Enterprise Edition Scalability

eCommerce Framework Built to Scale

Reading Time: 10 minutes

Broadleaf Commerce Scalability

About the Broadleaf Commerce Framework

An enterprise commerce framework based on best of breed open source technologies

Broadleaf Commerce (Broadleaf) provides companies with a platform for building high performance commerce solutions. Based on best of breed open source technologies including the Spring framework, Broadleaf was designed from the ground up to be extensible and scalable for businesses and institutions requiring a mission critical eCommerce solution.

Broadleaf is an open source framework that can be used by any developer needing a commerce solution. Broadleaf's Community Edition includes basic commerce features and administrative tools in a modern development framework suitable for small business needs. Broadleaf Enterprise Edition adds features needed by enterprise customers along with performance and scalability improvements.

This paper provides details of scalability tests performed with the Broadleaf Commerce Enterprise Edition. Testing was completed on average equipment (2 core servers with 4 GB RAM) using real world test methodologies. With the ability to easily scale to hundreds of transactions per second across tens of thousands of concurrent users and millions of products, the test results speak for themselves.

We would like to thank our friends at Rackspace (www.rackspace.com) for their contribution of cloud hardware and services for this load test study. Please see Appendix C for details on the Rackspace configuration used to achieve these results.

Test Methodology

Simulating real-world shopping scenarios with industry average conversion rates

Simulating real-world scenarios when testing an e-commerce application is critical in determining not only how efficiently the software performs under normal circumstances, but also how many users it can serve during peak demand. Though difficult to conduct, the test yields the best real world results.

While testing up to a 100% conversion rate to ensure performance on "Black Friday" and "Cyber Monday" behavior, most test cases conducted an average of an 11% add-to-cart action and an aggregated 3% conversion rate. Furthermore, a 300ms pause time was simulated for each checkout transaction, indicative of payment processing times.

In all cases, Broadleaf set out to report objective scalability numbers. In isolation, test cases for "home page views" or "orders" have no merit outside of simulated consumer behavior. The ability for a system to handle hundreds of millions of views to a single page without any other variable is a useless statistic in itself. Furthermore, test cases with varying concurrent user numbers hold no value unless tested against concurrent user behavior, not just hits to a website.

Testing could only be considered a "pass" if the Broadleaf framework responded with an average response time less than one second. Broadleaf detailed test result details (Appendix H) show specific measures of transactions per second (txns/sec) and actual response times against comprehensive page-level test details, server specifications and product catalog sizes.

Test Results

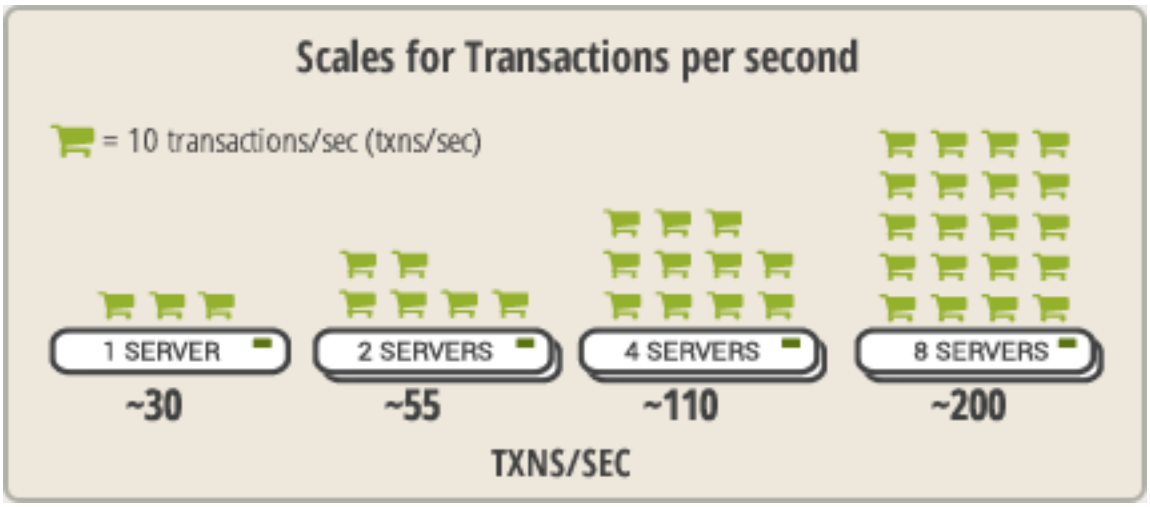
Against high traffic, large product catalogs and peak season spikes, Broadleaf Commerce proves the ability to handle the most stringent scalability situations

Peak demand is defined based on industry, customer base and seasonality. Through all major peak eCommerce variables, Broadleaf proves the ability to scale. Across high transaction volume, concurrent users, large catalogs and high conversion rates, Broadleaf exhibits consistent peak performance.

Test 1: High Transaction Volume

For test purposes, transactions have been defined as web interactions such as "view product page" or "add to cart," with a total of 24 pre-defined web interactions defined in Appendix A. Simulating linear scale as servers are added, Broadleaf easily handles roughly 30 transactions per second on a single server and 200 transactions per second in an eight server deployment of the system.

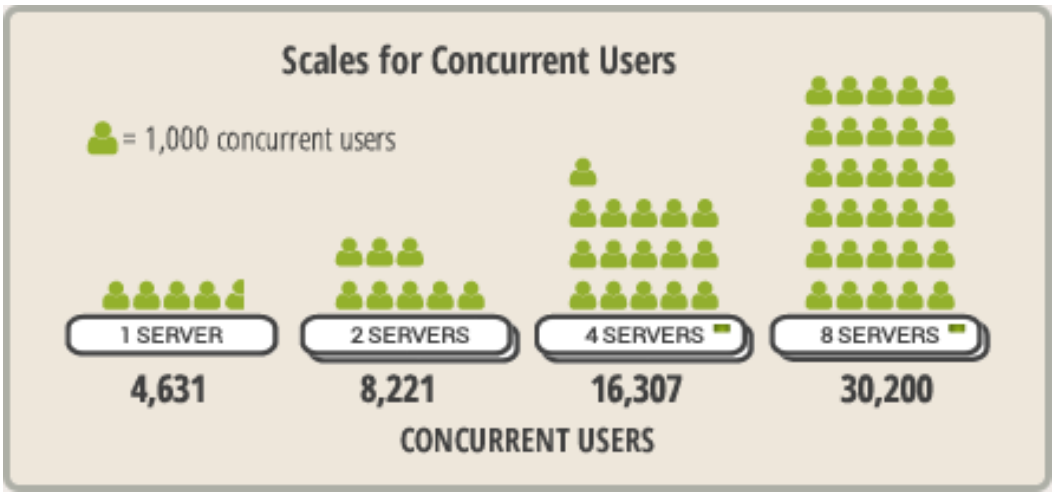
Broadleaf demonstrated horizontal scalability, handling 200 transactions per second



Test 2: Concurrent Users

In order to simulate transactions per second in a real world scenario, Broadleaf tested tens of thousands of concurrent users across architectural tiers from one to eight server scenarios. While many scalability studies determine concurrent users independent of transactions, Broadleaf's tests used concurrent user counts in order to prove transactions per second (Test 1), demonstrating realistic online traffic scenarios.

Broadleaf demonstrated the ability to handle tens of thousands of concurrent users

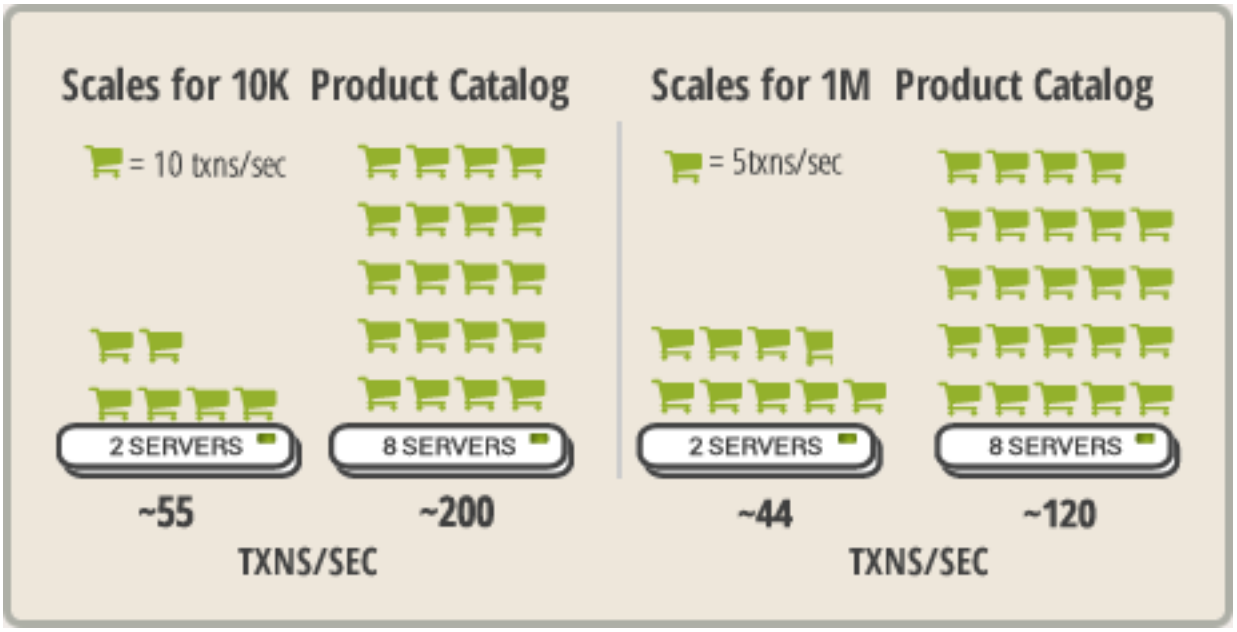


For companies needing more simultaneous threads (e.g., companies that want to be the next Amazon, Facebook, or Twitter), infrastructure build out and serious application performance tuning can be handled with Broadleaf's Professional Services. For most businesses with typical conversion rates, the numbers demonstrated above can handle sites generating **billions** of dollars in sales.

Test 3: Large Catalogs

For corporations requiring larger catalog sets, Broadleaf tested an online catalog with 1,000,000 products. While Test 1 and 2 covered a catalog of 10,000 products in achieving linear scalability, Broadleaf further proved linear scalability using a catalog of 1,000,000 products, using the same commodity hardware and test cases across the product catalog sizes.

Broadleaf demonstrated linear performance across catalogs of 10,000 and 1,000,000 products



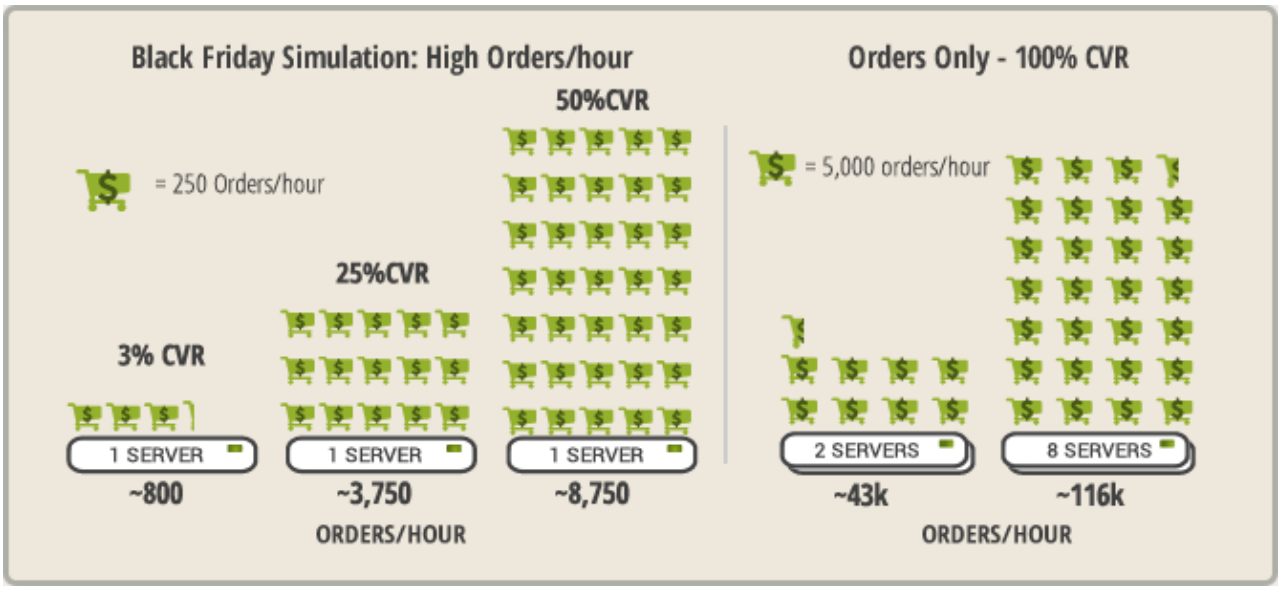
While larger catalog sizes predictably have an impact on Broadleaf's ability to handle transactions per second, hardware scaling proved to be linear. Broadleaf did not 'top out' or otherwise plateau at any combination of catalog size, concurrent user or transaction throughput test. For enterprise clients looking for immediate scale or plenty of room to grow, Broadleaf demonstrates proven proficiency.

Test 4: High Conversion Rates

For corporations with increased conversion rates above industry averages, Broadleaf tested up to 100% conversion in proving, once again, linear scale. While Tests 1 through 3 assumed an industry average conversion rate of 3%, the system was pushed up to a 100% conversion rate.

With a unit of measure being orders per hour rather than web transactions per second, a single server instance easily handles 8,750 orders per hour assuming a 50% conversion rate on site traffic, simulating conversion rates retailers see during an event such as Black Friday. In exploring the total number of orders able to be processed in an acceptable threshold (under a second), Broadleaf's two server configuration handles approximately 43,000 orders per hour, with eight servers processing approximately 116,000 orders per hour.

Broadleaf demonstrated linear performance across high conversion rates, easily handling order spikes



Performance Considerations and Recommendations

Best practice infrastructure and general configuration guidance should be followed

Caching

Caching is an important aspect of tuning a Broadleaf Commerce implementation for performance. Turning on the Thymeleaf cache and tuning the Hibernate Level 2 cache configuration should be considered, as both are beneficial to increasing the efficiency of the application. The default configuration of Broadleaf Commerce includes catalog cache configuration with long time-to-live values and large cache sizes, which should be enabled for optimum performance (explained in Appendix D).

Ehcache is the recommended Hibernate level 2 cache for Broadleaf. It is readily supported by Hibernate, backed by Terracotta, and can be expanded in a number of interesting way to help with individual cases. Ehcache can be turned into a distributed cache, as well as expanded to use "Big Memory." Big Memory is a Terracotta product utilizing off-heap memory space to store cache records, which can help with garbage collection costs normally associated with large in-memory caches. With large product catalog or inventory caches, Big Memory should be an architectural consideration.

Finally, it is recommended to configure specific applications with a single, standalone Hibernate level 2 cache per node, rather than distributed cache. Distributed caches tend to be overly chatty over the network and are generally slower. For the same result, a JMS queue approach that is notified of any cache eviction event is recommended. Every node listens to the queue and can handle its individual cache grooming, assuming the requirement is only near real-time updates of cache (correct in most cases).

Session Affinity

The recommended configuration for Broadleaf Commerce is to enable session affinity so that once a conversation has started with a user, the user will be sent back to the same node for each subsequent request while the session is still active. Centralized session solutions that attempt to store all user state in one location, while flexible, are slower because user state must be constantly pushed to the central location. In addition, the effectiveness of the Hibernate level 2 cache for Customer and Order is negated when the user does not return to the same server during the session.

Database

Broadleaf Commerce can be used with a variety of database platforms. The scalability tests described in this paper were executed using a MySQL database. Broadleaf supports open source databases including MySQL and Postgres. Supported commercial databases include Oracle and Microsoft SQLServer. Tuning any database to be able to handle expected demand is an important step in readying a Broadleaf Commerce implementation for production. Configuring max connections and table buffers are required, in addition to other vendor specific tweaks. Appendix D contains specific tuning notes for MySQL that were used in this test. Consult specific vendor documentation for more information on database tuning and configuration.

Search

Broadleaf Commerce uses Solr as its search engine. By default, Solr is configured to run as an embedded server. This serves well for smaller catalogs, since the memory and cpu requirements are small. And with the embedded server, there is no HTTP connection overhead, so overall it runs quite fast. At larger product catalog sizes (30,000+), we recommend configuring your Broadleaf Commerce implementation to refer to an external, standalone Solr instance (or cluster). Otherwise, the memory and CPU requirements are disruptive to the overall application performance.

CDN

The load test did not retrieve graphics or other static assets from the application container, nor did it engage the built-in asset server for any managed asset retrieval. Most high volume sites will benefit from having static assets, including images, JavaScript, and other media, be delivered to users via a CDN (Content Delivery Network). CDN solutions offer hi-speed nodes located across the world with delivery of your assets coming from the nodes closest to a given user. This serves to reduce response times for your application and lessens unecessary load on your application container. Rackspace offers an easy-to-use CDN for such purposes.

Third Party Integrations

The load test accounted for a simple, third party integration during the submit order process by including an arbitrary 300ms wait time to represent a call to a payment processor.

Typical enterprise eCommerce systems can contain ten or more integrations. Each integration has the potential to negatively affect the scalability of the system. It is important to use best practices for integrating with other systems to ensure that one poorly performing third party integration does not bring down the entire system.

Custom Code

The Broadleaf Framework is highly extensible. The customizations made for each implementation will require additional resources (e.g., CPU and Database).

The specific implementation may use the system in a way that differs from the scalability tests as we designed them.

It is recommended that all medium to large businesses utilizing Broadleaf Commerce execute their own scalability and performance tests.

Conclusion

The Broadleaf Commerce framework scales linear across all testing metrics to meet the needs of even the most demanding eCommerce sites

Well known retailers and businesses depend on Broadleaf Commerce to power their eCommerce solutions, as Broadleaf provides best-in-class eCommerce capabilities at the highest value. Broadleaf proved real business use cases across multiple scenarios, demonstrating:

- Hundreds of transactions per second
- Tens of thousands of concurrent users
- Millions of products
- Billions of dollars worth of sales

Furthermore, the provided test results demonstrate that Broadleaf Commerce scales horizontally by adding additional servers. This type of scaling is ideal for cloud based and virtualized environments. The tests showed scalable results for systems utilizing between one and eight application servers that had catalog sizes ranging from 10,000 items to 1,000,000 items.

Broadleaf's Enterprise Edition meets the scalability needs of the large majority of companies utilizing commodity hardware, providing by far the best overall value among enterprise eCommerce solutions.

For more information on Broadleaf, please visit: www.broadleafcommerce.com
For more information on Rackspace, please visit: www.rackspace.com

Appendix A – Detailed Test Plan

The data below provides a detailed list of the flows that constituted the test plan

URLs and Percentage Weighting[1] for Standard Ecommerce Test

1. Home Page – 100%
2. Category Page Browse – 98%
3. Search Page – 60%
4. Product Page Browser – 98%
5. Product Page Add To Cart – 11%
6. Add To Cart Page Without Product Options – N/A
7. Add To Cart Page With Product Options – N/A
8. View Cart Page – 5.4%
9. Start Checkout Page – 5.4%
10. Anonymous Checkout page - 3%
11. Registered Customer Login Page – 2.4%
12. Registered Customer Login Submission Page – 2.4%
13. Shipping Option Page – 4.7%
14. Address Information Entry Page – 4.7%
15. Pay Using Credit Card and Complete Checkout Page – 3%

URLs for Order Test

16. Product Page Add To Cart
17. Add To Cart Page Without Product Options
18. Add To Cart Page With Product Options
19. View Cart Page
20. Start Checkout Page
    a. Anonymous Checkout page
    b. Registered Customer Login Page
21. Registered Customer Login Submission Page
22. Shipping Option Page
23. Address Information Entry Page
24. Pay Using Credit Card and Complete Checkout Page

All tests were run on the Broadleaf Commerce Heat Clinic demonstration application using Broadleaf Commerce Enterprise Edition version 2.2.1. See Appendix B for hardware configurations.

Notes

1. The system added an artificial payment processor latency of 300 ms during the submit order step
2. The test environment was put through a brief warm-up period before starting to capture metrics. Each test was run until throughput stabilization was achieved.
3. Throughput is represented as samples per second. Order throughput is measured as orders per hour.
4. JMeter was used to generate virtual user load in Master/Slave configuration to maximize virtual user efficiency.
5. Catalog data at 10,000 and 100,000 and 1,000,000 sizes was generated artificially using data generation software.
6. SSL termination was utilized at the load balancer. This configuration would normally be inappropriate for a cloud installation of Broadleaf Commerce because of security concerns, but is suitable for our wider goal of load testing.
7. A load test is considered successful when aggregate response times for all pages register near or under 1 second for 90% of requests.

Appendix B – Hardware Configuration

Broadleaf's Rackspace deployment used Ubuntu 12.0.4 LTS as the operating system with the minimum configuration tested being 4GB Ram with 2 cores

Web Servers

- Apache 2.0 (4 GB RAM, 2 Cores)

Application Servers

- Tomcat 7.0 (4 GB RAM, 2 Cores)

Search Server

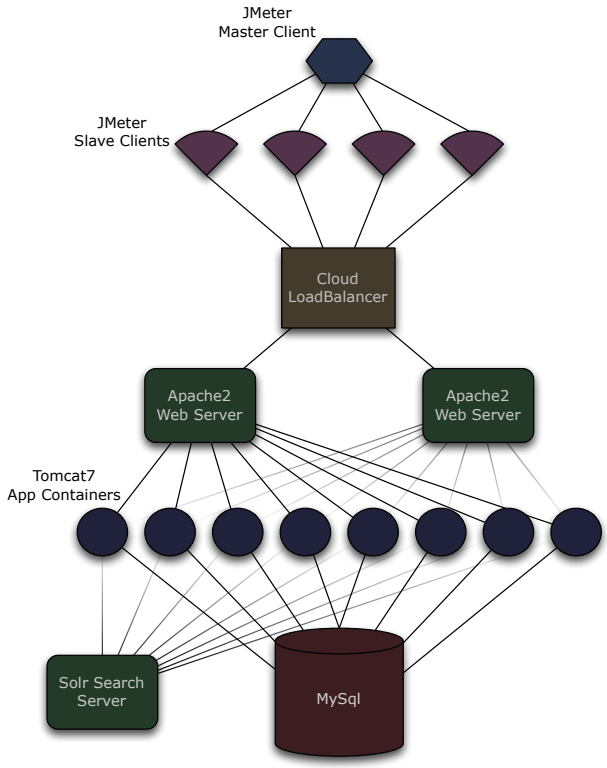
- Solr 4.0 (4 GB RAM, 2 Cores)

Database Server

- MySQL 5.5 (30 GB RAM, 8 Cores)

Client Test Machines

- JMeter 2.8 Master (2GB RAM, 2 Cores)
- JMeter 2.8 Slave (2GB RAM, 2 Cores)



Notes

1. Configured in the Rackspace Cloud Control Panel
2. SSL terminated at the load balancer
3. Rackspace offers a more scalable version of MySQL in the form of their Cloud Database product, though due to testing specifics, Broadleaf used a single MySQL installation of a standard cloud server

Appendix C – Database Tuning

Specific MySQL tuning changes were made for Broadleaf's load test

When using MySQL, Broadleaf Commerce depends on the use of the InnoDB table type. There are several configurations for MySQL and InnoDB that can be made to optimize performance at the database layer. MySQL configuration changes are enacted by editing MySQL's my.cnf file.

Variable	Value	Notes
thread_cache_size	Should be equal to or greater than the max_used_connections status variable	Amount of threads MySql keeps in cache to serve connections
max_connections	Should be high enough to handle peak load	Maximum number of connections MySql will allow
thread_concurrency	Should be between 2 to 4 times the number of cores	Test benchmark to determine the best value
table_cache	1024	Number of tables MySql will cache
innodb_buffer_pool_size	80% of available RAM	Data and index cache
innodb_thread_concurrency	At least 8, even on 1 core. Calculate as (2 * core count) + 2.	Regulate the count of threads working inside InnoDB
innodb_flush_method	O_DIRECT	Prevents double buffering of MySql cache by the OS
innodb_log_buffer_size	4M	Reduce log flush frequency
innodb_file_per_table	This is an option and has no value	Prevent single tablespace bloat
query_cache_size	64M	Cache common queries
query_cache_limit	2M	Limit the size of a query MySql will cache

Appendix D – Other Configuration Considerations

Specific configuration settings were used for Apache, Tomcat, and Broadleaf

Apache Web Server

Apache is very efficient at serving content and routing connections to your app container. However, it should still be configured to handle your target connection count. Test benchmarking will provide an indication of how many total connections and how many separate Apache web server installations you will need.

On a standard Apache2 installation on Ubuntu 12.0.4 LTS, Apache is installed in worker mpm mode. The snippet below in apache2.conf changes the max number of clients Apache will handle to 800.

```
...
<IfModule mpm_worker_module>
    StartServers          16
    MinSpareThreads       25
    MaxSpareThreads       75
    ThreadLimit           64
    ThreadsPerChild       25
    ServerLimit           32
    MaxClients           800
    MaxRequestsPerChild   0
</IfModule>
...
```

Tomcat Application Container

Tomcat will also need to be configured to handle a target connection count as follows:

1. Set the maxThreads attribute on the connector in use in server.xml to the max number of concurrent requests for the instance to be able to handle. 500 is a good value for a Tomcat on a 2-core server.
2. Set the max heap size for Tomcat to between 1.2 GB and 2.5 GB. Perm gen size to 256M. If using the asset server with a lot of image effects, or if using the embedded Solr server with a large catalog, more heap may be required.
3. A good value for the container connection pool max connection count is 100.
4. Use Tomcat 7's jdbc connection pool. It is faster and more reliable than other Java connection pools at the moment. See Appendix F for a sample connection pool configuration for Broadleaf Commerce.

Broadleaf Commerce Implementation

Each implementation will also need to be configured to take advantage of Hibernate level 2 cache and Thymeleaf cache (Optional – if using Thymeleaf for the presentation layer). Enable these beans in the implementation's application context xml file. See Appendix F for a sample bl-override-ehcache.xml configuration.

```
<bean id="blWebTemplateResolver"
class="org.thymeleaf.templateresolver.ServletContextTemplateResolver">
        <property name="prefix" value="/WEB-INF/templates/" />
        <property name="suffix" value=".html" />
        <property name="templateMode" value="HTML5" />
        <property name="cacheable" value="true"/>
        <property name="characterEncoding" value="UTF-8" />
 </bean>

    <bean id="blEmailTemplateResolver"
class="org.thymeleaf.templateresolver.ClassLoaderTemplateResolver">
        <property name="prefix" value="emailTemplates/" />
        <property name="suffix" value=".html" />
        <property name="templateMode" value="HTML5" />
        <property name="cacheable" value="true"/>
        <property name="characterEncoding" value="UTF-8" />
 </bean>

<bean id="blMergedCacheConfigLocations"
class="org.springframework.beans.factory.config.ListFactoryBean">
        <property name="sourceList">
            <list>
                <value>classpath:bl-override-ehcache.xml</value>
            </list>
        </property>
 </bean>
```

Appendix E – Sample Tomcat JDBC Configuration

Tomcat JDBC configuration was added to context.xml packaged with Broadleaf

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
    <Resource name="jdbc/web"
              auth="Container"
              type="javax.sql.DataSource"
              factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
              testWhileIdle="true"
              testOnBorrow="true"
              testOnReturn="false"
              validationQuery="SELECT 1"
              timeBetweenEvictionRunsMillis="30000"
              maxActive="100"
              maxIdle="10"
              minIdle="5"
              removeAbandonedTimeout="60"
              removeAbandoned="false"
              logAbandoned="true"
              minEvictableIdleTimeMillis="30000"
         jdbcInterceptors =
"org.apache.tomcat.jdbc.pool.interceptor.ConnectionState;org.apache.tomcat.jdbc.
pool.interceptor.StatementFinalizer"
              username="${database.user}"
              password="${database.password}"
              driverClassName="${database.driver}"
              url="${database.url}"/>
    <Resource name="jdbc/storage"
              auth="Container"
              type="javax.sql.DataSource"
              factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
              testWhileIdle="true"
              testOnBorrow="true"
              testOnReturn="false"
              validationQuery="SELECT 1"
```

```
          timeBetweenEvictionRunsMillis="30000"
              maxActive="15"
              maxIdle="10"
              minIdle="5"
              removeAbandonedTimeout="60"
              removeAbandoned="false"
              logAbandoned="true"
              minEvictableIdleTimeMillis="30000"
              jdbcInterceptors =
"org.apache.tomcat.jdbc.pool.interceptor.ConnectionState;org.apache.tomcat.jdbc.
pool.interceptor.StatementFinalizer"
              username="${database.user}"
              password="${database.password}"
              driverClassName="${database.driver}"
              url="${database.url}"/>
    <Resource name="jdbc/secure"
              auth="Container"
              type="javax.sql.DataSource"
              factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
              testWhileIdle="true"
              testOnBorrow="true"
              testOnReturn="false"
              validationQuery="SELECT 1"
              timeBetweenEvictionRunsMillis="30000"
              maxActive="15"
              maxIdle="10"
              minIdle="5"
              removeAbandonedTimeout="60"
              removeAbandoned="false"
              logAbandoned="true"
              minEvictableIdleTimeMillis="30000"
              jdbcInterceptors =
"org.apache.tomcat.jdbc.pool.interceptor.ConnectionState;org.apache.tomcat.jdbc.
pool.interceptor.StatementFinalizer"
              username="${database.user}"
              password="${database.password}"
              driverClassName="${database.driver}"
              url="${database.url}"/>
</Context>
```

Appendix F – Sample Ehcache Configuration

Adjusting the built-in Ehcache XML best fits individual catalog size and preferences

This configuration in bl-override-ehcache.xml that is packaged with the Broadleaf application will increase the size of the cache that holds catalog items. Adjust according to each catalog size and preferences. Note, this configuration also assumes that the cache is eternal and items are evicted from cache via another mechanism (e.g. Nightly job or JMS queue listener). Set eternal to false and add time-to-live values for regular cache timeout needs.

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <cache
        name="blStandardElements"
        maxElementsInMemory="1000000"
        eternal="false"
        timeToLiveSeconds="3600"
        overflowToDisk="true"
        statistics="true">
        <cacheEventListenerFactory
class="org.broadleafcommerce.common.cache.engine.HydratedCacheEventListenerFacto
ry"/>
    </cache>

</ehcache>
```