Server Compiler Inlining Messages

This is a short description of the inlining messages from the server compiler. The policy is generally very greedy and several of the limits are intended to keep the compiler from attempting to inline the whole program.

In the descriptions below a call site is considered hot if the ratio of number of times this call site was executed to the number of times the callee was called greater than or equal to InlineFrequencyRatio or the number of times the call site was executed is greater than InlineFrequencyCount. These descriptions are derived from the inlining policy logic in src/share/vm/opto/bytecodeInfo.cpp and may change as the code evolves.

- too big: the call site isn't hot according to profiling but the bytecodes are larger than MaxInlineSize.
- hot method too big: the call site is hot according to profiling but the bytecodes of the callee are larger than FreqInlineSize.
- already compiled into a big method: there is already compiled code for the method that is called from the call site and the code that was generated for is larger than InlineSmallCode
- already compiled into a medium method: there is already compiled code for the method that is called from the call site and the code that was generated for is larger than InlineSmallCode / 4
- call site not reached: the profiling for the call site records no calls calls so it appears to have never been executed.
- exception method: methods that inherit from Throwable are only inlined in Throwable related methods
- never executed: the method being called has never incremented it's invocation counter so it has never been called.
- executed < MinInliningThreshold times: the invocation counter for the method reports less than MinInliningThreshold invocations.
- inlining too deep: the call depth is greater than MaxInlineLevel
- recursively inlining too deep: the call depth is greater than MaxRecursiveInlineLevel so inlining is stopped at this point
- native method: native Java method.
- size > DesiredMethodLimit: the inlining that's been done so far has inlined more than DesiredMethodLimit bytecodes so inlining will be stopped.
- NodeCountInliningCutoff: the inlining that's been done so far has created more than NodeCountInliningCutoff IR nodes so inlining will be stopped.
- unloaded signature classes: a class mentioned in the signature of the call site hasn't been loaded and can't be resolved. Since the compiler wants precise types is make it impossible for the method to be inlined since the types can't be resolved.

Common targets for tuning are to adjust the FreqInlineSize, MaxInlineSize and InlineSmallCode. Raising some of the other limits like DesiredMethodLimit or NodeCountInliningCutoff might work well for particular programs but they require a bit more care since they may create compiles with more than MaxNodeLimit nodes in the IR and will bail out of the compile. This means we won't produce compiled code at all for the root method of the compile. Look for compilation bail out messages that mention "out of nodes" in their description.