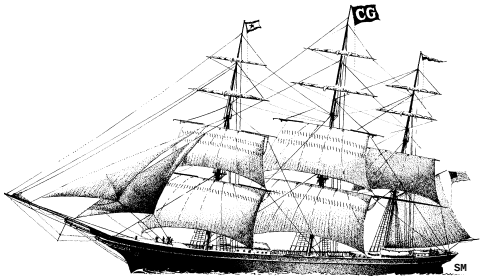
THE CLIPPER GROUP
Navigator™



*Navigating Information Technology Horizons*℠

Published Since 1993 Report #TCG2006063R July 21, 2006

# The Scale-Out Cornucopia — Terracotta Enhances IT Strategies of Plenty

Analyst: Anne S. MacFarland

We are living in a time of plenty, as far as IT capacities are concerned. While products and solutions still cost a pretty penny, the raw costs of memory and disk space, processor clock speeds and network bandwidth, have tumbled. New IT strategies are now taking advantage of this cornucopia of plenty to address remaining areas of dissatisfaction – parallelizing process to achieve better time to completion with architectures like grid[1] and application optimization[2]. Standards like XML, now so ubiquitously used, could not have arisen when the basic set of IT resources – cycles, memory, and networking – were more constrained. One of the remaining concerns is *system brittleness* and how to build in resilience and fault tolerance at the application level. Brittleness is particularly troublesome in the J2EE applications that often underlie Web application functionality – an area of operations that is subject to spikes in demand. Resilience is needed at the session level, in particular, to support constructs such as the shopping carts by which e-commerce is done[3].

The problem has become more urgent because customer-facing applications with self-service elements are inherently prone both to bursts of concurrent use and to lags in response due to user browsing and indecision. Either situation can cause a session to fail. With many options to be considered, sessions have become a complex litany of transactions that sometimes have to be rolled back.[4] The end result of any of these challenges to session functionality is that the user often loses the data he or she has painfully entered, or loses a shopping basket. If this happens frequently, the site loses the customer.

A company called Terracotta, based in San Francisco, CA, has addressed this problem with software that automates session-level clustering. The hardware used for its Terracotta Sessions can be what you have at hand. Their legerdemain takes the plentiful capacities of today's IT systems and addresses a problem that, while a problem of very small, short lived things, affects business competence, business revenue, and business survival.

For more details, please read on.

## IN THIS ISSUE



[1] See *The Clipper Group Captain's Log* dated October 27, 2004, entitled *Grids for Business – Business as Grids*, available at http://www.clipper.com/research/TCG2004087.pdf.

[2] See **The Clipper Group Navigator** dated May 7, 2006, entitled *The Intelligent Person's Guide to Shrinking Application Process*, available at http://www.clipper.com/research/TCG2006032.pdf.

[3] Of course, elements other than shopping carts need to be persisted – but they provide a familiar example of session-level functionality.

[4] For more about the particulars of fault-tolerance in memory structures, see **The Clipper Group Navigator** dated December 17, 2004, entitled *Real-Time Enterprise IT Building Blocks – Tangosol's Coherence Clustered Data Caching*, available at http://www.clipper.com/research/TCG2004098.pdf.

## The Application Tier

The Web application tier is often a prototypical scale-out environment. Small servers or blades (for the transactions do not need symmetric multi-processing capabilities) are deployed with a load balancer in front to distribute workloads to available nodes. The servers apply to back-end systems for the necessary data (and, sometimes, processes). They complete the transaction, and await the next job. Like call center operators, these nodes live lives of transitory intensity.

Sessions are the functional units of this IT transitory intensity – like a customer session is for a call center. As when problems are escalated in a call center, applications time-out if there is too much delay in a particular routine. The application process is restarted, in legacy mainframe platforms, by the operating system. In open (commodity) systems, the operating systems instead invoke a *fail-over* to another instance of the application. This application-scale failover is a big deal. Hardware and applications must be clustered carefully for that contingency, for there are many parts to be recovered in the proper order. Recovery of stalled application processes via virtual machines is simpler, and more congruent with the real-time requirements of Web applications. Recovery of session state, like the transfer of relevant information during a call center escalation, is key to a good experience.

The session state information that is needed to allow a session to fail over has no long-term value. It is the information needed to get a transaction done, not the documentation of the transaction itself. Unlike call center operations[5], there are no dependent benefits of aggregating experience at the transaction level. The challenge is to preserve ephemeral data for session failover without incurring an over-build of functionality that would impair the speed of the process itself. J2EE environments are a bit like the world of quantum mechanics.[6] In quantum mechanics, measurements must be unintrusive and precise, or they distort the situation. Providing resilience to IT applications at the session level similarly requires working at a small scale very carefully.

### *Subtlety*

Any actions to optimize the session environment must be lightweight and non-intrusive. Capturing the information needed to enhance application resilience must not require changes to the application code itself. This is particularly true if the application is deployed across a bank of slightly heterogeneous servers (say, with different patch levels), where touching the code might have to be done multiple times.

### *Atomicity*

The constant nature of the updates needed to maintain current information in the name of failover often is confounded by the size of the information being transferred. You can do frequent, or you can do large, but, as with the measurement of particles, you can't do both. J2EE functions at the object level. Sending a whole object's worth of state every time something changes takes up too much processing power and too much bandwidth. It decreases performance and increases the probability of application timeout. J2EE environments are gossamer networks of independent entities operating at high-speed. The entities are linked in run-time containers to form an application or an application subroutine. They support a high volume of transactions. Anything that causes clogs does a great deal of damage. Thus, the resilience-enhancing solution must have the awareness to be able to extract not only just the necessary objects, but also only the changes within those elements.

## Terracotta Sessions

In the *Terracotta Sessions* solution, the recovery supported is at the session level, and is supported by clustering Java Virtual Machines (JVMs) underneath the application. It is best used to cluster the objects at runtime that are needed for application resilience, as opposed to all objects. By locking at a very granular level within the JVM, it avoids the locking strategy of serialization[7]. The Terracotta strategy scales to support tens of thousands of concurrent users. It is useful in an SOA, where the objects in a JVM can be used by different applications, for it is providing a service — high-availability as a service at runtime. Terracotta gives a service orientation to the Java application stack.

State information is stored on Terracotta

[5] Good call centers often track the nature of customer problems to identify flaws in products and services.

[6] Heisenberg's statement that it is impossible to know, precisely both the position and motion of a particle at the same time is still, generally speaking, true. And, the challenges to it have been accomplished by the same kind of indirection that Terracotta uses as its strategy.

[7] Serialization adds expense in development and deployment and makes application modification difficult. For more details about transaction isolation levels, see Exhibit 3 in the Tangosol bulletin cited on page 1.

Servers, which are hosted on a redundant active-passive server set[8] with storage networked to both servers (usually via SAN). Terracotta drop-in agents on the JVMs lightly monitor the entirety of their business logic. The *Sessions* software pushes all session state changes to the Terracotta Servers at a very granular level, so locking during this state replication is not a problem. At the first sign of session failure, Sessions has the aggregated session information needed to support the quick re-instantiation of the session on another processor. Electronic shopping carts do not disappear. Customers and other end users remain happily unaware that there has been a problem.

Residing within off-the-shelf Java servers, Terracotta software clusters the Java Virtual Machines (JVMs) within an application, not the application itself. As it is only loosely coupled to the application, it can be agnostic about application and operating system particulars. It is quick and easy to install for those familiar with J2EE environments.

The use of Terracotta can benefit more than just the business applications it enhances. The operational use of Terracotta makes development simpler, giving developers one less thing to work into the equation. It simplifies deployment by automating clustering at the session level. The Terracotta cluster tracks and "knows" a lot about the behavior of an application in production and about its data usage. Many Terracotta customers have found this visibility a compelling side benefit. They can use the console to diagnose application problems, and also can use it to identify the best data strategies for their scale-out J2EE runtime environments.

Today, Terracotta can be of more general use as a state-keeper for a Web application tier. Users of Web applications want the best of both worlds – they want interactive functionality and they want to access it on their small, cheap, semi-disposable device. To support interactive functionality, there must be session resilience. Thus, there must be redundant session-level business logic, preferably not co-mingled with the application logic. A state-keeping element may be just what is needed, not only to cluster sessions, but also to keep the segmented environments inherent in the provision of Web applications well regulated. This might allow J2EE environments to grow bigger and richer, instead of just more numerous.

## Conclusion

JVM-level IT resilience matters to business. In addressing the problem of session-level resilience, Terracotta offers a key tool to optimize Web-tier application environments to meet the needs of highly-demanding, self-serving customers. If your enterprise must deal with such people – or is comprised of such people, check it out.

[8] This hardware sits in the application tier on separate boxes or blades. In theory, a Terracotta server could sit as a virtual machine on the Application Server. To date, customers have preferred to have them housed separately. Over time, Terracotta will support active-active and larger clusters of Terracotta servers as Web applications develop the sophistication to benefit from it.

### 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 Data Strategies and Information 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. 128. (Please dial "128" when you hear the automated attendant.)***

### Regarding Trademarks and Service Marks

**The Clipper Group Navigator**, **The Clipper Group Explorer**, **The Clipper Group Observer**, **The Clipper Group** *Captain's Log*, **The Clipper Group Voyager**, and *"clipper.com"* are trademarks of The Clipper Group, Inc., and the clipper ship drawings, "*Navigating Information Technology Horizons*", and "*teraproductivity*" are service marks of The Clipper Group, Inc. The Clipper Group, Inc., reserves all rights regarding its trademarks and service marks. All other trademarks, etc., belong to their respective owners.

### Disclosure

Officers and/or employees of The Clipper Group may own as individuals, directly or indirectly, shares in one or more companies discussed in this bulletin. Company policy prohibits any officer or employee from holding more than one percent of the outstanding shares of any company covered by The Clipper Group. The Clipper Group, Inc., has no such equity holdings.

### Regarding the Information in this Issue

The Clipper Group believes the information included in this report to be accurate. Data has been received from a variety of sources, which we believe to be reliable, including manufacturers, distributors, or users of the products discussed herein. The Clipper Group, Inc., cannot be held responsible for any consequential damages resulting from the application of information or opinions contained in this report.