Mentor Graphics Changes to the Eclipse Tracing and Monitoring Framework (TMF)

Document Version: 1.0
Creation Date: September 17, 2010

Embedded Systems Division Austria

Table Of Contents

Mentor Graphics Changes to the Eclipse Tracing and Monitoring Framework (TMF)

Overview

The following document lists our major changes to the TMF framework. A good understanding of the behavior and concept of the original TMF framework is recommended.

The base version of these changes is the public repository http://dev.eclipse.org/svnroot/technology/org.eclipse.linuxtools/lttng/trunk revision **24106**.

Event skipping

Event skipping means to advance a ***TmfTraceContext*** – a pointer into an event trace – over any desired numbers of events at the ***TmfTrace*** level. Thus a custom implementation derived from the ***TmfTrace*** class can support more efficient implementations that just repeatedly single stepping events.

ITmfTrace / TmfTrace

The method ***skipEvents*** is added to advance a ***TmfTraceContext*** over a number of events at once. This is similar to repeatedly call ***getNextEvent*** for a given number of events or until a specific stop condition is reached, but has the following advantages:

- For fixed size events the skip operation is simple and can be implemented to be very fast.
- An additional stop condition can be used to move the ***TmfTraceContext*** to a required time stamp or to the first event that matches a given condition.
- The custom implementation derived from the ***TmfTrace*** class can prevent creation of ***TmfEvent*** objects that are skipped and thus improve performance.

The ***TmfTrace#seekEvent*** methods are changed to use the new skip feature to improve their performance.

Indexing

TmfTrace

The indexing operation of ***TmfTrace*** objects is changed to use the new skip feature. As such, indexing performance is vastly improved, and indexing of fixed size event streams is now very fast.

Some methods to control indexing are added. Beside the existing method ***indexStream*** the new methods ***reindexStream***, ***isIndexingStream*** and ***stopIndexingStream*** are added.

TmfExperiment

The indexing operation of ***TmfExperiment*** objects is completely rewritten to improve performance. A heuristic is used to employ the new skip feature as much as possible, even when multiple traces are in the experiment.

An ***EventIterator*** class is added that provides significantly better performance than the ***getNextEvent*** method. When the ***getNextEvent*** method had an effort of o(n) where n is the number of traces in the experiment, the new class has an effort of o(1) – in the case when multiple traces in the experiment do not overlap in time – to o(log n) – in the case when multiple traces in the experiment overlap in time.

The ***getNextEvent***, ***positionTraces*** and ***processDataRequest*** methods are changed to use the new ***EventIterator*** class.

Searching and filtering

The searching and filtering extension to TMF is based on a set of methods and classes that complement the unfiltered access methods available in original TMF. The actual operation of testing if an event matches the filter criteria or not is implemented in the custom implementation derived from the ***TmfTrace*** class, thus no custom or event specific behavior is part of the TMF framework itself. Filter parameter are passed as a verbatim Object to the underlying TmfTrace implementation.

TmfTrace

The newly introduced ***TmfTrace#skipEvents*** method supports a filter parameter, and will stop skipping events when an event matches the filter. Because custom implementation derived from the ***TmfTrace*** class can implement filtering at a low level, ***TmfEvent*** objects that do not match are not even created for performance reasons.

Another new method ***TmfTrace#testEvent*** tests if ***TmfEvent*** objects match a given filter criteria. It must be implemented by custom implementations derived from the ***TmfTrace*** class. The default implementation matches every event.

It is possible to share the work between ***TmfTrace#skipEvents*** and ***TmfTrace#testEvent*** by ***skipEvents*** performing pre-filtering, thus reducing the amount of event objects that are created, and ***testEvent*** will perform the full featured filtering.

TmfExperiment / TmfDataRequest

The ***TmfDataRequest*** class is complemented with a new ***TmfFilteringDataRequest*** class that includes the additional filter criteria object. It is used with a new ***TmfExperiment#processRequest*** method that takes a ***TmfFilteringDataRequest*** object as parameter. The events returned by this request all match the filter criteria object as tested by the ***TmfTrace#testEvent*** method as described above.

A new experiment class is derived from ***TmfExperiment***, ***TmfFilteredExperiment***, that accepts a filter criteria object in the constructor. It provides all standard ***TmfExperiment*** interfaces, but returns only event data that match the filter criteria.

In other words, the ***TmfFilteringDataRequest*** is typically used in find / search scenarios, whereas the ***TmfFilteredExperiment*** is typically used where existing components like views shall be fed with filtered event data.

New public interfaces

TmfTimeRange TmfTimeRange#getIntersection(TmfTimeRange other)

Get intersection of two time ranges

boolean TmfDataRequest<V>#waitForCompletion(long timeout)

To suspend the client thread until the request completes (or is canceled).

long ITmfTrace#skipEvents(TmfTraceContext context, long number, TmfTimestamp time, Object filter, TmfTimestamp[] skipped)

Skips the given number of events or until the given time stamp is reached, updates the context to the next event, and returns the number of events that have been skipped.

If filter is not null, processing stops when the filter condition is fulfilled.

FindResult ITmfTrace#findNextEvent(TmfTraceContext context, Object filter)

Find the next event that matches the filter condition. Returns the event and the index of that event in this trace file.

The context is incremented after the result match - but not necessarily to the next match!

boolean ITmfTrace#testEvent(TmfEvent event, Object filter)

Test if the event matches the filter condition.

boolean TmfExperiment#getIndexing()

Indicates that an indexing job is already running

void TmfExperiment#processRequest(TmfFilteringDataRequest<TmfEvent> request, boolean waitForCompletion)

Process a data request for filtered data.

long TmfTrace#skipEvents(TmfTraceContext context, long number, TmfTimestamp time, Object filter, TmfTimestamp[] skippedTS)

See ITmfTrace#skipEvents

void TmfTrace#reindexStream()

Restart indexing the stream from the beginning. When another indexing operation is currently in progress, it is aborted first.

boolean TmfTrace#isIndexingStream()

Returns if an indexing operation is currently in progress.

void TmfTrace#stopIndexingStream()

Stop an indexing operation when one is currently in progress.

FindResult TmfTrace#findNextEvent(TmfTraceContext context, Object filter)

See ITmfTrace#findNextEvent

boolean TmfTrace#testEvent(TmfEvent event, Object filter)

See ITmfTrace#testEvent

void TmfTraceContext#set(TmfTraceContext other)

Set trace context from other context.

New public classes

class TmfFilteringDataRequest<V> extends TmfDataRequest<V>

Data request class that supports filtered results.

class TmfFilteredExperiment extends TmfExperiment

A TmfExperiment that supports filtered results.