

Trend-Based Multi-Level Adaptive Tracing

Mohammed Adib Khan
ak19qp@brocku.ca

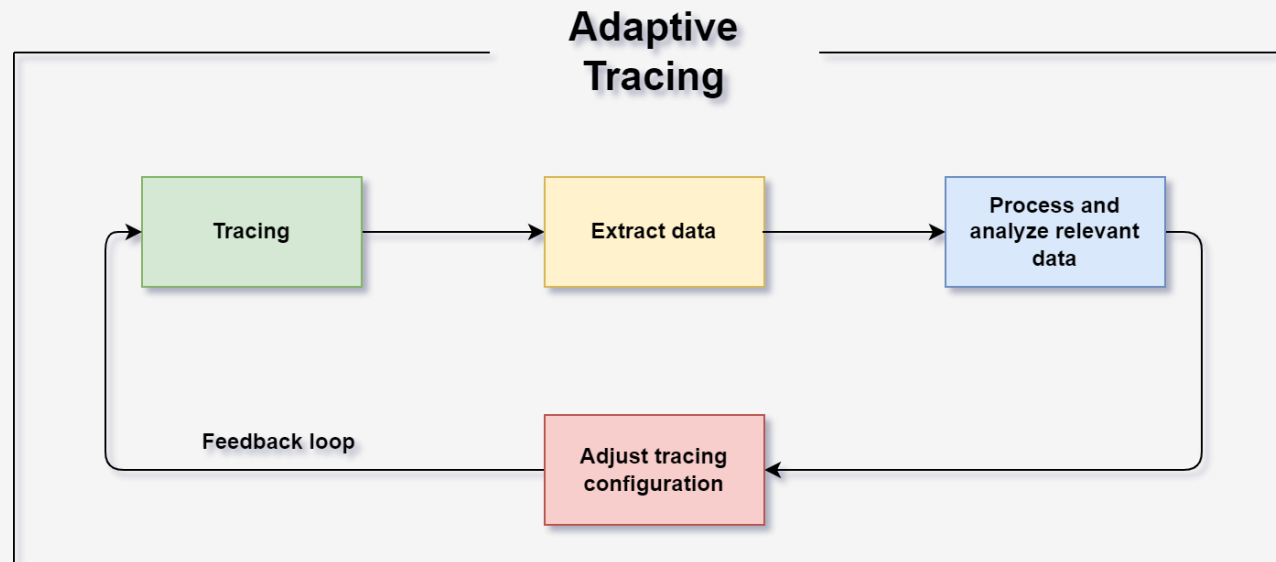
Supervisor: Dr. Naser Ezzati-Jivan

Department Of Computer Science

Brock University, Canada

Adaptive Tracing

Adaptive tracing – tracing instruments are controlled by a framework to figure out points of interest and keep on adjusting them automatically.



Ideas & Research Questions

❑ Different types of approach:

- Anomaly based
- Clustering / Grouping
- Pairwise similarity
- Trend based

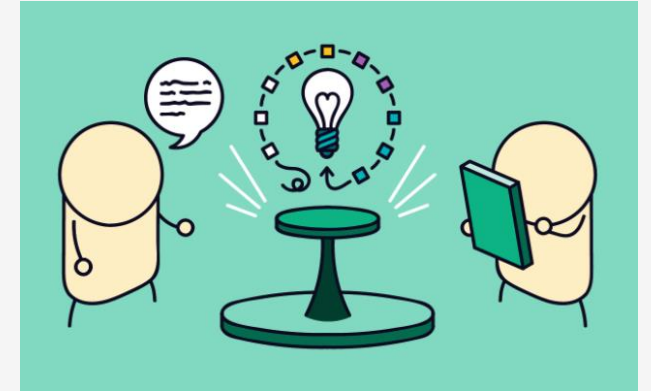
❑ We are interested in change detection of a trend.

- We assume a change in trend of method execution time or system call execution time may become a culprit for performance problems at present or in the future.

❑ Problem we are trying to address:- performance related issues in production.

❑ Research questions:

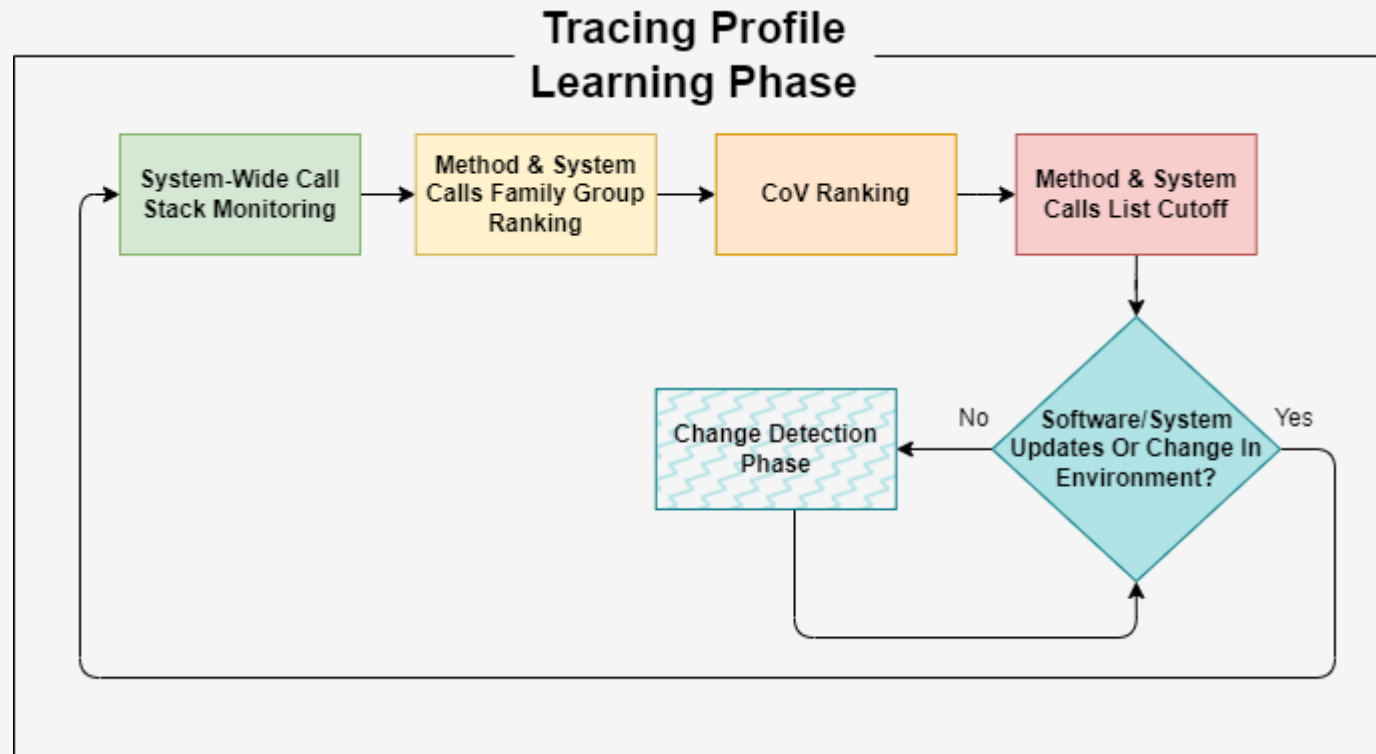
- How can we learn the tracing profile of each application?
- How can we detect behavior changes on runtime?
 - Should we adjust tracing config whenever there is a major change?
 - What are the possible actions in regard to the detected changes?
- How do we decide to change the tracing config?
- How do we evaluate the trace adjustments?



Proposed Method

- ❑ Multi-level (user level and kernel level) trend-based adaptive tracing.
- ❑ The method has two phases:
 - Learning phase-
 - ❖ Build application tracing profile.
 - Change detection phase-
 - ❖ Detect change in trend of methods and syscalls execution time.
 - ❖ Adjust tracing configurations based on the detected changes.

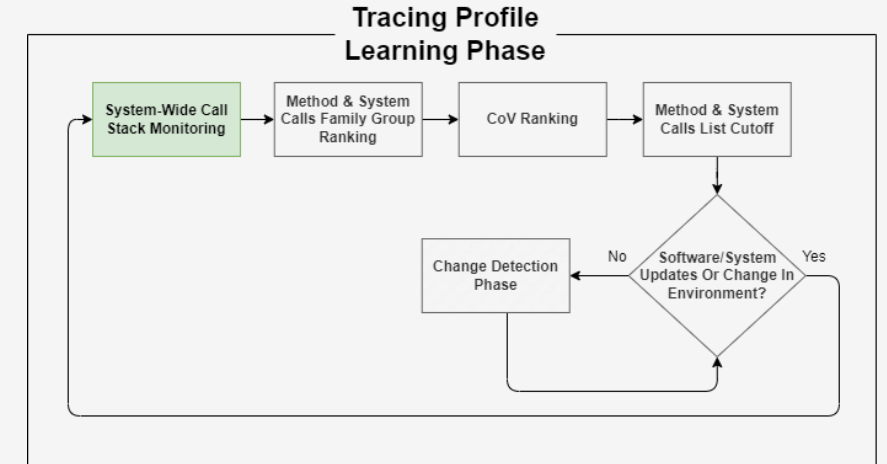
TBML Adaptive Tracing Method – Tracing Profile Learning Phase



TBML Adaptive Tracing Method – Tracing Profile Learning Phase

❑ Preliminary Monitoring:

- Application and system-wide monitoring of call stacks.
- Set a reasonable snapshot period (e.g 10ms for a browser)
- Stop after sufficient number of tasks have been performed by the system and recorded.
- We run the applications several times under different loads and stress saturations for the different scenarios of the software.

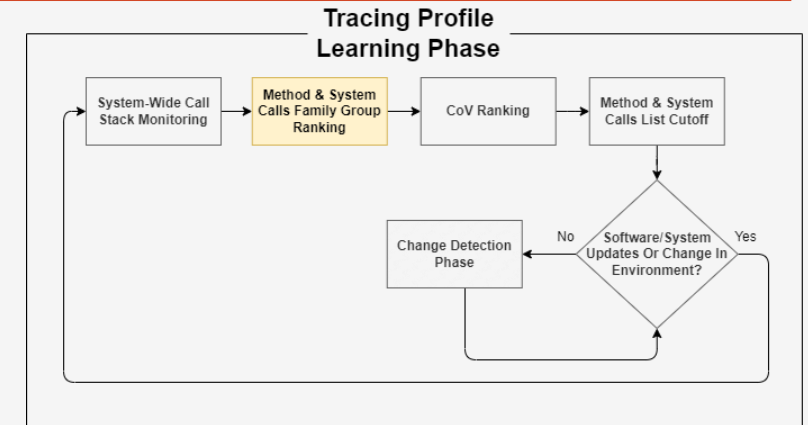
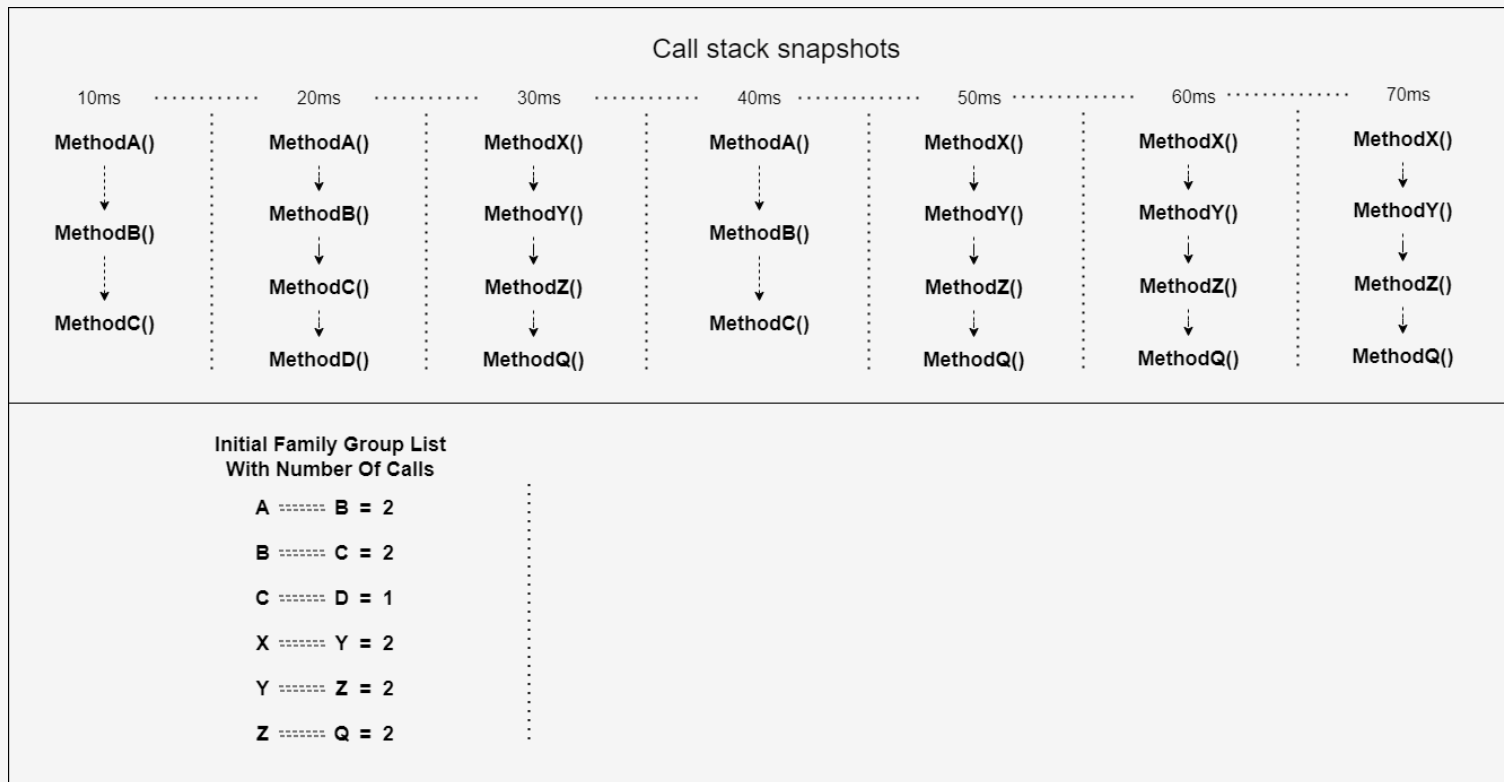


Application Run #1			Application Run #2		Application Run #n	
30 ms			... ms		...ms	
10 ms	10 ms	10 ms	10 ms	10 ms	10 ms	10 ms
Call Stack#1	Call Stack#2	Call Stack#...	Call Stack#1	Call Stack#...	Call Stack#...	Call Stack#...
Method_x	Method_f	...	Method_x
Method_y	Method_c	...	Method_y
...
...	Syscall_x
...
Method_z	Method_z	...	Method_z
Syscall_x	Syscall_y	...	Syscall_x

TBML Adaptive Tracing Method – Tracing Profile Learning Phase

Family group identification:

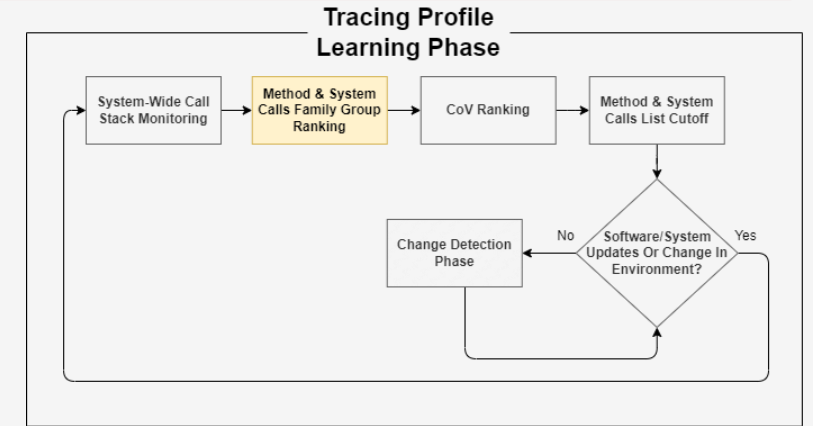
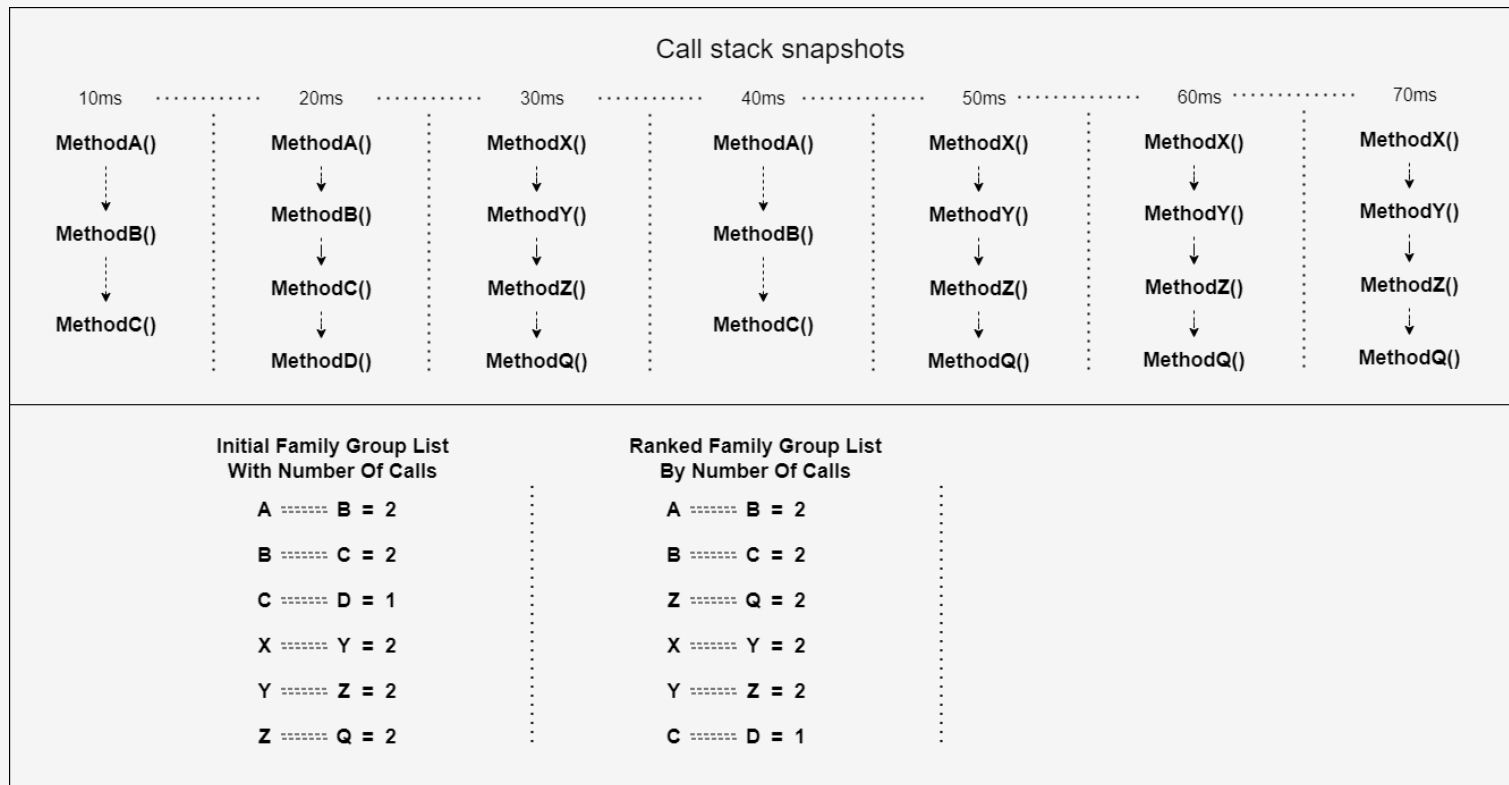
- Breakdown method calls into pairs of parent and child.



TBML Adaptive Tracing Method – Tracing Profile Learning Phase

Family group identification:

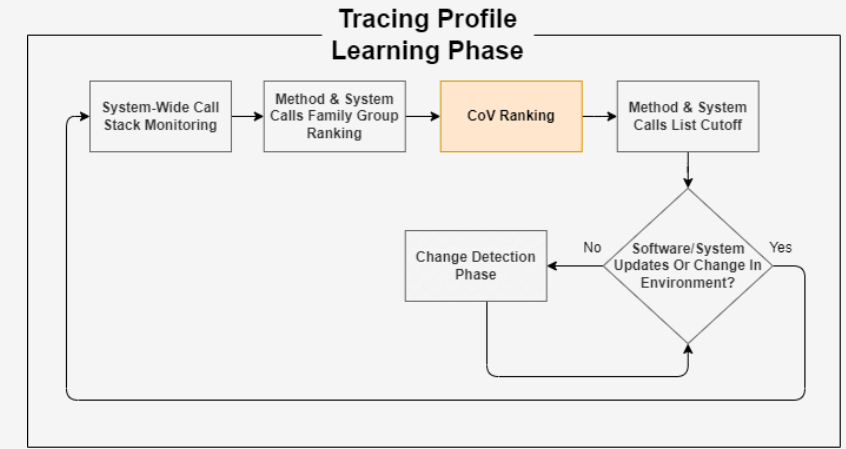
- Breakdown method calls into pairs of parent and child.
- Rank list in descending order of number of calls between parent and child.



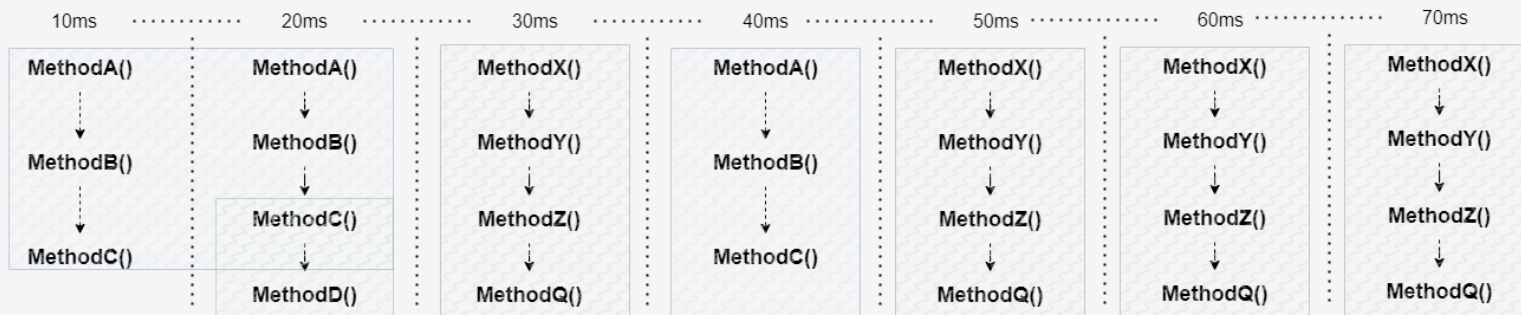
TBML Adaptive Tracing Method – Tracing Profile Learning Phase

Family group identification:

- Breakdown method calls into pairs of parent and child.
- Rank list in descending order of number of calls between parent and child.
- Find execution time of method calls.



Call stack snapshots



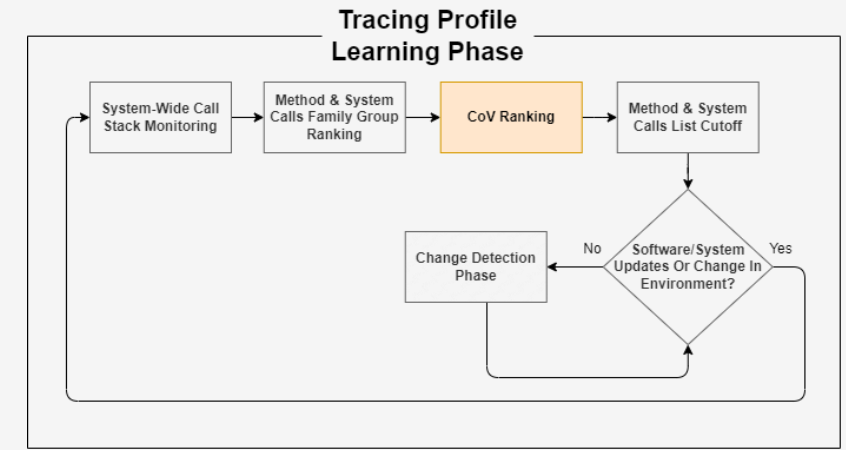
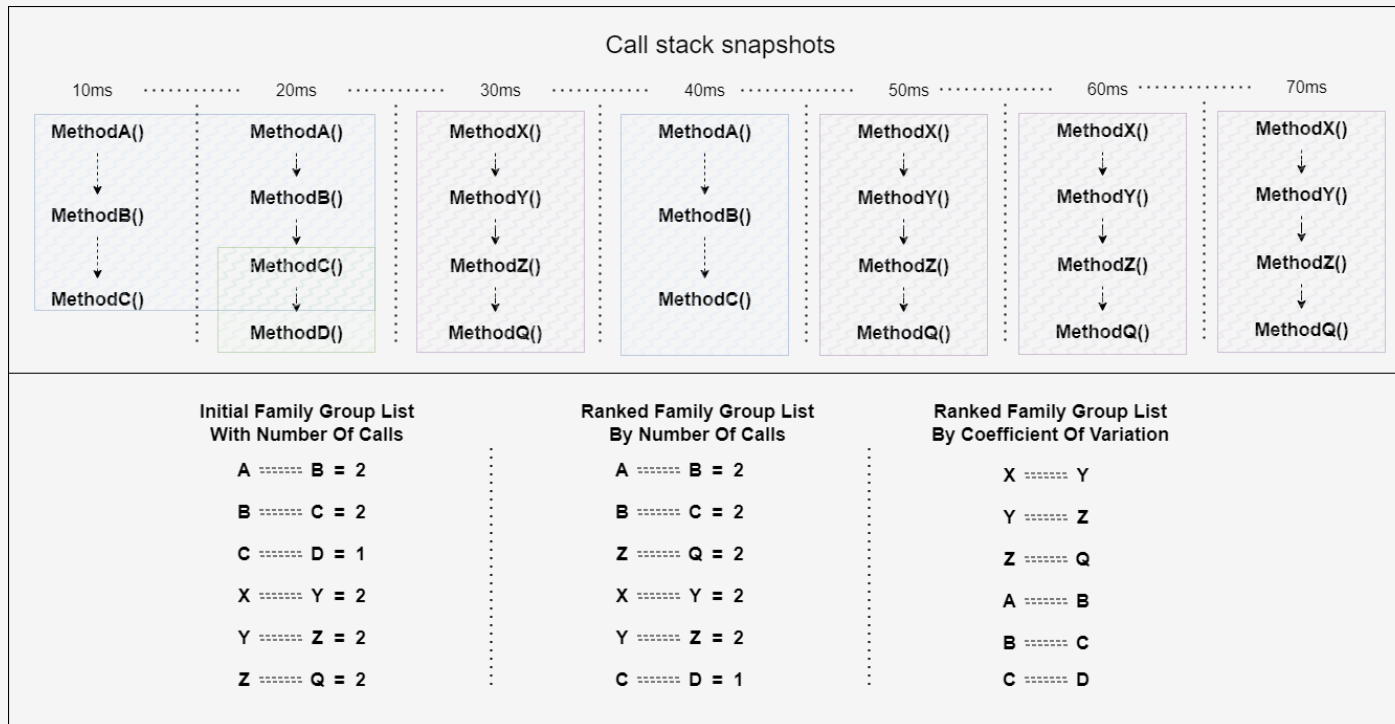
Time series data of method execution time

A = [20ms, 10ms, ...]	C = [10ms, 10ms, ..]	Y = [10ms, 40ms, ...]	Q = [10ms, 40ms, ...]
B = [20ms, 10ms, ...]	X = [10ms, 40ms, ...]	Z = [10ms, 40ms, ...]	

TBML Adaptive Tracing Method – Tracing Profile Learning Phase

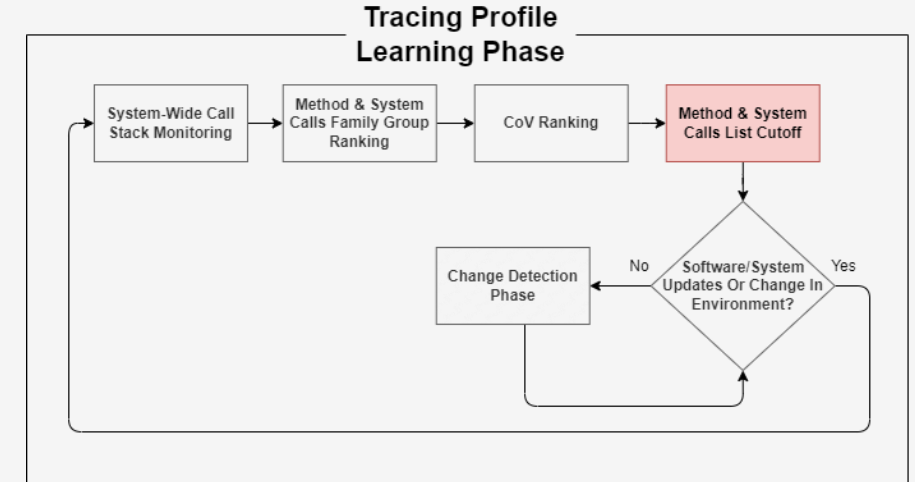
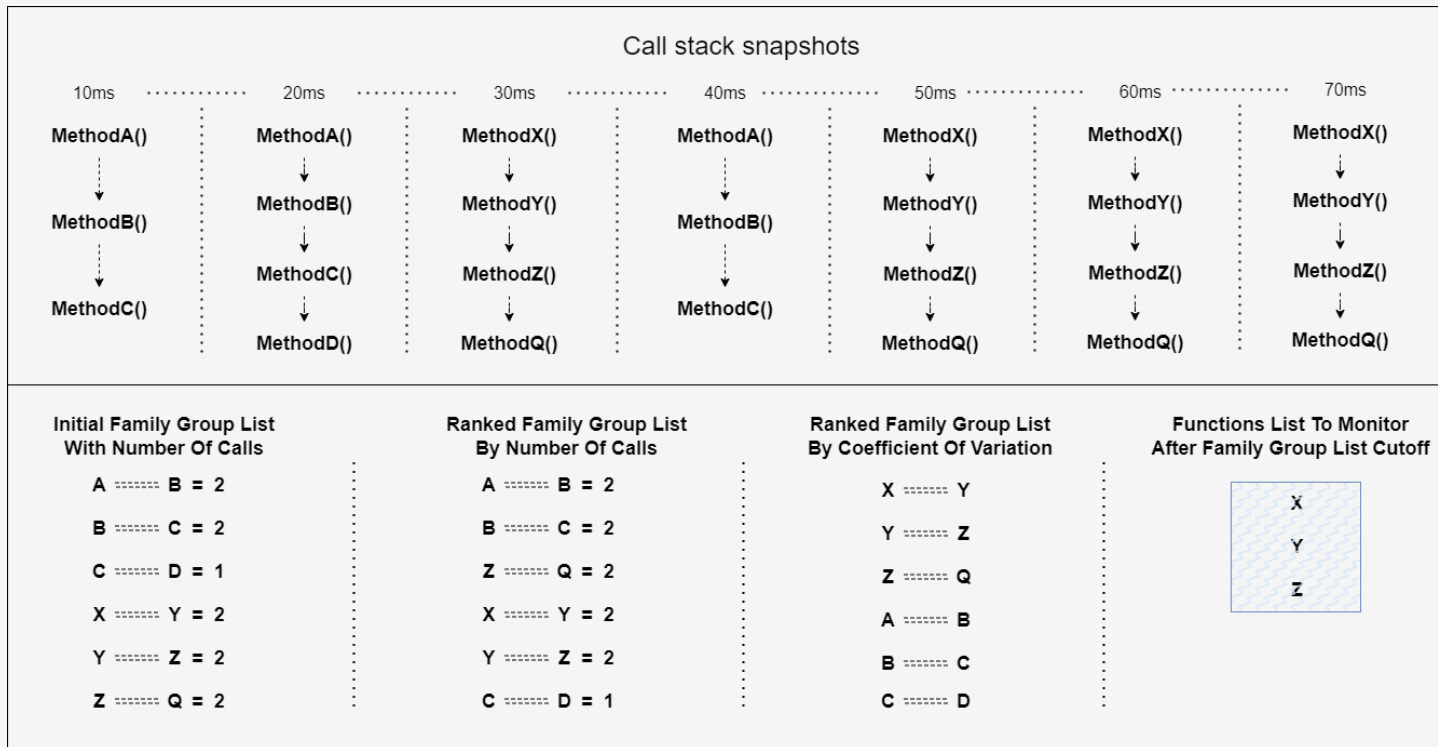
Family group identification:

- Breakdown method calls into pairs of parent and child.
- Rank list in descending order of number of calls between parent and child.
- Find execution time of method calls.
- Rank list in descending order of coefficient of variation of their execution time (i.e., fluctuation of their duration).



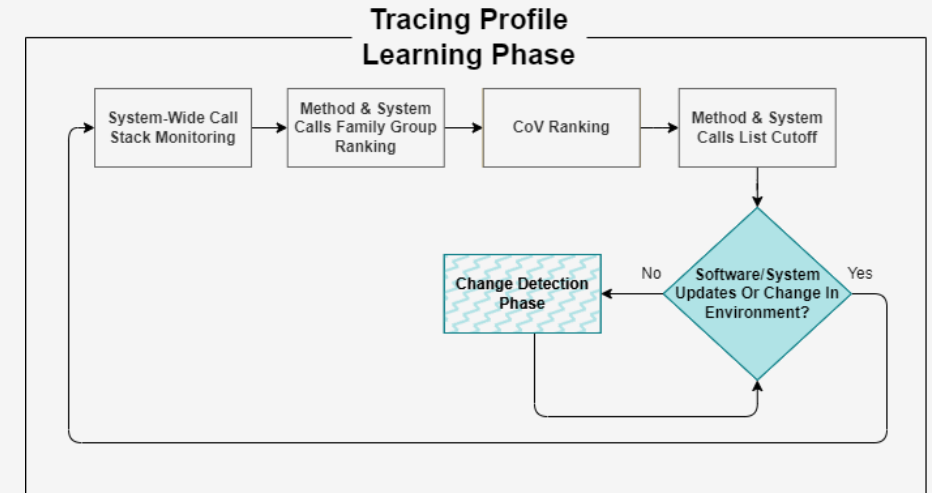
TBML Adaptive Tracing Method – Tracing Profile Learning Phase

- Select a reasonable cutoff number for the list. Example: top 500 method calls (~5 % of the methods) for Firefox used for PDF reading.

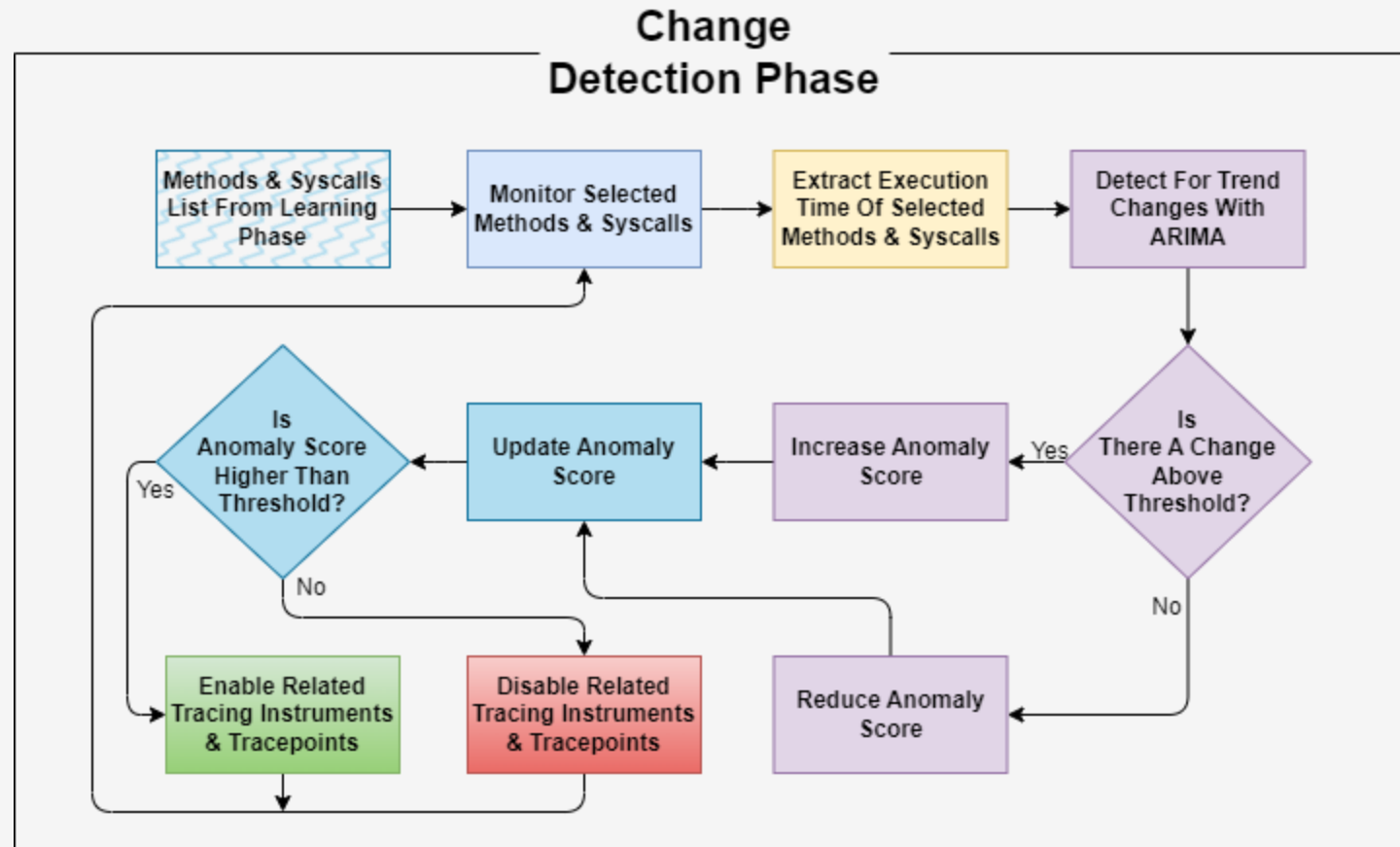


TBML Adaptive Tracing Method – Tracing Profile Learning Phase

- ❑ We must update the tracing profile whenever there is a change in the application usage, software usage, system updates, etc.
- ❑ Feed the methods and system calls list to follow to the change detection process.

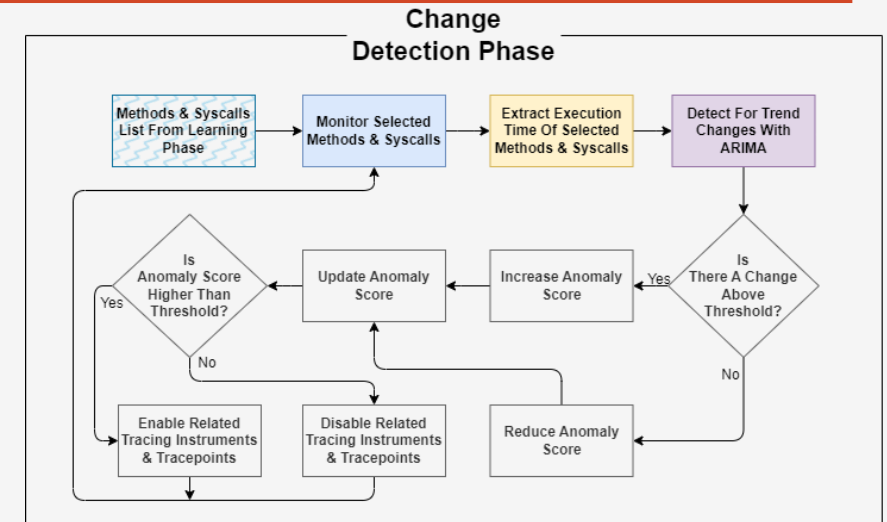


TBML Adaptive Tracing Method – Change Detection Phase



TBML Adaptive Tracing Method – Change Detection Phase

- ❑ Monitor selected methods and system calls continuously unless a changed list is supplied from the learning phase.
- ❑ Extract the methods and system calls' execution time.
- ❑ Predict the execution time at time T using autoregressive integrated moving average (ARIMA) while feeding it time series data till T-1.



TBML Adaptive Tracing Method – Change Detection Phase

- ❑ ARIMA is an analysis methodology that utilizes time series data along with statistical analysis to predict future values based on data trends.
- ❑ Components of ARIMA:
 - **Autoregression (AR):** a paradigm in which a moving variable is pushed back on its previous, or lag, values.
 - **Integrated (I):** depicts raw data separation in order for the time series towards becoming stabilized.
 - **Moving average (MA):** displays the relationship between a sample and the remaining error of a delayed moving average model.
- ❑ Input parameters of ARIMA:
 - **p:** the model's quantity of lag observations.
 - **d:** the extent to which the raw readings are differed.
 - **q:** the moving average window size.

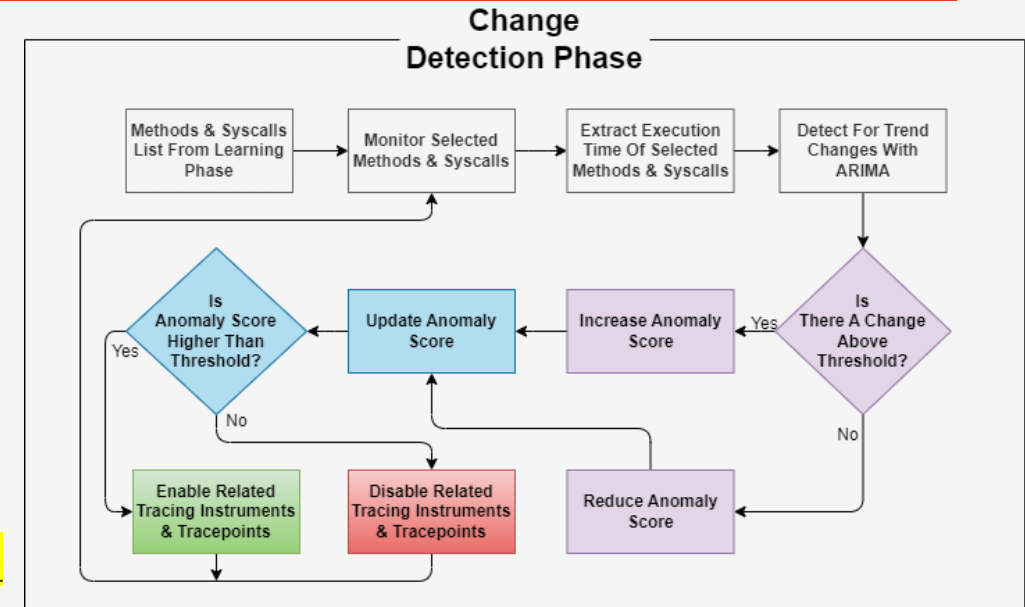
TBML Adaptive Tracing Method – Change Detection Phase

❑ Why choose ARIMA?

- ARIMA models make the implicit assumption about the future that it will be very similar to the past trends. As a result, in particular market situations, such as economic crises or periods of fast technology development, they may prove to be wrong.
- It is this drawback property of ARIMA that we are interested in.
- **A mismatch between actual reading and ARIMA's predicted reading would mean previous trend has been broken.**
- If the mismatch is beyond a threshold, then we could flag that method/syscall as problematic.

TBML Adaptive Tracing Method – Change Detection Phase

- ❑ Should we change our tracing configuration whenever there is a mismatch with ARIMA's prediction?
 - No! What if it was just one anomaly which is to never happen again? It's not worth it to keep changing tracing configuration every time for such occurrences which doesn't result in any noticeable performance issue.
- ❑ Anomaly Score:
 - **$\text{anomaly score} = (\beta * \text{is anomaly}) + ((1 - \beta) * \text{anomaly score})$**
 - Higher frequency of anomaly -> rapid increase of anomaly score.
 - Lower down of anomaly occurrence -> anomaly score drops down.
- ❑ If anomaly score is above threshold → enable related tracing instruments, tracepoints, events, etc.
- ❑ If anomaly score is below threshold → disable related tracing instruments, tracepoints, events, etc.



Case-studies

- ❑ Three different types of issues in Firefox (as a complex multithreaded application):
 - Issues with application method calls to graphics driver.
 - Specific methods in application source code level causing performance problems.
 - System level components and system calls responsible for performance problems.

Case-studies: Issues with method calls to a driver in Firefox

- ❑ Example1: Firefox Vsync issue with larger PDFs.
- ❑ Issue: problem between Firefox and the way it calls graphic driver functions.

 **clessili** Reporter
Description • 7 months ago

STR:

- download a large PDF (I used <https://fabiensanglard.net/gebbdoom.pdf>)
- try opening it locally
- opening it is slow (around 6 seconds) and a progress bar is shown




It seems the viewer does almost nothing during 6 seconds except showing the progress bar and triggering vsync: <https://share.firefox.dev/3M2bDCg>

Commenting line 858 of `toolkit/components/pdfjs/content/web/viewer.js`, (<https://searchfox.org/mozilla-central/source/toolkit/components/pdfjs/content/web/viewer.js#858>) responsible for showing the initial loading of the document (not its pages), brings the loading down to around 1 second.

An other clue of this: loading the PDF in a background tab also takes way less time than if the tab is in foreground.

1768481 - "loading bar" makes o x +

bugzilla.mozilla.org/show_bug.cgi?id=1768481

 Bugzilla  

Closed Bug 1768481 Opened 7 months ago Closed 16 days ago

"loading bar" makes opening a local PDF very slow

Categories


Product: Firefox
Component: PDF Viewer
Version: Firefox 102
Platform: x86_64 Windows 10



Type: defect
Priority: P2
Severity: S3

Tracking


Status: RESOLVED WORKSFORME
Project Flags: Performance Impact low



People

Assignee:  calixte

Reporter:  clessili
Triage Owner:  calixte
CC: 7 people

References

Depends on:  [4772598](#)

See Also:  [4668214](#)
 [1769023](#)

Details

Keywords: [perf](#), [perf:responsiveness](#)
Whiteboard: [pdfjs-performance]
Votes: 0

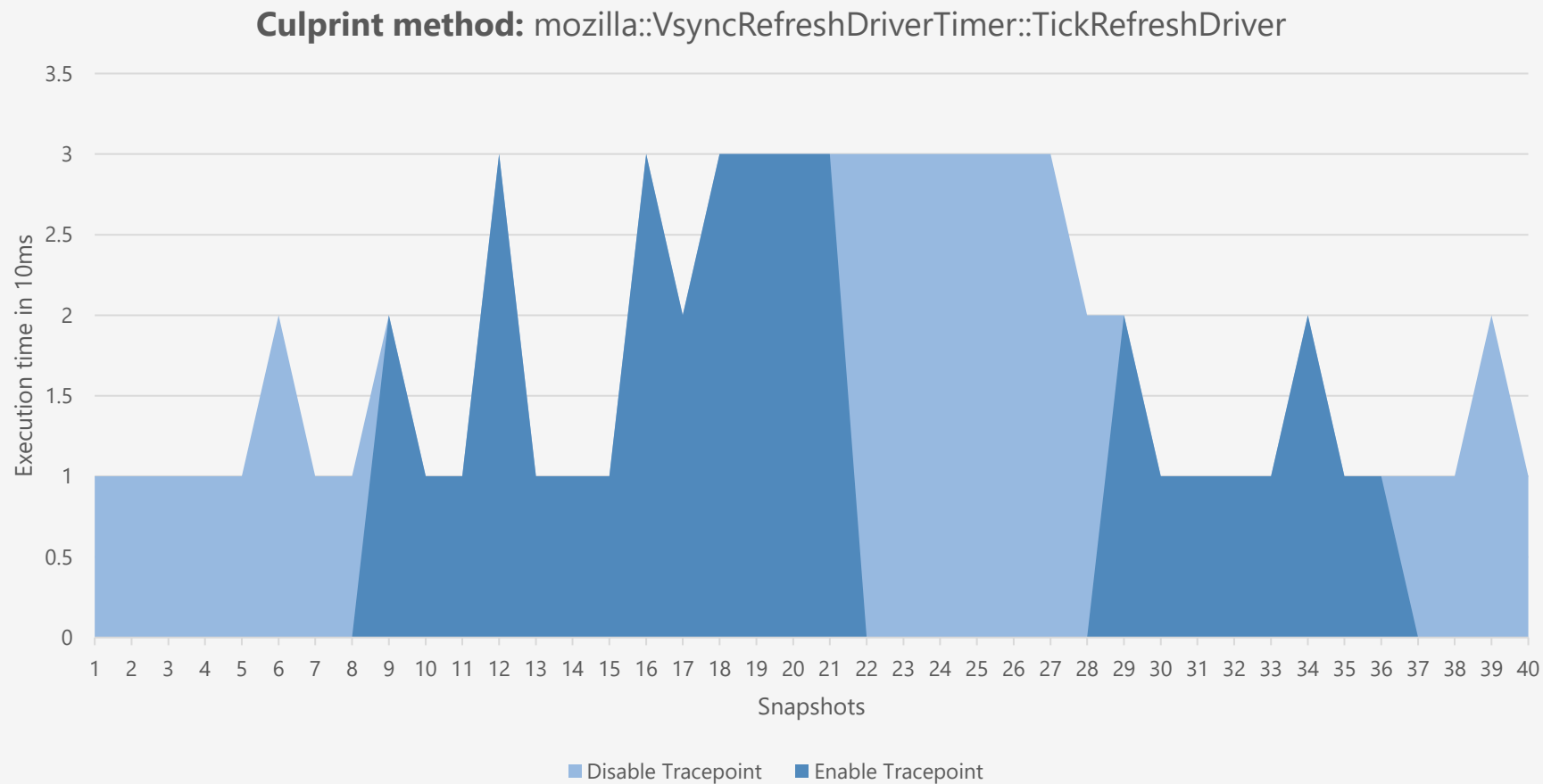
Case-studies: Issues with method calls to a driver in Firefox

- ❑ After the learning phase, only ~500 methods (5%) were selected to be followed out of ~10,000 method calls.
- ❑ Multi-level TAT method flagged Vsync related method calls within its sorted top 20 method calls list.

```
1 ('js::Call-----js::InternalCallOrConstruct', 30267)
2 ('js::InternalCallOrConstruct-----js::RunScript', 27707)
3 ('js::RunScript-----Interpret', 22644)
4 ('JS::Call-----js::Call', 8826)
5 ('js::jit::DoCallFallback-----js::InternalCallOrConstruct', 7347)
6 ('mozilla::TaskController::ExecuteNextTaskOnlyMainThreadInternal-----mozilla::TaskController::DoExecuteNextTaskOnlyMainThreadInternal', 6676)
7 ('mozilla::TaskController::DoExecuteNextTaskOnlyMainThreadInternal-----mozilla::RunnableTask::Run', 6317)
8 ('mozilla::TaskController::ProcessPendingMTTask-----mozilla::TaskController::ExecuteNextTaskOnlyMainThreadInternal', 6227)
9 ('PromiseReactionJob-----js::Call', 5245)
10 ('js::jit::InvokeFunction-----js::Call', 4958)
11 ('mozilla::RefreshDriverTimer::TickRefreshDrivers-----nsRefreshDriver::Tick', 4752)
12 ('std::panicking::try-----<core::panic::unwind_safe::AssertUnwindSafe<F> as core::ops::function::FnOnce<()>>::call_once', 4731)
13 ('mozilla::RefreshDriverTimer::Tick-----mozilla::RefreshDriverTimer::TickRefreshDrivers', 4598)
14 ('mozilla::VsyncRefreshDriverTimer::RunRefreshDrivers-----mozilla::RefreshDriverTimer::Tick', 4420)
15 ('mozilla::VsyncRefreshDriverTimer::TickRefreshDriver-----mozilla::VsyncRefreshDriverTimer::RunRefreshDrivers', 4242)
16 ('std::panic::catch_unwind-----std::panicking::try', 4202)
17 ('MessageLoop::Run-----mozilla::ipc::MessagePump::Run', 4052)
18 ('mozilla::detail::RunnableFunction<mozilla::TaskController::InitializeInternal():$2>::Run-----mozilla::TaskController::ProcessPendingMTTask', 4028)
19 ('mozilla::VsyncRefreshDriverTimer::NotifyVsyncOnMainThread-----mozilla::VsyncRefreshDriverTimer::TickRefreshDriver', 3803)
20 ('nsThread::ProcessNextEvent-----mozilla::detail::RunnableFunction<mozilla::TaskController::InitializeInternal():$2>::Run', 3673)
21 ('mozilla::VsyncRefreshDriverTimer::RefreshDriverVsyncObserver::NotifyVsyncTimerOnMainThread-----mozilla::VsyncRefreshDriverTimer::NotifyVsyncOnMainThread', 3639)
22 ('js::CallGetter-----js::Call', 3076)
23 ('MessageLoop::Run-----mozilla::ipc::MessagePumpForNonMainThread::Run', 2957)
24 ('entry_SYSCALL_64_after_hwframe-----do_syscall_64', 2906)
25 ('<core::panic::unwind_safe::AssertUnwindSafe<F> as core::ops::function::FnOnce<()>>::call_once-----std::sys_common::backtrace::__rust_begin_short_backtrace', 2682)
26 ('webrender::renderer::Renderer::render-----webrender::renderer::Renderer::render_impl', 2672)
27 ('wr_renderer_render-----webrender::renderer::Renderer::render', 2507)
28 ('mozilla::VsyncRefreshDriverTimer::RefreshDriverVsyncObserver::NotifyVsync-----
    mozilla::VsyncRefreshDriverTimer::RefreshDriverVsyncObserver::NotifyVsyncTimerOnMainThread', 2451)
29 ('mozilla::wr::RendererOGL::UpdateAndRender-----wr_renderer_render', 2327)
```


Case-studies: Issues with method calls to a driver in Firefox

- ❑ How the change detection phase adjusts tracepoints:




Case-studies: Specific method call in Firefox causing performance problems

❑ Firefox 3d transformation rendering issue:


 **Jeff Muizelaar** [jrmuizel] Reporter
Comment 4 • 3 years ago

How about here: <https://searchfox.org/mozilla-central/rev/9f074fab9bf905fad62e7cc32faf121195f4ba46/gfx/layers/wr/WebRenderCommandBuilder.cpp#303>

Flags: needinfo?(jmuizelaar)


 **Jeff Muizelaar** [jrmuizel] Reporter
Comment 5 • 3 years ago

FWIW, upstream fixed/improved the issue: <https://github.com/jonobr1/two.js/commit/ff9ade2308da3e28b43c9d8ed1adea79646d9ca8>

 **Glenn Watson** [gw]
Comment 6 • 4 months ago

This is still quite bad - looks like it's all in blob rasterization.


Flags: needinfo?(tnikkel)

 **Timothy Nikkel** (tnikkel)
Updated • 4 months ago

Flags: needinfo?(tnikkel)


You need to [log in](#) before you can comment on or make changes to this bug.

[←](#) [→](#) [C](#) [bugzilla.mozilla.org/show_bug.cgi?id=1637586](#) [🔗](#)



 **Bugzilla** 🔍 [New Account](#) [Log](#)


Open Bug 1637586 Opened 3 years ago Updated 4 months ago

Terrible performance on <https://looping-squares.superhi.com>

Categories
Product: Core ▾
Component: Graphics: WebRender ▾
Type:  defect
Priority: Not set Severity: S3

Tracking
Status: NEW

People
Assignee: Unassigned
Reporter:  jrmuizel
Triage Owner:  gw
NeedInfo From: **tnikkel** 4 months ago
CC: 9 people

References
Blocks:  1637580
Dependency tree / graph

Details
Votes: 4

Case-studies: Specific method call in Firefox causing performance problems

- ❑ The problematic method call was found in the top 10 by following the ranked list of 1000 (5%) method calls. Actual number of unique callee and caller pair methods exceeded over 20,000.

[illegible]

Case-studies: Network connectivity issue causing high loading time with Firefox

❑ No network connectivity and DNS issue:

 **james.cook** Reporter
Description • 6 months ago

User Agent: Mozilla/5.0 (X11; OpenBSD amd64; rv:101.0) Gecko/20100101 Firefox/101.0

Steps to reproduce:

Tried to start Firefox (using the command "firefox") when my network connection was poor --- specifically, it seems to happen when my OS thinks I have a network connection, but it is temporarily not working, e.g. a flaky mobile hotspot.

I am using OpenBSD current. I think this has been happening for years; haven't bothered to report it until now.


I noticed https://bugzilla.mozilla.org/show_bug.cgi?id=433665 looks similar, in that it involves DNS and slow startup, but I don't know if this has anything to do with profile locking.

Actual results:

It takes a long time --- maybe a minute? --- before any Firefox windows appears.

I briefly looked at ktrace output. I see a bunch of stuff at 0 seconds, then a bunch of stuff at 5 seconds, then at 15, then 35... I guess it is retrying something. Here is the block at 15 seconds:

```
90429 firefox 15.067960 STRU struct kevent
90429 firefox 15.067968 STRU struct pollfd { fd=8, events=0x1<POLLIN>, revents=0<> }
90429 firefox 15.067969 RET poll 0
90429 firefox 15.067974 CALL recvfrom(8 0x29fabh1f000 0x1000 0 0 0)
```

 Bugzilla New Account

Open Bug 1776469 Opened 6 months ago Updated 5 months ago

very slow startup when network is broken (I suspect DNS)

Categories

Product: Core
Component: Networking
Version: Firefox 101

Type: defect
Priority: P3
Severity: S3

Tracking

Status: UNCONFIRMED

People

Assignee: Unassigned

Reporter: james.cook
Triage Owner: jesup
CC: 2 people

Details

Whiteboard: [necko-triaged]
Votes: 0

Attachments

[log files; moz_log.txt.moz_log.0 is clipped after 1000 lines](#)
5 months ago james.cook
19.86 KB, application/octet-stream

Case-studies: Network connectivity issue causing high loading time with Firefox

- ❑ Sendmsg system call was flagged.
- ❑ Linux system call table shows its related to net/socket.c which is the network component of Linux kernel.

The screenshot shows a debugger window with two tabs: 'output2.txt' and 'output1.txt'. The 'output2.txt' tab displays a stack trace with line numbers 576 through 603. A red circle highlights the entry at line 579: `('do_syscall_64-----x64_sys_sendmsg', 156)`. The 'output1.txt' tab displays a system call table entry for `sendmsg`. The entry is highlighted in orange and blue. The table has three columns: `%rdi`, `%rsi`, and `%rdx`. The corresponding values are `int fd`, `struct msghdr __user * msg`, and `unsigned int flags`. The file `net/socket.c` is also listed.

%rdi	%rsi	%rdx
int fd	struct msghdr __user * msg	unsigned int flags

Evaluation

1. Effectiveness:

- ❑ **Informativeness:** Can we collect relevant information of the issues(s) whenever they are needed without tracing the whole application?
 - ✓ Three Firefox issues showed that the proposed method is effective
- ❑ **On-time:** How early we detect the change and start collecting the data about the issue?
 - ✓ By adjusting the β value of the anomaly score function and the anomaly threshold, depending on the seriousness of the application or system, we could adjust how early and how fast we would want to start collecting data about an issue.

2. Overhead: What is the computational overhead of this method?

- ❑ **Learning phase of tracing-profile**
 - ❑ It is done offline
 - ❑ Depends on how many times you would want to run your application to learn.
 - ❑ Under different runtime issues
- ❑ **Runtime Analysis**
 - ❑ Collecting the trace of the selected functions
 - ❑ User space tracing with LTTng or
 - ❑ Profiling
 - ❑ Change detection based on Arima
 - ❑ Based on each function-tracing

Conclusion

- ❑ Adaptive tracing is a challenging task.
- ❑ We proposed a trend based adaptive tracing method.
- ❑ Our method can handle multi-level adaptive tracing (application and system level).
- ❑ Our method successfully tracked performance issues in the following categories:
 - ✓ Issues with driver and application communication.
 - ✓ Specific methods in application source code level causing performance problems.
 - ✓ System level components and system calls responsible for performance problems.

Thank you!



ak19qp@brocku.ca

<https://github.com/ak19qp/dorsal-conference/>