SREA Goes Through The Roof! UP 106.6%

Score One Inc. (SREA)
$0.301 UP 106.6%

Investors are all over SREA as frenzy buying pushes shares prices over 106% following recent news releases. Read up, watch for more news, and get on SREA first thing Friday!

Another reason for the low score is that there are several branches which are us ually taken in the reference workload, but usually untaken in the training workl oad.
so and pass it a Java class.
class files into a new SE.

The term general race condition throughout this article means a race condition w
hich is not data race type.
The term general race condition throughout this article means a race condition w
hich is not data race type.
Java callstacks, as reconciled, are shown in the Java Representation, and in the
 Expert Java Representations if they were captured for the event.
Less frequent but harder to find are general race conditions.
loaders will select the NetBeans data loader classes.
But for support questions, see the support page.
sh script in the WebLogic installation, and use it as a template for collectlaun
ch.
Java Callstacks The Sun Studio Performance tools collect their data by recording
 events in the life of the each LWP of the process, along with the callstack at
the time of the event.
A Java program is inherently multithreaded, and has one JVM thread for each thre
ad in the user’s Java program.
You can set all its methods to one color, all the methods from com.
You can also use the -d argument to specify a directory into which the experimen
ts should be stored.
If there is another thread operating on the objects in parallel, it may miss the
 target objects because they are homeless at the critical time.
degree from Columbia College and a Ph.
The data in the experiment will cover the interval between the two signals, whic
h should be the steady-state behavior.
There are some situations where race conditions in a parallel program for perfor
mance reason.
These scripts are relatively straightforward shell scripts, generated from the W
ebLogic Configuration Wizard.
The data race condition problem is often harder to debug than the problem caused
 by a corrupted pointer.
Finally, in the last section, we give links to the BEA WebLogic documentation.
Data race condition problems are common, while rarer, but harder, general race p
roblems can also occur.
, analyzing and optimizing the performance of applications on current and future
 UltraSPARC systems.
The relevant part of the file is shown below, with added code is indicated red.
One of these is the actual interpreter code used to process the applications Jav
a byte-code methods.
A corrupted pointer often points to an illegal region and just causes a segmenta
tion fault.
Each Java thread’s event callstack is shown with its Java methods.
The data in the experiment will show startup in the first sample, steady-state i
n the second, and shutdown in the third.
, analyzing and optimizing the performance of applications on current and future
 UltraSPARC systems.
jar file Ensuring the new SE.
In parallel programming, developers need to take a new perspective and design th
eir code as multiple flows of concurrent operations.
Benign Data Race Condition Not every data race condition is a harmful problem.
A workaround to support profiling for such an application, is to not set -j on f
or the initial collect invocation, and to ensure that the -Xruncollector option
is passed to the invoked JVM.
loaders will select the NetBeans data loader classes.
When the target has completed its startup, send the signal to it, and then let t
he target run to completion, or terminate it.
When the JVM starts, it creates a number of regions of dynamically-generated cod
e in its data space.
windows to another, all the methods in org.
It starts and stops specific servers.
It should be a locally-mounted, as opposed to NFS-mounted, file system.

At any point in the execution of any application, Java or otherwise, the callsta ck represents where the program is in its execution, and how it got there.
Another reason for the low score is that there are several branches which are us ually taken in the reference workload, but usually untaken in the training workl oad.
To use the sample mechanism, use -l instead of -y, and proceed as above, sending the signal to the target at the end of startup.
Both machine- and Java callstacks are collected for these events, but no synchro nization tracing data is collected for internal locks used within the JVM.
All methods whose name matches the algorithm chosen will be colored with the sel ected color.
This kind of checking and update pattern is quite common in parallel programming , but it causes a data race condition in nature.
It is important for programmers to be aware of these land mines before they step into the dangerous parallel programming zone.
Dynamically compiled methods are loaded into the data space of the application, and may be unloaded later.
Profiling Startup Time of an Application Either mechanism can be used to look a t the startup time of an application.
Each Java thread’s event callstack is shown with its Java methods.
All threads, including the JVM system threads are shown in the Timeline.
If every object’s attribute will match only one group attribute in this partitio ning example, this data race condition is benign and will not cause any harmful result.
It starts and stops specific servers.
Dynamically compiled methods are loaded into the data space of the application, and may be unloaded later.
In general the general race is subtle to avoid in the first place and also hard to fix even when you are lucky to discover the problem.
For example, you may want to look at the startup only, or you may want to look a t the data for the individual benchmark loads that were run.
The filtering can be changed at any time, and data is there for the entire run.
Initially, set all functions to some arbitrary color, say gray.
jar file and the collector.
Java callstacks, as reconciled, are shown in the Java Representation, and in the Expert Java Representations if they were captured for the event.
In this article, a simple and popular parallel partitioning program example is u sed to illustrate the above various race condition issues.
To use the sample mechanism, use -l instead of -y, and proceed as above, sending the signal to the target at the end of startup.
The second situation is where the branch behaviour of the application is unpredi ctable.
Less frequent but harder to find are general race conditions.
You may want to use the date command to generate a string representing the curre nt date and time and embed that string in the experiment name.
In addition, other code is generated in the data space to execute the transition s between interpreted and compiled code.
It starts and stops specific servers.
When it is stopped by the Node Manager, the experiment is terminated.
However if an object can belong to multiple groups, this data race condition is a real problem and will cause a harmful result.
Those mechanisms make it easier to understand the performance of part of the run of a program.
Initiating Data Collection This section describes how you can modify the scripts used to launch WebLogic servers to enable data collection.
In the end, it verifies that the total count is identical to the input object co unt to see if any objects were not collected in the partitioning.
When it is stopped by the Node Manager, the experiment is terminated.
For programs with a very long startup that is not interesting, pause and resume is usually preferable.

Java programs may have explicit synchronization, usually performed by calling th e monitor-enter routine.
When you are ready to terminate the run, send the signal again, which will disab le data recording.
Both scripts will generate the appropriate graphs if they are able to locate gnu plot on the path.
To use the sample mechanism, use -l instead of -y, and proceed as above, sending the signal to the target at the end of startup.
Collecting the Data Profiling a Java application is done by collecting data on t he JVM as it runs the Java application.
The big hazard of general race condition demonstrates an important design lesson for parallel programming.
Both callstacks are recorded during profiling, and are reconciled during analysi s.
Descendant Process Controls Many WebLogic Servers do not spawn additional proces ses, so -F on is not needed.
You may change the script to extract the directory from that path, and use it to put experiments in the same place the Node Manager logs are put.
One of these is the actual interpreter code used to process the applications Jav a byte-code methods.
In the partitioning example shown below, the main program processes the input so urce data into an object array and then creates N working threads to sort and co llect the objects in parallel.
In addition, the Timeline coloring techniques described above were used to color the callstacks by the classes to which each function belongs.
Then, set properties for the Node Manager to tell it to use the new script, and restart the Node Manager.
But why check for javac, as opposed to java?
Synchronization Tracing Synchronization tracing for Java programs is based on events generated when a thread attempts to acquire a Java Monitor.
Excellent Good Fair Poor Comments: If you would like a reply to your com ment, please submit your email address: Note: We may not respond to all submitte d comments.
Read Darryl’s blog at blogs.
A PC for a Java method in the Java representation corresponds to the method-id a nd a byte-code index into that method; a PC for a native function correspond to a machine PC.
But for support questions, see the support page.
It is hard to see that there is a data race problem here by just looking at the code.
Other java methods may be dynamically compiled using HotSpot, and are referred t o as compiled methods.
Both scripts will generate the appropriate graphs if they are able to locate gnu plot on the path.
The experiment begins when the server is launched, and terminates when the serve r exits.
You might find that your problem has already been solved.
All threads, including the JVM system threads are shown in the Timeline.
All methods whose name matches the algorithm chosen will be colored with the sel ected color.
This data race problem is quite subtle and hard to understand without a serious investigation.
In the machine representation, memory is allocated and deallocated by the JVM, t ypically in very large chunks.
It also will show the pseudo-functions from JVM overhead threads.
The overhead of these routines, and the cost of writing the experiments to disk will dilate the runtime of the Java program.
sh; the two properties to be set are: Property Value StartTemplate collectlau nch.
To profile a server, you must ensure that the JVM command launching the server i

s prepended with a collect command, with appropriate arguments, to invoke the Su
n Studio Collector.
The data race condition problem is often harder to debug than the problem caused
 by a corrupted pointer.
The effect of such a data race problem is unpredictable and may occur only once
during hundreds of runs.
The overhead of these routines, and the cost of writing the experiments to disk
will dilate the runtime of the Java program.
To make things worse, there are many subtle aspects surrounding race conditions.

It is important for programmers to be aware of these land mines before they step
 into the dangerous parallel programming zone.
To profile a server, you must ensure that the JVM command launching the server i
s prepended with a collect command, with appropriate arguments, to invoke the Su
n Studio Collector.
Another reason for the low score is that there are several branches which are us
ually taken in the reference workload, but usually untaken in the training workl
oad.
In this case, we choose to name the script collectlaunch.
It is important for programmers to be aware of these land mines before they step
 into the dangerous parallel programming zone.
jarfile, along with the other unmodified classes, maintaining its package struct
ure.
The data race condition problem is often harder to debug than the problem caused
 by a corrupted pointer.
Download Sun Studio Compilers and Tools today.
Profiling NetBeans The "NetBeans Performance Page" contains quite a few HOWTOs a
bout what the NetBeans team did in this area recently.
In this case, we choose to name the script collectlaunch.
It is hard to see that there is a data race problem here by just looking at the
code.
Data collection is initially paused.
java A menu selection is another important event, but it may be triggered just b
y moving the mouse, not pressing the mouse button or a key.
One of these is the actual interpreter code used to process the applications Jav
a byte-code methods.
Race conditions are hard to avoid and also hard to find with the conventional de
bugging methods and tools.
A general race condition is a much harder problem to detect than a data race con
dition.
Some data race conditions are not harmful and can be permitted in parallel softw
are for performance reasons.
In addition, other code is generated in the data space to execute the transition
s between interpreted and compiled code.
A data race condition is caused by unordered concurrent accesses of the same mem
ory location from multiple threads.
That generated machine code is also in the data space of the process.
Initially, set all functions to some arbitrary color, say gray.
This kind of checking and update pattern is quite common in parallel programming
, but it causes a data race condition in nature.
Another reason for the low score is that there are several branches which are us
ually taken in the reference workload, but usually untaken in the training workl
oad.
The JVM does a number of things that are typically not done by applications writ
ten in traditional languages.
so and pass it a Java class.
The big hazard of general race condition demonstrates an important design lesson
 for parallel programming.
sh; the two properties to be set are: Property Value StartTemplate collectlau
nch.

In the above partitioning example there is another data race that occurs in the collect routine.
It is hard to see that there is a data race problem here by just looking at the code.
It always helps to implement some auxiliary methods to verify the internal progr am states during each operation stage.
For programs which don’t run more than ten or fifteen minutes total, sample-mark ing is usually preferable.
The steps to be taken are: Modifying the classes Compiling the changed classes P acking the new .
However if an object can belong to multiple groups, this data race condition is a real problem and will cause a harmful result.
However this fix may not be a complete solution to meet some application partiti oning requirements.
For any particular Java method, there will be an interpreted version, and there may also be one or more compiled versions.
The code is modified to call the collector API to record a sample for each of th ose events.
The experiment begins when the server is launched, and terminates when the serve r exits.
Where Do I Go To Get Help?
To use the sample mechanism, use -l instead of -y, and proceed as above, sending the signal to the target at the end of startup.
If sample markers are used, the data volume will be higher, but you can filter b y sample to look at any part of a run, even parts you thought would be uninteres ting when the data was recorded.
Initiating Data Collection This section describes how you can modify the scripts used to launch WebLogic servers to enable data collection.
If the target sets a signal-handler, the collector will put a warning message in the experiment, but the data collection controls will not work properly.
java file from which it was compiled, with metrics on each source line.
The script also automatically creates a shell-script file named kill.
The upper-right and lower-left quadrants indicate that the branch behaviour is s imilar in the reference and training workloads.
An example of the Timeline from such an experiment is: In this example, the samp le numbers, and the legend superimposed on the timeline were added to the image by hand.
The set can be all those methods whose name starts with a given string, or ends with a given string, or contains a given string, or which matches a regular expr ession.
Let’s say after the first phase of collection, the partitioning program needs to fine-tune and shuffle some objects from a group container to another group cont ainer as shown in the code below.
In the Java representation they are all shown aggregated as a single method.
The stop command will terminate the profiling run.