

# Fine-Tuning Performance

**jive**



---

# Contents

<b>Fine-Tuning Performance</b> .....	2
<b>Adjusting Application Caches</b> .....	2
How the Application Uses Caches to Avoid Database Requests.....	2
How to Fine Tune Caches.....	2
<b>Client-Side Resource Caching</b> .....	3
<b>Server-Side Page Caching</b> .....	3
<b>Performance Recommendations for High Traffic Sites</b> .....	4
<b>Adjusting Java Virtual Machine (JVM) Settings</b> .....	5
<b>Assorted Performance Tuning Tips</b> .....	5

# Fine-Tuning Performance

Through adjustments to caches, JVM settings, and more, you can make sure that the application is performing well.

It's almost certain that you'll want to adjust application settings from their defaults shortly after you're up and running. In particular, you'll want to keep an eye on caching, but there are other things you can do to ensure that the application is performing as well as possible. See the following for tuning suggestions.

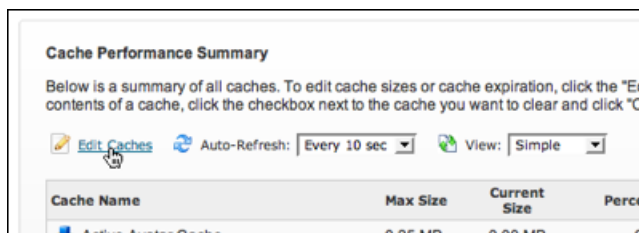
## Adjusting Application Caches

Jive SBS caches data in order to reduce the number of trips the application makes to its database. The caches improve performance by letting people get things done in the application more quickly. You can adjust the sizes of these caches so that they're best tailored for your community's needs. Adjusting cache settings often is the best way to improve performance.

You'll probably do most of your cache adjustment during the first few months of your deployment, then less frequently as use of the application grows.

## How the Application Uses Caches to Avoid Database Requests

- Each of nearly 100 kinds of data is cached in a separately adjustable cache. Each of these caches contains data that the application might need to retrieve and display for users.
- You set each cache's maximum size (in memory) based on how much data might need to be cached. If the maximum is reached during use, then the application can no longer store data in the cache and must make queries to the database to retrieve the data the user has requested (for example, by viewing a particular page).
- A cache's performance is measured in terms of **effectiveness**. Effectiveness is a percentage that represents how often the application successfully attempts to retrieve useful data from the cache. When the cache's maximum size is too small, the application must query the database instead.
- Default maximum sizes for the caches are usually not appropriate for a new community because circumstances vary so greatly from community to community.
- You can adjust cache maximums by clicking the **Edit Caches** link at the top of the **Caches** page.



## How to Fine Tune Caches

- Start by choosing a cache preset that makes sense for your use. When you're editing caches, you can choose one of the presets included with Jive SBS: small, medium, and large (there's also a custom option). Each of these sets all caches to particular default levels. A good rule of thumb is to start with a larger preset, such as medium or large, then adjust downward if it turns out you don't need that much.
- Watch for caches with poor effectiveness and adjust those caches by increasing their maximum size. The application signals especially poor effectiveness with a red box, as shown in the following illustration. The columns, left to right, are maximum size, current size, percent used, and effectiveness.

0.23 MB	0.00 MB	0.2%	41.0%
2.00 MB	0.23 MB	11.4%	97.4%
4.00 MB	3.83 MB	95.7%	59.1%
Unlimited	0.00 MB	0.0%	100.0%
2.00 MB	0.07 MB	3.7%	99.5%

- Use the Auto-Refresh interval on the cache performance summary page to have the page refresh regularly with current numbers.
- For the first few weeks in production keep a close watch for low cache effectiveness (note the red boxes!). When effectiveness gets too low for a particular cache, increase its maximum size. Here's a suggested schedule:
  - Watch effectiveness hourly immediately after launch.
  - After caches become stable, watch daily.
  - On an ongoing basis, watch weekly.
- In general, try to keep effectiveness over 90 percent.
- Look for caches you can set to very low levels, such as those for features you're not using. For example, if you're not using blogs, you might set blog-related caches to 0.5MB. (Don't set it to 0, though, which effectively sets it to "unlimited.") This frees up overall memory you can use for other cache maximums.
- Ensure that the total of cache maximums doesn't exceed one-third of the maximum memory you've assigned to the JVM. The total is shown at the bottom of the cache summary list, while your total Java memory is shown at the very bottom of the page where the Java memory monitor is displayed. See below for more information on adjusting the JVM settings.

**In the UI:** Admin Console: System > Settings > Caches

## Client-Side Resource Caching

The platform HTTPD server is pre-configured for optimal caching of static production content. Default configuration values for content caching can be found on a Jive-managed server at `/usr/local/jive/etc/httpd/conf.d/cache.conf`. You can edit this file to change default cache time or headers for specific scenarios (changing length of time static images are cached, for example). Changes to this file will be preserved across upgrades to a file named "cache.conf.rpmnew". If this file is changed, be sure to check for new enhancements when upgrading.

**Note:** Certain resources in plugins and themes are cached for 28 days by default. These include the following file types: `.js`, `.css`, `.gif`, `.jpe`, `.jpg`, and `.png`. This means that clients won't see updated versions of those files until their cache expires or is cleared. Of course, changing the resource's file name will also cause it to be downloaded because it isn't yet cached.

## Server-Side Page Caching

You can adjust server-side page caching for anonymous users when their having the very freshest content is less of a concern. With server-side caching on, the server caches pages that are assembled dynamically from data and resources. Retrieving a page from the cache can save the time needed to assemble a fresh page. However, if the data that makes up the page has changed, the page in the cache won't be as fresh as a new one would be.

With server-side page caching disabled (and for registered users, whether or not caching is enabled), Jive SBS sends its default HTTP headers. With page caching enabled, in addition to the server-side page cache stored in memory, the application also sets the HTTP header in the response to `Cache-Control max-age=3600`.

The value set for `max-age` is configurable as described below.

You can set these with system properties in the admin console.

Property	Description	Values
jive.pageCache.enabled	Enables server-side page caching.	false (default) to disable page caching; true to enable it. When enabled, only anonymous or guest users will receive cached content.
jive.pageCache.maxage.seconds	Sets the age after which the server will create a fresh page rather retrieve the page from the cache.	Defaults to 60 seconds. This sets Cache-Control: max-age=60 in the HTTP headers for the page.
jive.pageCache.expiration.seconds	Sets the number of seconds after which a page will be removed from the cache.	Defaults to 30 seconds
jive.pageCache.maxEntries	Sets the maximum number of pages that can be maintained in the cache. Note that increasing this value might require that you provide more system resources for the application.	Defaults to 1000 entries.

Note that turning on developer mode by setting the `jive.devMode` property to true will disable the `maxAgeFilter` setting (effectively setting `jive.maxAgeFilter.enable` to false). The `jive.devMode` property is intended for situations when you're developing themes or plugins, In those situations, caching can hinder you from seeing the results of your development work.

**In the UI:** Admin Console: System > Management > System Properties

## Performance Recommendations for High Traffic Sites

Here are a few recommendations for high-traffic deployments in which most users are anonymous. Anonymous users tend to visit merely to read content rather than contribute. Because they're anonymous, by default there's very little about their experience that will be customized. As a result, you can typically afford to serve them cached content rather than freshly rendered dynamic content.

- Recognize anonymous users from the cookies the application sets. These cookies are available in 1.x versions as of 1.10.3 and in version 2.0.1, but are unavailable in version 2.0. If you're using a version in which they're not available, you should consider upgrading.
  - The application sets a `jive.user.loggedIn` cookie whose value is true if the user is registered and logged in.
  - Use the `jive.server.info` cookie to preserve consistency between user requests. This cookie contains information about the server from which the user receive content. Its value might be:

```
serverName=myhost:serverPort=8080:contextPath=/our-
community:localName=myhost:localPort=8080:localAddr=127.0.0.1
```

- Offload page caching to dedicated cached-content servers. With large numbers of anonymous users, you should consider using a content delivery network (CDN) such as Akamai or Limelight to serve cached content to anonymous users.
- Avoid sending a max age page cache header back to the client if the anonymous user will be able to log in. If the browser caches the page, then the newly logged in user might not receive fresh dynamic content.
- Consider caching resources such as CSS, Javascript, and decorator image files, which tend to rarely change. In fact, consider caching these resources for days if you expect not to be deploying new versions of them.
- Use the Query Stats feature to track database queries. The Query Stats page of the admin console can display SQL statements executed, how often they're executed, and how long the query takes.

The number of queries you see when browsing your front pages should be low. For more information about this feature, see *Examining Database Queries*.

- When you have a large percentage of anonymous users but the overall number of page views is smaller, use the application's built-in server-side page caching. Starting with version 2.0.0, the application can [cache rendered pages for anonymous users](#) (page 3) . You can enable the cache and set maximum age and expiration thresholds.

## Adjusting Java Virtual Machine (JVM) Settings

As with any Java-based web application, you can sometimes improve performance by assigning particular values to Java Virtual Machine options. The default location for environment configuration is located at /usr/local/jive/applications/sbs/bin/instance. Within this file, it is possible to edit JVM maximum memory settings by adding a value for JVM\_HEAP\_MAX. This value is expressed in MB, so for example, to set the max heap available to the application to 4GB, add the following lines to the instance file:

```
JVM_HEAP_MAX=4096
```

```
JVM_HEAP_MIN=4096
```

Jive recommends setting the MIN and MAX values the same to ensure that the system is capable of allocating the desired maximum value and that an OutOfMemoryException will not occur during runtime.

Note that in addition to the heap designation above, a typical Jive SBS instance will consume 512M of JVM "PermGen" heap. In the above example, increasing the MIN and MAX values to 4GB would result in a JVM process consuming slightly more than 4.5GB RAM.

## Assorted Performance Tuning Tips

Here are a few ways to get the application running the most efficiently.

- Consider setting the default for threaded discussions to "flat." People will still be able to set thread mode their own views to "threaded," but setting the default will ensure the "flat" mode for new users.
- If you're using a load balancer, make sure it's configured for session affinity/sticky sessions.
- On Oracle 11g, use the prepared statement cache to reduce database overhead.
- The OCI driver must be used as opposed to the "thin" JDBC Type4 driver. Using the OCI driver will require installation of Oracle native binaries compatible with the operating system hosting the Jive SBS installation.