Intergral Information Solutions GmbH

Schickardstr 32 • D-71034 • Böblingen • Germany

Quick Start to Troubleshooting Server Problems with FusionReactor

Configuration Suggestions for Immediate Troubleshooting and Production Monitoring

Doc. Rev. 82, 12 February 2008

Feedback

We welcome feedback on all our products and publications. Please e-mail them to support@fusion-reactor.com and we will address them as quickly as possible.

Published in Germany

Contents

Introduction

Introduction

This document helps you configure FusionReactor. There are many different scenarios in which FusionReactor is used, each with its own conditions, and each requiring a different configuration. What we aim to do in this document is cover sample configuration for short term (immediate) troubleshooting and longer-term production monitoring.

It's not possible to come up with a universally-perfect configuration, because every scenario is different. What we've tried to do in this document is cover the most important points to consider when configuring FusionReactor for different scenarios.

Getting FusionReactor

We are assuming you've installed FusionReactor already; if not, go ahead and install it now. The installation process is easy – in almost every case the installer package will take care of all the details. We've prepared a comprehensive **Installation Guide** which covers all the details:

http://www.fusion-reactor.com/fr/downloads.cfm

Configuration And Diagnosis In Emergencies

This section helps you configure FusionReactor to diagnose a crashing server.

Initial Configuration

The emphases for configuration of a system in trouble should be on getting immediate data with a reasonable history.   Once the system can be recovered sufficiently to keep it running for longer periods, you can proceed to "***Configuration and Diagnosis for Production***" on page 10 for further tuning.

Logging

The **Resource** and **Request** logs are important to diagnose server problems, since they record the memory/CPU usage of the system and the running requests respectively. They are potentially very useful in an unstable environment, where restarts will cause FusionReactor to lose its in-memory data.

All logging within FusionReactor is a computationally inexpensive operation, the limiting factor being available disk space.

If you have disk space available, you can afford to be generous with the log file size. FusionReactor is delivered with a configured value of **10MB** for each log file, with a log history of **5** files. These files are rotated, so the maximum log set for each log category will be **50MB**. This may seem like a lot, but it will be invaluable in the event of an issue.

FusionReactor's log files are located in `/FusionReactor/conf/INSTANCE_NAME/log`. The newest log file is always named `something-0.log`, the second newest `something-1.log` and so on. When FusionReactor needs to create a new log file, it renumbers the existing files, deleting the oldest, and creating a new one at the start of the list. When an issue occurs, you should zip up these log files. You may need them later to analyze what happened.

To set the **request** log size, select **Requestes -> Request Settings -> Request Logging -> Enabled**; set **File Size** to as much disk space as you can afford, bearing in mind the system will potentially create *n* files of this size (specified in **File Count**). To set the **resource** log, select **Resources -> Resource Settings -> Resource Logging.**

You can get more information on how to use the logs in "***FusionReactor's Logs Explained***" on page 16.

In-Memory Metrics

FusionReactor stores a number of tables and lists in memory, in order to produce useful metrics and graphs. These metrics will be your first point of observation to address issues on an ailing server.

Metrics

The **Metrics** section will be vitally important in diagnosing immediate problems with the system. With metrics enabled, you will be able to immediately visualize the state of the system. **System Metrics** provides six graphs to help you:

Request Activity measures the number of requests completing over the last second (in 1-minute range mode) or the number of requests completing over the last minute (in 1-hour range mode) on the **blue** chart. The **gray** chart displays the number of requests which were active (but may not have yet completed) over the same time period.

JDBC Request Activity similarly measures JDBC request completions and activity.

Memory Usage charts the JVM demand for memory over the two periods. The **gray** data series denotes the maximum memory – the maximum memory that the operating system may make available to Java. The **white** series shows the memory Java has currently demanded from the operating system. The **blue** series shows exactly how much memory Java has actually used for objects.

Average Request Time shows, on the **blue** chart, the time elapsed, on average, for all requests completing in the last sample period (see **Request Activity**). The **gray** chart shows the average time including requests which are currently running.

Average JDBC Time similarly measures the average time of completed and currently-executing JDBC requests.

CPU Usage measures, on the **gray** chart, the quantity of CPU time consumed overall by all processes on the system. The **blue** chart measures the quantity of CPU time consumed by the J2EE (ColdFusion) server.

These graphs may be run in ranges of 1 minute (for immediate troubleshooting) or 1 hour (for trend spotting). The graph ranges can be switched individually or together at any time to either mode: no data is lost.

Immediate Diagnosis

Log Save

When a J2EE (ColdFusion) crash occurs, **before restarting, create a log set**. If, after restarting, FusionReactor rotates the log files, data may be lost. The files in the FusionReactor log folder should be zipped up, or at least copied and saved. Your J2EE log files (`typically ColdFusion/logs/*`) and container log files (`/ColdFusion8/runtime/logs`) should also be saved.

System Metrics -> Memory

Memory should be your first observation. If the **blue** chart of the **Memory Usage** graph is **consistently** near the top of the graph, consider making more memory available to the J2EE (ColdFusion) process. This can be altered within the **ColdFusion Administrator** in the section **Server Settings -> Java and JVM -> Maximum JVM Heap Size**. If there is insufficient memory on the system to increase the JVM memory, consider reducing the size of the **Template Cache** and **Cached Queries**.

A **sawtooth pattern** in the blue memory graph is normal. This shows Java periodically garbage collecting objects. When **used** memory begins to approach the **allocated** memory value, you may see one or more sawtooth (garbage collection) patterns, as Java attempts to reclaim memory before asking for more. If insufficient memory is reclaimed, you will see the white **allocated** memory bound increase, as Java demands memory from the operating system. If the blue portion steadily rises over the course of an hour, this often indicates a **memory leak**. These are becoming more common as the complexity of J2EE applications increases.

If the used memory (blue) graph is growing rapidly, you can try to ask Java to perform garbage collection yourself by clicking on the **trash can** icon on the Memory Usage graph. However, Java's memory algorithm is very sophisticated and it's unlikely a manual collection will have any signigicant effect.

System Metrics -> CPU Usage

CPU Usage is another useful metric. If the instance is consistently busy (see the blue graph on the **Memory Usage** chart) with low load (see **Request Activity** graph), this might indicate a problem with the pages being run (see **Request History**), such as infinite loops or runaway queries (consider using the **FusionReactor JDBC Driver Wrapper** to analyse and prevent JDBC problems).

The **Metrics -> Longest Requests** page shows the longest running requests. FusionReactor also flags long-running requests with an appropriate label in the **Request History** table. You can configure what FusionReactor considers a long-running request, and how many to store in the Longest Requests table, in the **Metrics Settings** page.

CPU/Memory Resource Graphs

In addition to the System Metric graphs, FusionReactor also provides more resolute **Resource Graphs.** These graphs use a moving window, and can be zoomed in to a high level of detail. Once you have identified a time period in which an issue occurred, the Resource Graphs can be zoomed in to that point to get a detailed picture of the system during the incident.

Two graphs can be independently controlled: the **CPU Graph** and the **Memory** Graph. The resolution and range of the graphs is controlled by the **Resource Settings** page. By default, the data for the Resource Graphs is collected once every 5000ms (5s), and a total history of 8640 samples is stored. This gives a total range of 12 hours of data with 5s resolution.

This is typically enough for most production monitoring tasks; however if you do require more resolution to diagnose a specific problem, the sample interval can be lowered to as little as 100ms which will provide very resolute data.

The data used in the resource graph is an **in-memory structure**, and although a single sample takes up very little memory, please do bear in mind that increasing history and/or resolution will have an impact on available memory.

Configuration And Diagnosis For Production

This section takes the longer view on production monitoring and troubleshooting. We'll talk about how to identify trends, and how to pick intelligent settings for Crash Protection and monitoring.

Production Systems

When we say **Production** we mean a stable system which is serving live requests. These systems aren't used to test new applications, and they usually settle into a steady state over time. Once they stabilize, we can perform some simple analysis to tune FusionReactor.

The principal of tuning a production system is to watch for anomalies in the performance of the system, in essence defining exactly what constitutes an "exception case". Once we've identified this, Crash Protection rules can be defined to report when this case occurs.

There are essentially two methods of defining these cases: gut feeling and numerical analysis. We'll give an example of both, and how they apply to each of the Crash Protection settings.

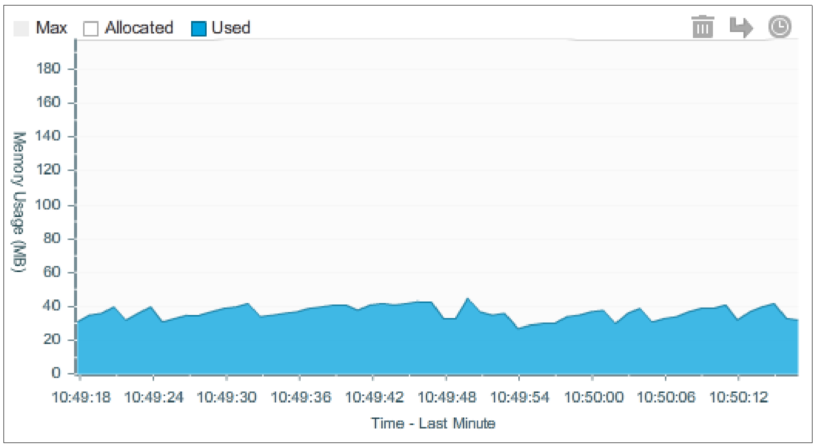
Gut Feeling

The "gut feeling" principal is best supported by the **System Metrics** graphs. With the system running steadily for at least an hour, the graphs – when switched to **Hour** mode - clearly show how the system is being used.

Memory Protection

As an example, we set up a load test to simulate a fairly constant load on the system.

You can see in the System Metrics page that the memory displayed in the **Memory Usage** graph is reasonably constant. If we observe the system for an hour, we would get an even better picture of how the memory is used. For even longer periods, we can set the **Resource Settings -> Resource Sampling** interval to 60000ms (1 minute), with a long history size. We can then view this data in the **Resources -> Memory Graph** page. Using a longer time span allows us to get a picture of time-dependent fluctuations such as peak load and scheduled tasks.



Using the data on the Memory Usage graph, we can see that the steady state for our test is around 40MB. Allowing a 100MB overhead for scheduled tasks and peak-time usage, this would give an acceptable operational peak of 140MB, with a remaining "emergency" margin of 60MB.

We might then set Crash Protection **Memory Protection** to **30%** (60MB margin = 30% of the total memory).

The system should then be left in **Notify** mode for a period of time. While running in this mode, adjustments can be made to the protection value based on the notification email. When you are satisfied that the value you have represents the upper margin for the system, change the notify mode to **Abort and Notify** or **Queue and Notify**.

Timeout Protection and Request Protection

The **System Metrics** charts may also be used to generate values for **Timeout Protection** (using the **Average Request Time** graph), and **Request Protection** (using the **Request Activity** graph).

Restrictions

In the detailed example above, we've allowed a margin in the operational memory estimation to account for periodic rises in website usage, and scheduled tasks. Scheduled tasks often run at periods of low demand, and are typically resource-intensive. Search indexing, reporting and cleanup operations often run in this batch mode, and often consume memory and CPU cycles. These tasks also tend to run for long periods of time.

Once you have a picture of these tasks, you can add them as **CP Restrictions** (Crash Protection -> CP Restrictions). Ensure that **CP Settings -> CP Restrictions -> Restrictions** is **Enabled**, and the mode is **Ignore Matching Requests**. CP will then ignore any requested pages set in the **CP Restrictions** page.

Once you have added your scheduled tasks as restrictions, the operational values for **Timeout**, **Memory** and **Request Protection** can be adjusted, since they will no longer apply to the exceptional cases.

Numerical Page Analysis

The second method of setting Crash Protection values uses the log files generated by Crash Protection to generate a measure of the statistical spread of page runtimes. Using this measure, we can compute a statistical upper bound to the runtime, which can serve as a starting point for further tuning.

Computing Standard Deviation of Page Runtime

Computing the statistical standard deviation of page runtimes gives you a weighted measure of the spread of the values. We can then use this measure to set the **Timeout Protection**.

The simplest method of computing this value is to take advantage of FusionReactor's **Request Log**.

You can get more information on how to use the logs in "***FusionReactor's Logs Explained***" on page 16.

Methodology in OpenOffice

1. Ensure **Requests -> Request Settings -> Request Logging** is **Enabled.**
2. Allow the system to run in its steady state for a while (the longer the better).
3. Load the **request log** (`/FusionReactor/instances/NAME/log/request-0.log`) into OpenOffice Calc
    1. Open the file as a **Text CSV** file, with options: **Separated by: Space**, **Text delimiter: "**
4. Create a new sheet (**Insert -> Sheet**), naming the new sheet **Sheet2.**
5. Switch back to **Sheet1**, where the data is, and select **Edit -> Select All**.
6. Select **Data -> Filter -> Standard Filter**, setting **Column F**[1] equal to **COMPLETED**. This will remove any other request statuses from the list. Click the **More** button, and select **Copy Results To**. Enter **`$Sheet2.$A$1`** in the destination box and click **OK.**
7. When the copy is complete, you should have only the **COMPLETED** entries in Sheet 2.
8. In a free cell which is not in column L[2], enter the formula:
   `=AVERAGE(L1:L65535)+STDEV(L1:L65535)`
    1. This value is the upper limit at **one standard deviation away from the average.**

1 You can find a complete list of the meanings of each log column in **Help**.

2 Column L is the runtime of the page, measured in ms.

9. Statistically,this may still produce too many false positives for Crash Protection, so we may want to use 2, 3 or even 4 standard deviations:

    1. **`=AVERAGE(L1:L65535)+4*STDEV(L1:L65535)`**

As with the gut feeling approach, this number can only ever be a starting point for tuning: you are bound to get some exceptions, and these should help you decide how to adjust this value.

Configuration For Enterprise And High-Availability Environments

Introduction

FusionReactor Enterprise Edition makes monitoring more than one J2EE (ColdFusion) instance easy with the **Enterprise Dashboard**. Any FusionReactor Entperise Edition installation is able to report the state of the J2EE server remotely to any other FusionReactor Enterprise Edition sever.

We are often asked about the best configuration for enterprise environments: this section aims to provide some guidance about how best to set this up.

Setting Up A Separate Monitoring Instance

By default, FusionReactor Enterprise Edition is configured to monitor itself. You'll see the local instance in the **Enterprise Dashboard**. We don't recommend running any scripts or using the **Server Shutdown/StartUp Alerts** functionality in this mode, since we can't guarantee the behaviour of a system monitoring itself.

In Enterprise or High-Availability Environments, we recommend installing FusionReactor as a separate monitoring solution. This will involve using a J2EE container containing **only** FusionReactor, whose sole purpose is to monitor other instances. This solution insulates the master monitoring FusionReactor from problems occuring in other J2EE (ColdFusion) instances. Because FusionReactor is licensed on a per-machine basis, no extra licenses are required in this scenario.

We recommend installing a new Tomcat[3] instance for this solution. Simply add the monitored machines to the **Manage Servers** page and use the **Enterprise Dashboard** to monitor the status. If more detailed information is required, click on the small **arrow** icons on the dials or the server cube: you'll be logged into the remote server automatically and directed to the appropriate detail page.

3 http://tomcat.apache.org/

FusionReacor in High-Availability Environments (HAE)

In a High-Availability Environment (HAE), we recommend a redundant Dashboard configuration: FusionReactor is installed on two (or more) physical servers. Each monitor is configured to monitor all the cluster instances (or a subset of them) **and the other monitor**. Using **Server Shutdown/StartUp Alerts**, the system administrator can monitor the status of all the affected servers. For maximum protection, each FusionReactor monitor should be installed in its own Tomcat server.

If you configure FusionReactor in this way, you should be careful how you handle **Enterprise Scripting** if you are using it to perform restart functions.

1. Never use scripting to restart the monitoring instance. Because of the order that FusionReactor's subsystems start and stop, a normal restart may cause a script to fire.
2. Always offline servers in the Dashboard before performing scheduled maintenance. Otherwise, Enterprise Scripting may try to restart them.
3. Do read the document **FusionReactor Enterprise Scripting**, which is available on our website[4] and is installed along with FusionReactor.

Enterprise Dashboard: Servers and Groups

Another recommendation is for people with large numbers of servers. The **Enterprise -> Manage Groups** page allows you to group together monitored instances into logical groups. You can use this functionality to partition the Dashboard into logical areas that fit your organisation. There's no limit to the number of servers that can go into a group.

If you need an instance to be present in more than one group, simply add it again using **Manage Servers**, then put it in the second group. Provided the **URL** used to access the monitored instance is the same, FusionReactor will only poll the system once, and make that data available in both Dashboard icons.

Once your servers have been organised into groups, you can then simply monitor the **group cube** – it will always reflect the **worst status** of the group.

4 http://www.fusion-reactor.com/fr/helpdocs/FusionReactor%20Enterprise%20Scripting.pdf

FusionReactor's Logs Explained

FusionReactor provides many helpful pages and metrics to help you see how your system is performing, but what happens when your server begins to crash?

As we mentioned previously, lots of this data is stored in-memory so it can be used to generate reports and lists for the FusionReactor Admiinstrator. If you are having problems which cause (or necessitate) restarts of the server process, this in-memory data is lost.

In these cases, you may want to make use of FusionReactor's extensive logs. This data can in almost all cases be used in a post-mortem session to pinpoint the location of problems within a system.

In addition to immediate post-mortem work, the logs can also be used to analyse usage patterns, trends and to derive longer-term aggregate data. One example of this data is given on page 12, where we used the **request log** with standard deviation to derive a nominal page runtime value.

Available Logs

You can find FusionReactor's logs in `/FusionReactor/conf/INSTANCE_NAME/log`. They are rotating files, the newest having the suffix **-0.log**, the next oldest **-1.log** etc.

Here's an overview of the available log files, with examples of the data contained within them:

request-0.log	Contains log messages about the requests which ran inside FusionReactor and how they were processed. If requests were queued, aborted or rejected, an approriate message will be written to this log. Normal execution of requests is also logged here, with time, page name, memory details etc.
resource-0.log	Contains period information on memory usage, CPU load, request load and JDBC activity.
jdbc-0.log	Contains information reported from wrapped

	datasources: query text, rows retrieved, time spent waiting for the database etc.
crashprotection-0.log	Contains log messages written by Crash Protection about what action it has taken on a request, along with the state of the system at the time the CP fired.
reactor-0.log	This is a free-form text log containing diagnostic messages from FusionReactor.

Log Contents

Describing each log file in detail is beyond the scope of this guide. However, you will find a **TYPE-headers.txt** file in the log folder, one for each log. This file contains the symbols used in the logs.

Additionaly, FusionReactor's online help system (accessible by opening the **FusionReactor** section in the FusionReactor Administrator, then selecting **Help**) contains extensive information about each log column. The help is also provided on our website:

http://www.fusion-reactor.com/fr/help/help.htm

Exactly **what** is logged is configurable in the FusionReactor Administrator, in the corresponding configuratino section. For instance, to configure request logging, navigate to **Requests -> Request Settings -> Request Logging**.

Format

The logs are formatted as **space-delimited, string quoted** files. This means that each field in the log file is separated by a space. Where a string whicih may contain spaces appears in the log file, it is supplied in quotes.

For example, using this methodology, the following log record:

REQUEST 1 2 “Hello, World!” 3

... contains the following discrete fields:

REQUEST

1

2

Hello, World

3

Most log analysis software, database import and spreadsheet tools are able to import this data correctly.

Getting Help

Where To Get Help

Fusion Products Knowledge Base

We provide a knowledge base full of technical articles and helpful hints at the following URL:

http://www.fusion-reactor.com/fr/support.cfm

From here you can search the support database, see what's new, what our hot issues are, as well as finding online versions of our printed manuals.

Google Group

We also provide a Google Group for the community to ask questions in.

http://groups.google.com/group/fusionreactor

We check and reply to this group regularly, so it's also worth searching to see if your question has already been answered.

Support Mailbox

.. and last but not least, you can mail us FusionReactor questions at:

support@fusion-reactor.com

We can also provide software consulting services if you're having problems with your application:

http://www.fusion-reactor.com/support/supportservices.cfm