Ruby master - Feature #16663

Add block or filtered forms of Kernel#caller to allow early bail-out

02/28/2020 10:27 PM - headius (Charles Nutter)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

There are many libraries that use caller or caller_locations to gather stack information for logging or instrumentation. These methods generate an array of informational stack frames based on the current call stack.

Both methods accept parameters for level (skip some number of Ruby frames) and length (only return this many frames). However many use cases are unable to provide one or both of these.

Instrumentation uses, for example, may need to skip an unknown number of frames at the top of the trace, such as to dig out of rspec plumbing or active_record internals and report the first line of user code. In such cases, the typical pattern is to simply request *all* frames and then filter out the one that is desired.

This leads to a great deal of wasted work gathering those frames and constructing objects to carry them to the user. On optimizing runtimes like JRuby and TruffleRuby, it can have a tremendous impact on performance, since each frame has a much higher cost than on CRuby.

I propose that we need a new form of caller that takes a block for processing each element.

```
def find_matching_frame(regex)
  caller do |frame|
    return frame if frame.file =~ regex
  end
end
```

An alternative API would be to allow passing a query object as a keyword argument, avoiding the block dispatch by performing the match internally:

```
def find_matching_frame(regex)
  caller(file: regex)
end
```

This API would provide a middle ground between explicitly specifying a maximum number of stack frames and asking for all frames. Most common, hot-path uses of caller could be replaced by these forms, reducing overhead on all Ruby implementations and drastically reducing it where stack traces are expensive.