performance management spaces - for whom Coherence is a basic building block. Tangosol has had early success in banking, trading, and insurance environments. It is now looking to raise awareness of what Coherence can do for a wide variety of scale-out application environments.

Conclusion

Tangosol delivers on the vision of a cache that can be optimized for present use and evolved over time, without touching the code of the application. Tangosol's attitude of *Why don't we do it all in cache?* may be just what you need to bring your response times under control. It may be what you must have to address data-intensive, large-scale opportunities that bollix up IT systems with less sophisticated cache.

The advanced caching that Tangosol offers is a basic tool for optimizing the large IT systems that support today's dynamic and interconnected business processes. Consider what some fault-tolerant, Coherence cache clusters could do for your systems!



[8] Available for Coherence 2.5 and thereafter.

About The Clipper Group, Inc.

The Clipper Group, Inc., is an independent consulting firm specializing in acquisition decisions and strategic advice regarding complex, enterprise-class information technologies. Our team of industry professionals averages more than 25 years of real-world experience. A team of staff consultants augments our capabilities, with significant experience across a broad spectrum of applications and environments.

- ***The Clipper Group can be reached at 781-235-0085 and found on the web at www.clipper.com.***

About the Author

Anne MacFarland **is Director of Enterprise Architectures and Infrastructure Solutions for The Clipper Group.** Ms. MacFarland specializes in strategic business solutions offered by enterprise systems, software, and storage vendors, in trends in enterprise systems and networks, and in explaining these trends and the underlying technologies in simple business terms. She joined The Clipper Group after a long career in library systems, business archives, consulting, research, and freelance writing. Ms. MacFarland earned a Bachelor of Arts degree from Cornell University, where she was a College Scholar, and a Masters of Library Science from Southern Connecticut State University.

- ***Reach Anne MacFarland via e-mail at Anne.MacFarland@clipper.com or at 781-235-0085 Ext. 28. (Please dial "1-28" when you hear the automated attendant.)***

Regarding Trademarks and Service Marks

Disclosure