NodeCompass: A new Paragdigm for Peformance Evaluation in Event-based Node.js

Progress Report Meeting

Hervé KABAMBA

PhD Candidate

Supervisor: Michel Dagenais

May 16, 2022

Polytechnique Montréal

Département de Génie Informatique et Génie Logiciel

Agenda

1. Introduction

Agenda

1. Introduction
2. Architecture

Agenda

1. Introduction
2. Architecture
3. Use Cases

Agenda

Introduction

NodeCompass: A new Paradigm?

Introduction

NodeCompass: A new Paradigm?

Yes. Performance Analysis is done differently

Introduction

NodeCompass: A new Paradigm?

Yes. Performance Analysis is done differently

We consider all layers interacting in the system as part of the Perf. Analysis

Introduction

NodeCompass: A new Paradigm?

Yes. Performance Analysis is done differently

We consider all layers interacting in the system as part of the Perf. Analysis

Reduced Instrumentation overhead by using LTTng in the JavaScript land

Introduction

NodeCompass: A new Paradigm?

Yes. Performance Analysis is done differently

We consider all layers interacting in the system as part of the Perf. Analysis

Reduced Instrumentation overhead by using LTTng in the JavaScript land

Focus on High Level of Granularity in the Perf. Analysis

Introduction

NodeCompass: A new Paradigm?

Yes. Performance Analysis is done differently

We consider all layers interacting in the system as part of the Perf. Analysis

Reduced Instrumentation overhead by using LTTng in the JavaScript land

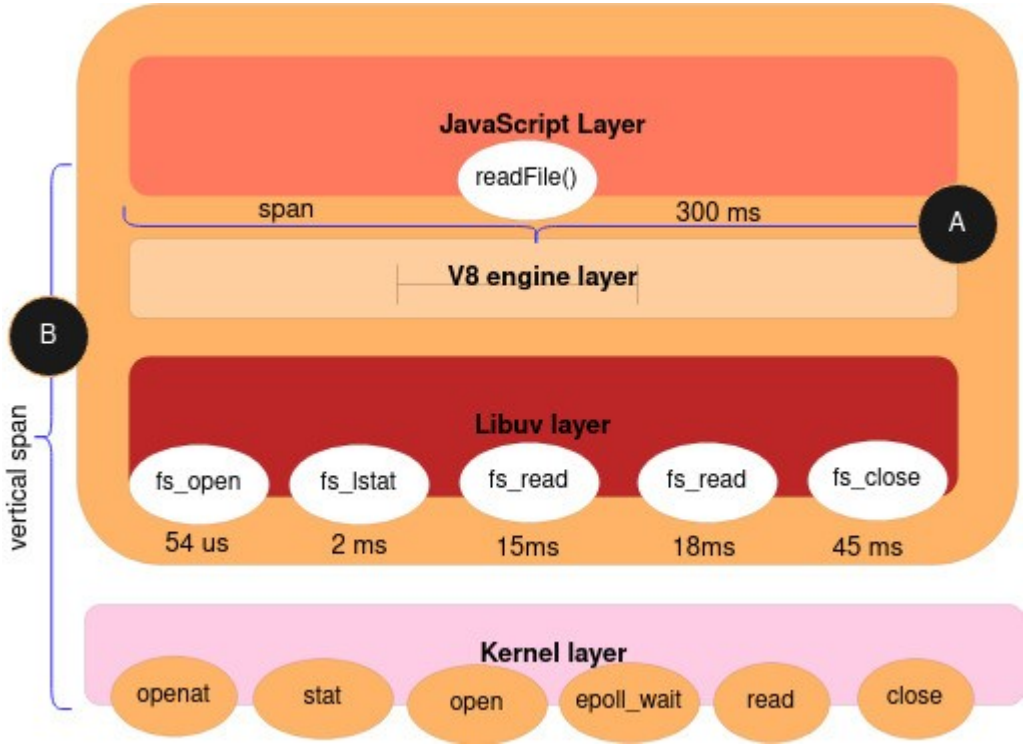
Focus on High Level of Granularity in the Perf. Analysis

Uncover Bugs, Bottlenecks, Root causes, Race conditions either in Node.js internals or in JavaScript land

Introduction

Why is NodeCompass different?

Current approaches :

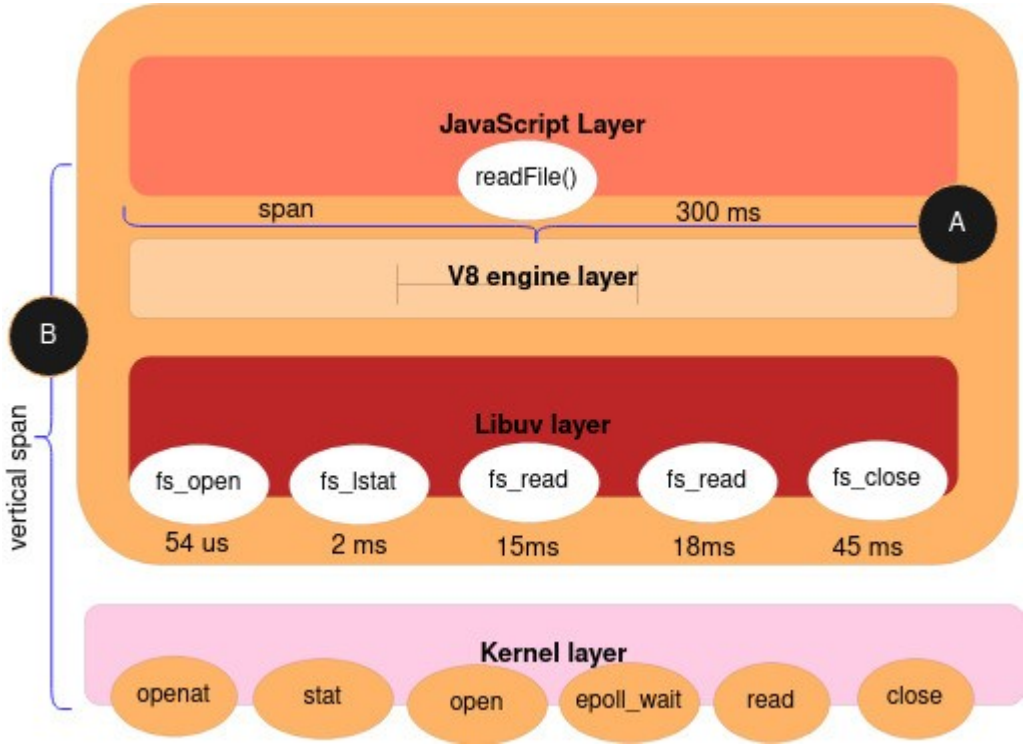


Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer

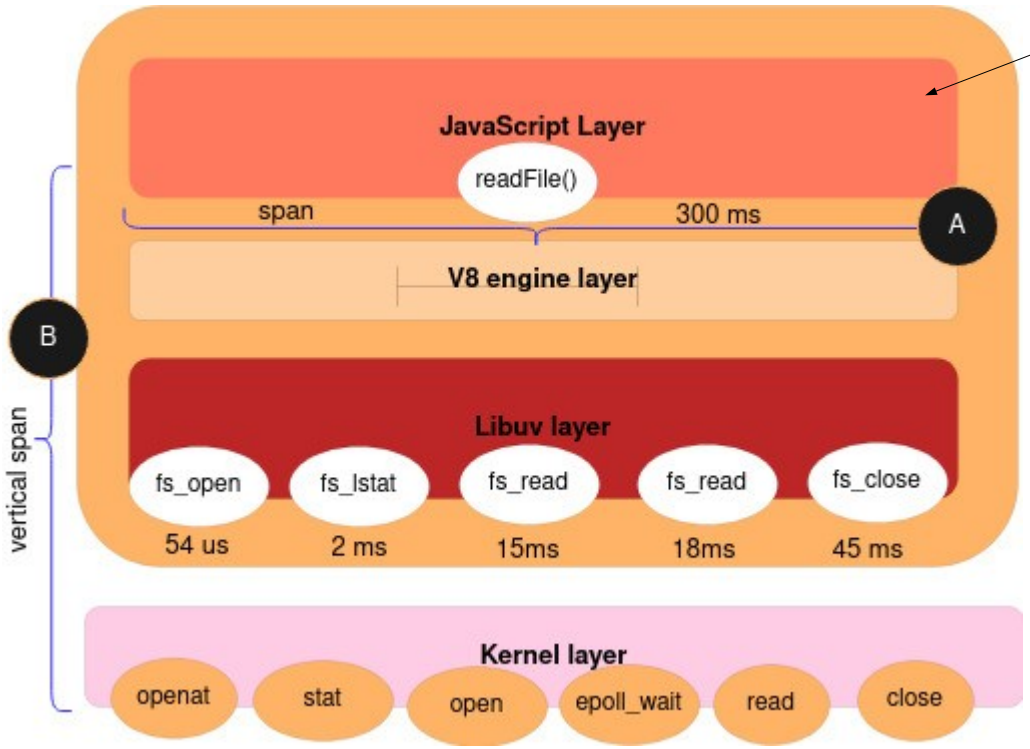


Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer



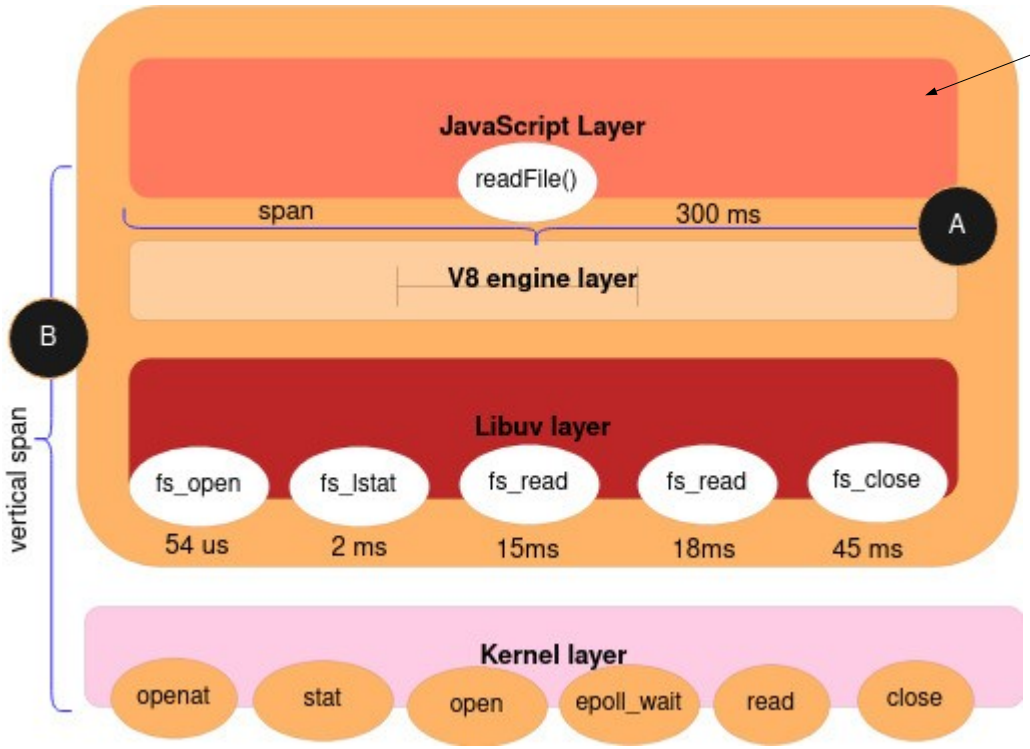
Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer

- Distributed tracers

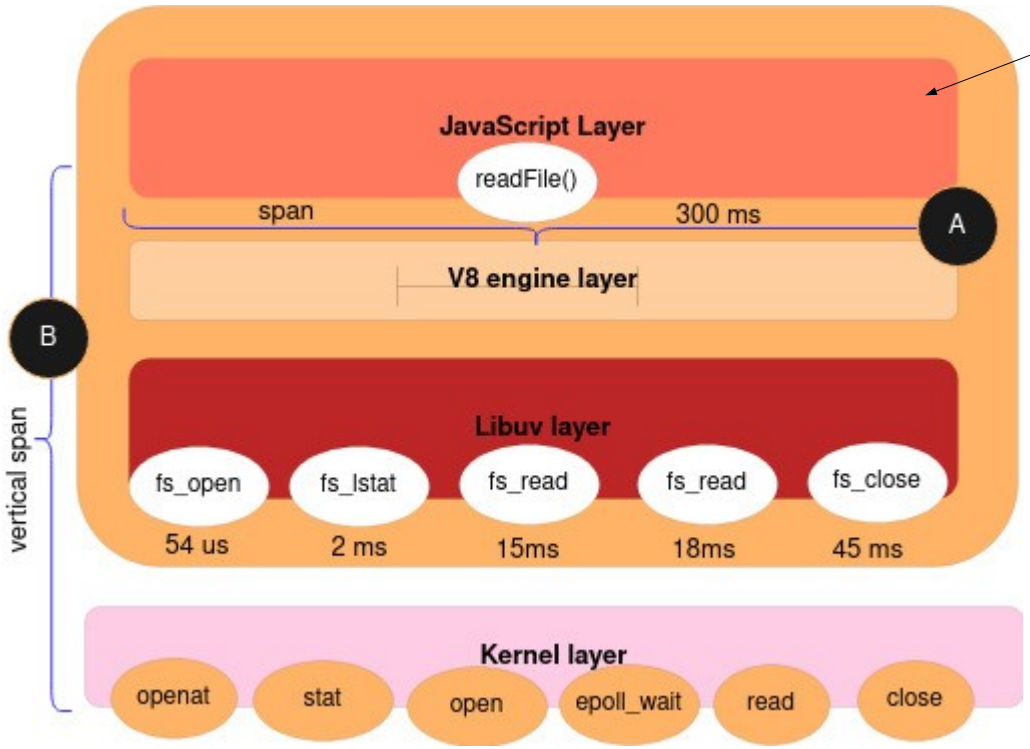


Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer
- Distributed tracers
- Spans are exposed

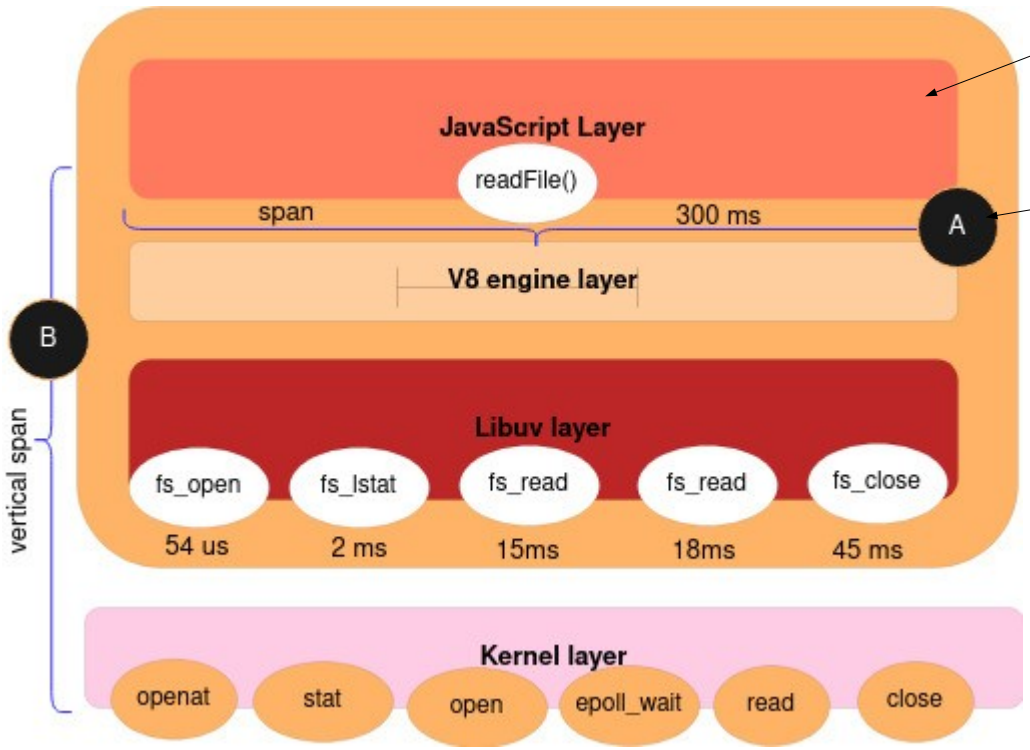


Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer
- Distributed tracers
- Spans are exposed

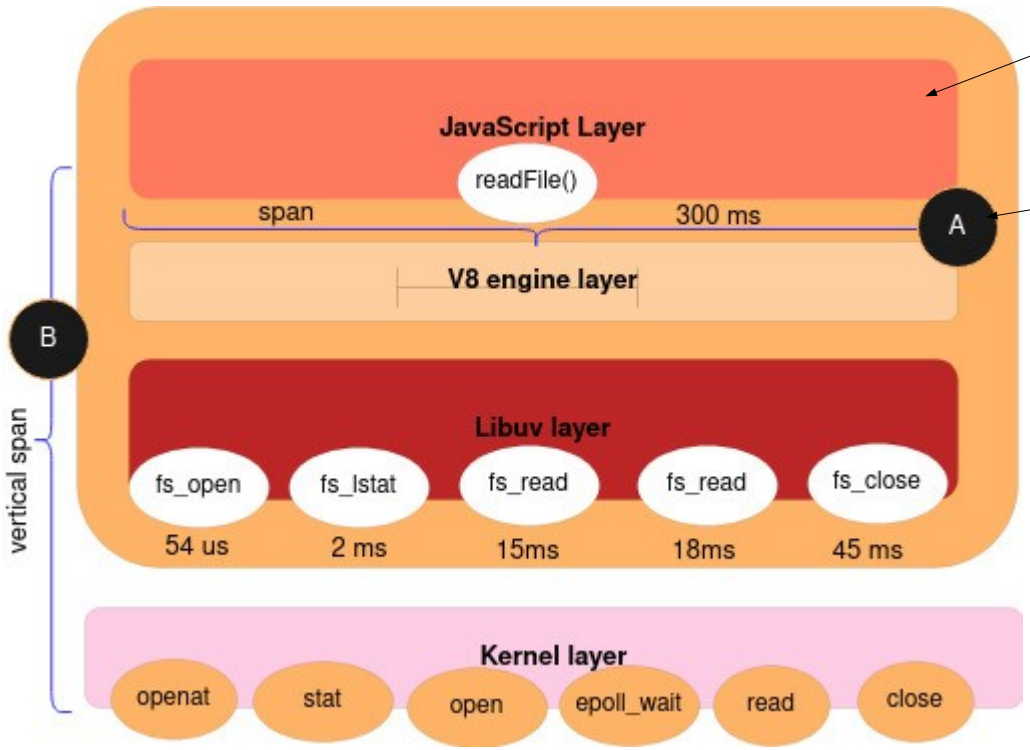


Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer
- Distributed tracers
- Spans are exposed



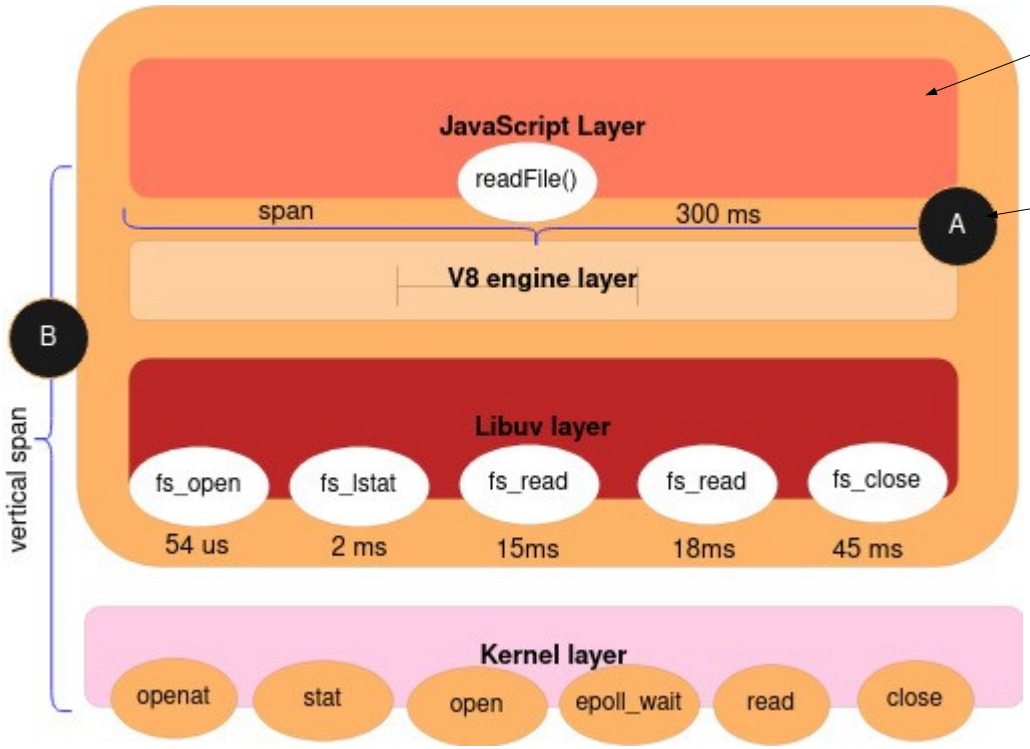
Cons

Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer
- Distributed tracers
- Spans are exposed



Cons

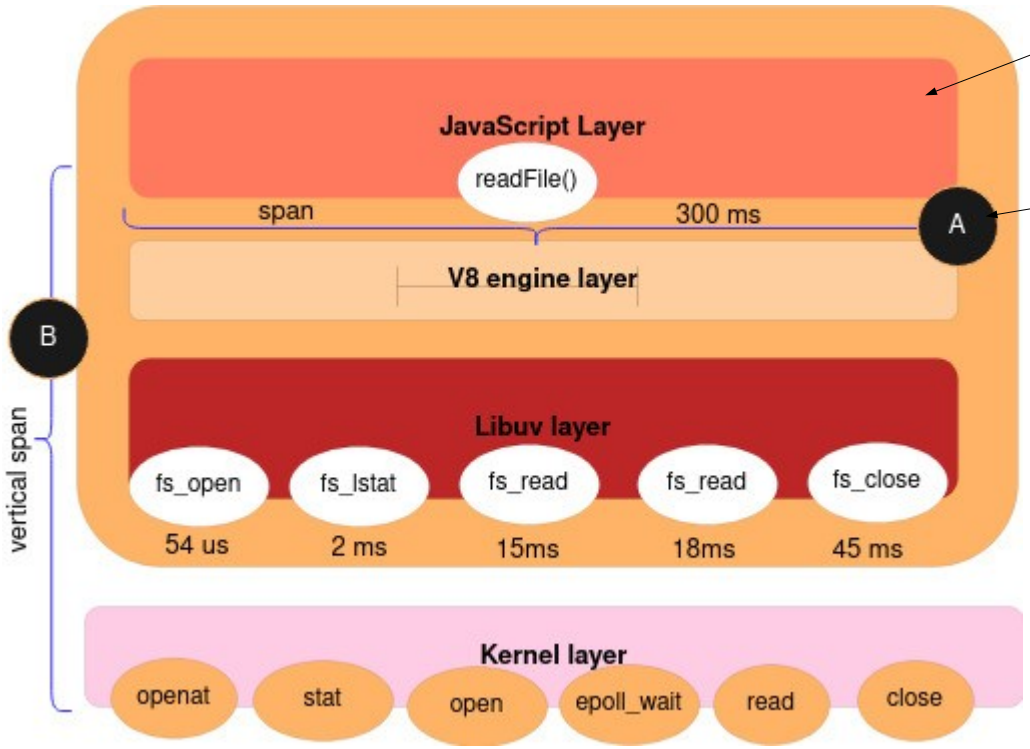
- Give only indications of the function latency

Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer
- Distributed tracers
- Spans are exposed



Cons

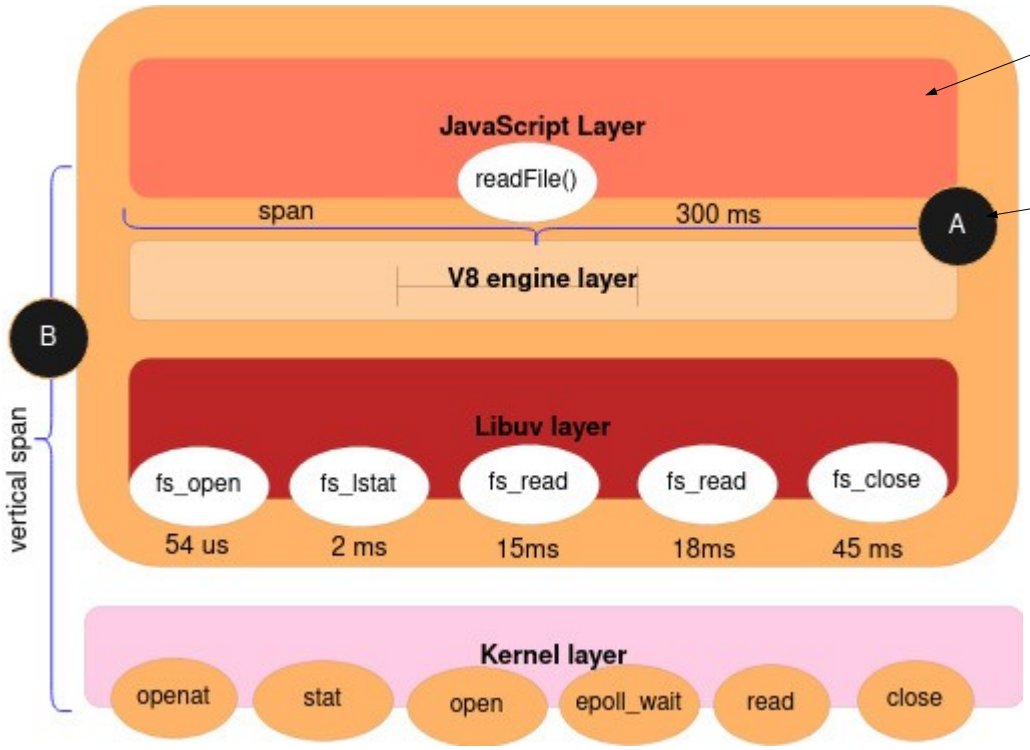
- Give only indications of the function latency
- Lack granularity

Introduction

Why is NodeCompass different?

Current approaches :

- Focus on higher layer
- Distributed tracers
- Spans are exposed

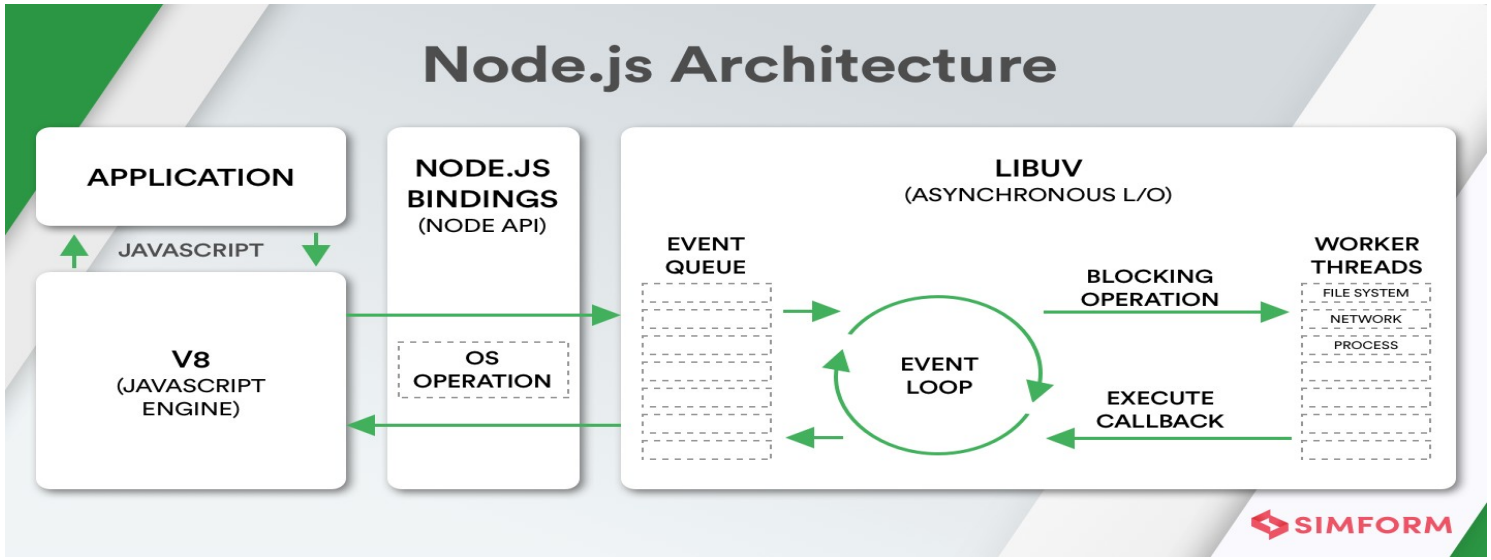


Cons

- Give only indications of the function latency
- Lack granularity
- Too much effort to understand the root cause based on the given information

Introduction

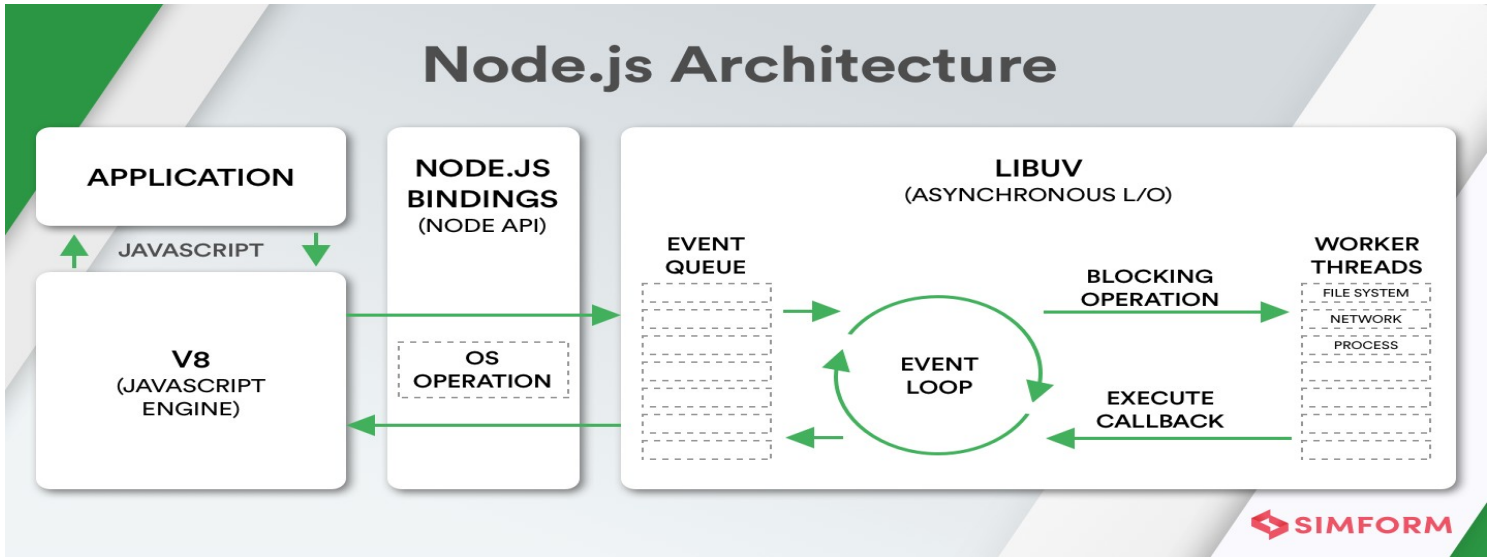
NodeCompass approach for Perf. Analysis



Introduction

NodeCompass approach for Perf. Analysis

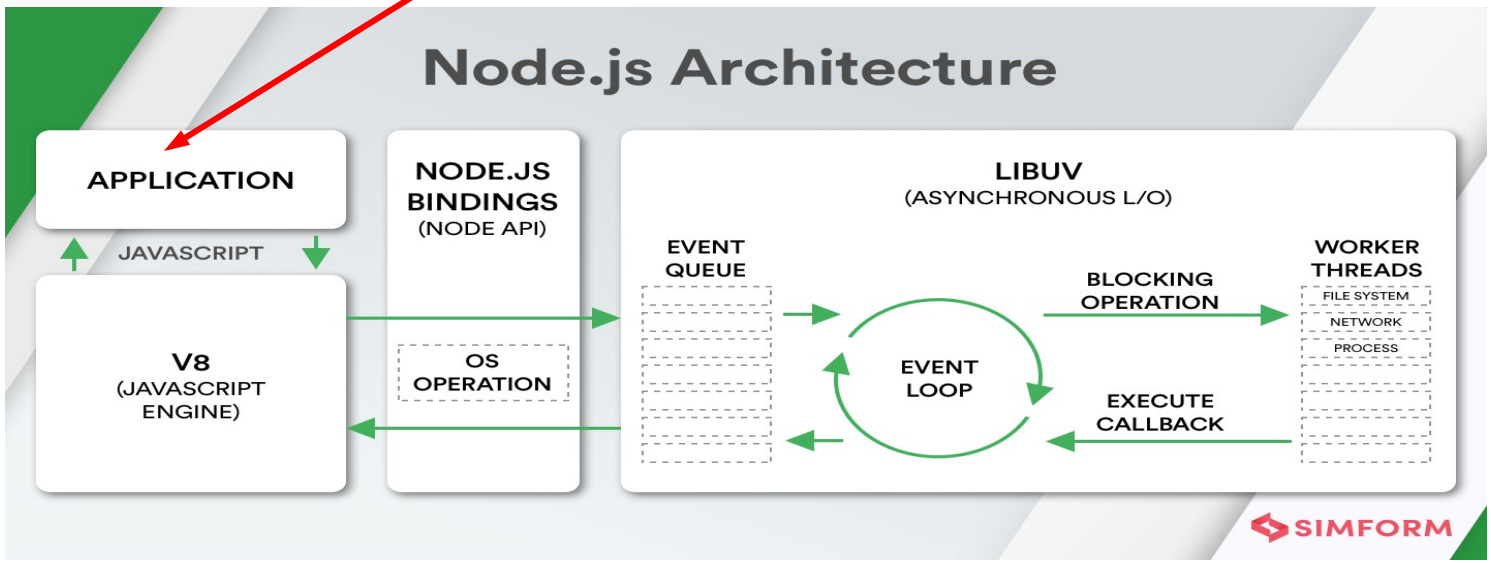
Tracepoints are inserted in the JavaScript land



Introduction

NodeCompass approach for Perf. Analysis

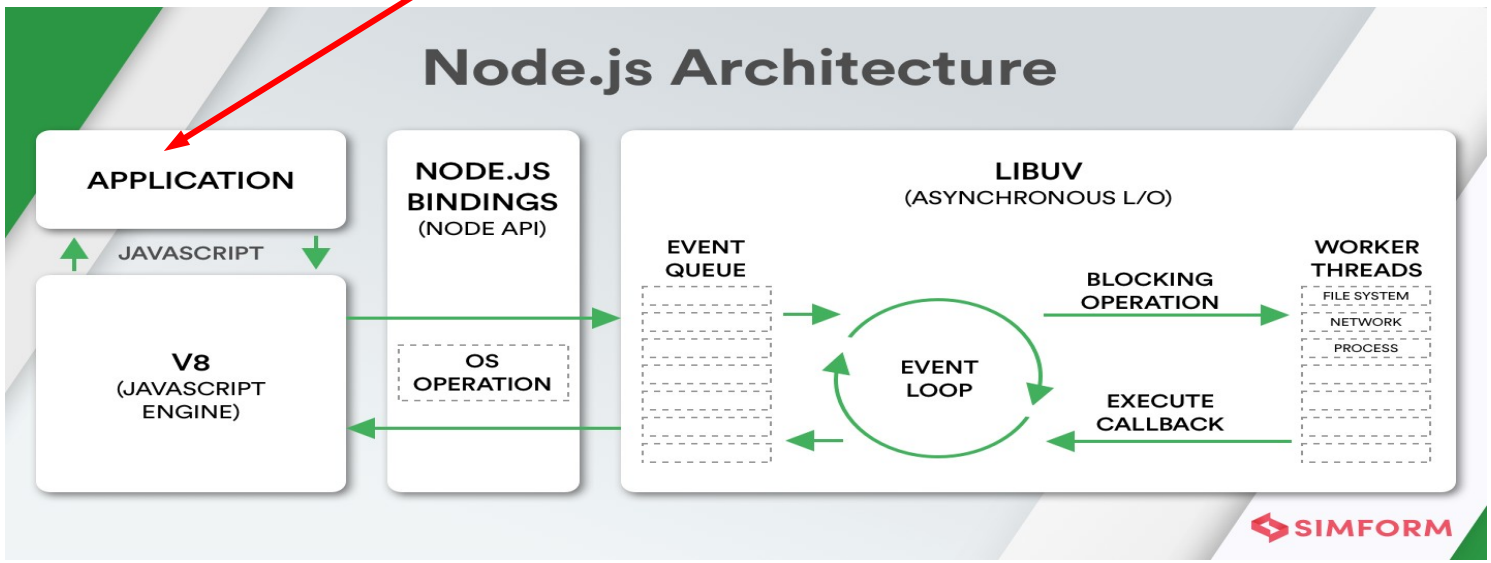
Tracepoints are inserted in the JavaScript land



Introduction

NodeCompass approach for Perf. Analysis

Tracepoints are inserted in the JavaScript land

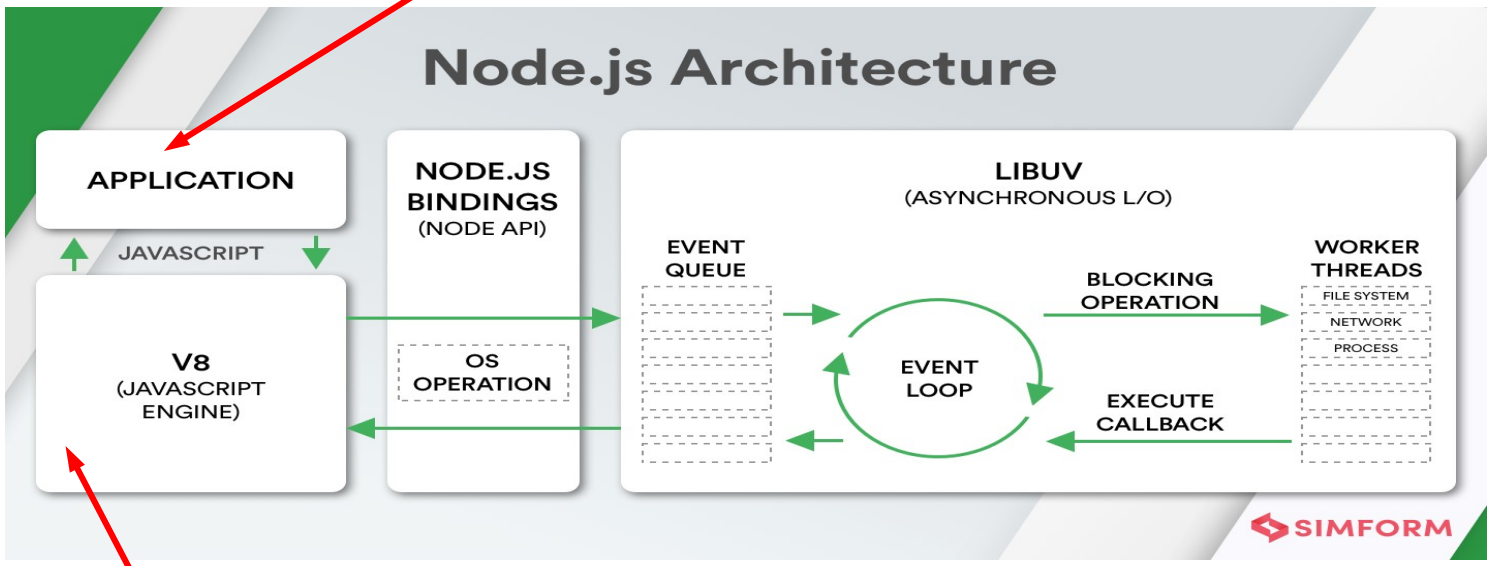


Tracepoints are inserted in the V8 OS

Introduction

NodeCompass approach for Perf. Analysis

Tracepoints are inserted in the JavaScript land

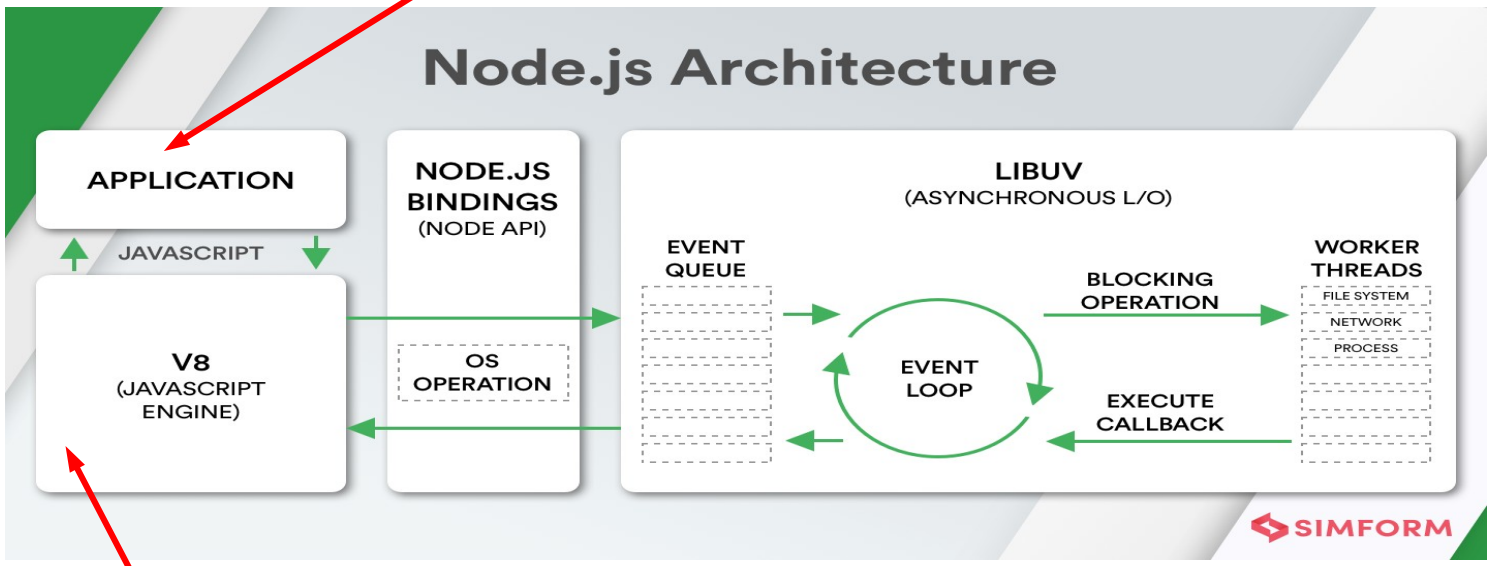


Tracepoints are inserted in the V8 OS

Introduction

NodeCompass approach for Perf. Analysis

Tracepoints are inserted in the JavaScript land



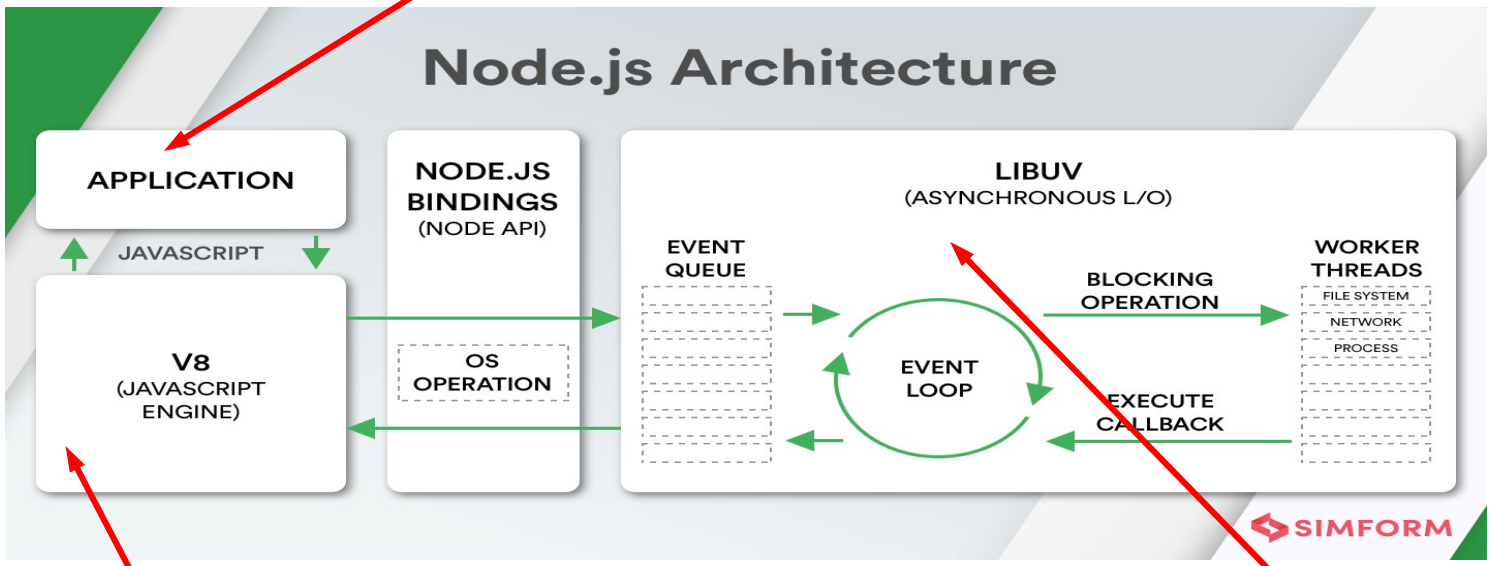
Tracepoints are inserted in the orchestration engine

Tracepoints are inserted in the V8 OS

Introduction

NodeCompass approach for Perf. Analysis

Tracepoints are inserted in the JavaScript land



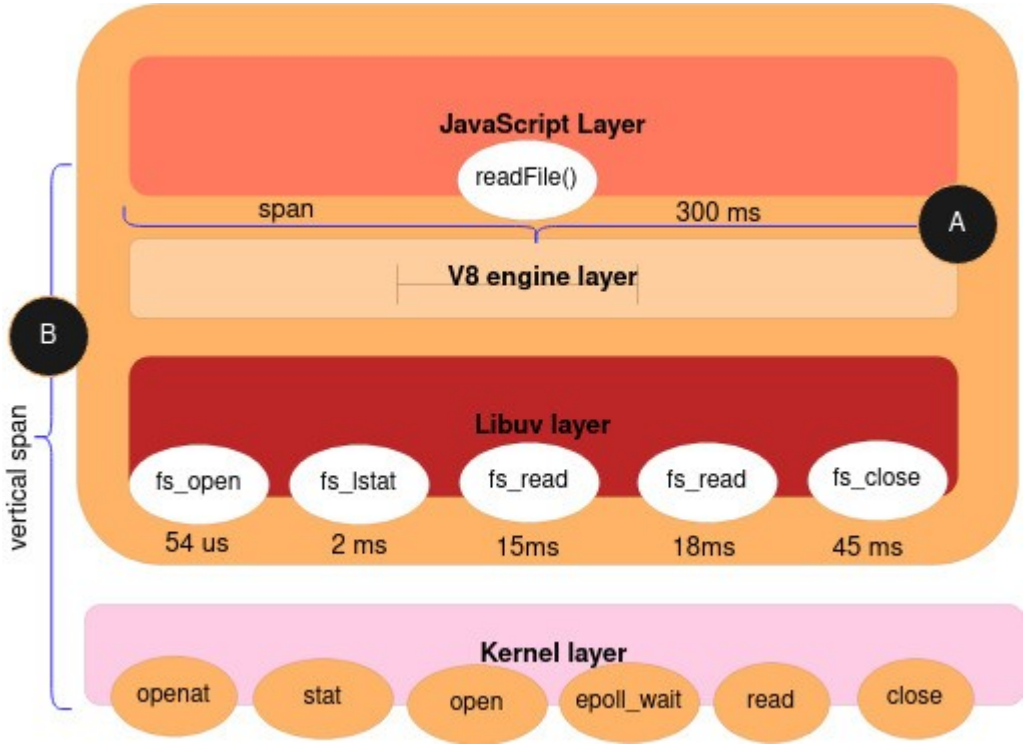
Tracepoints are inserted in the orchestration engine

Tracepoints are inserted in the V8 OS

Introduction

NodeCompass approach

New paradigm:

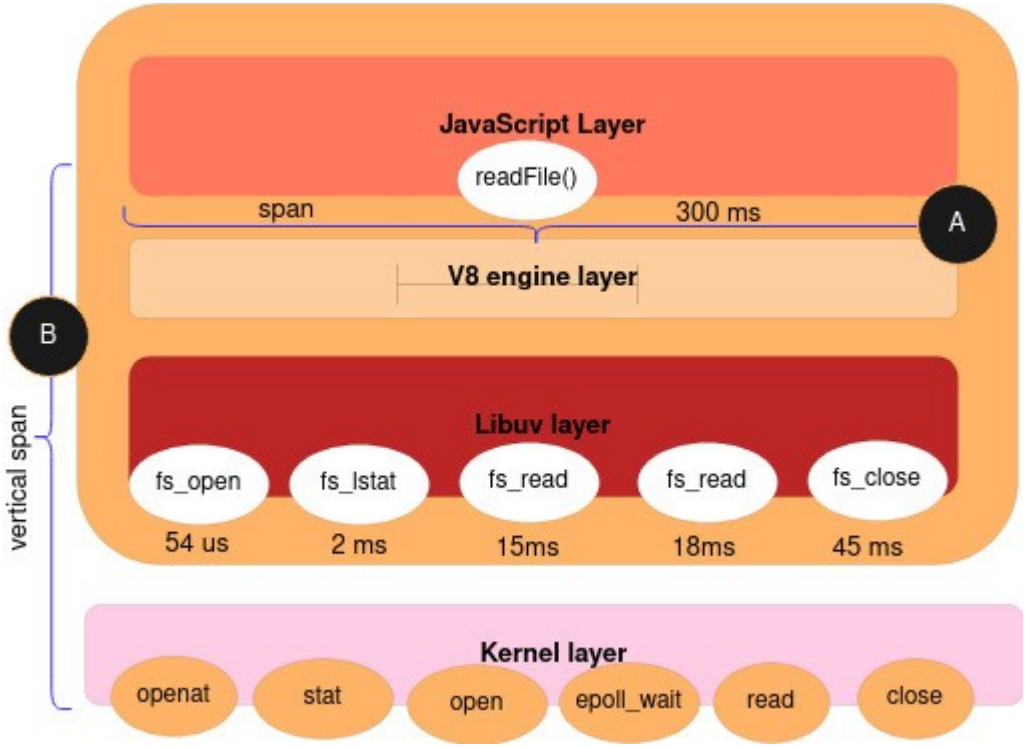


Introduction

NodeCompass approach

New paradigm:

- Introduce Vertical Span Model (VSM)

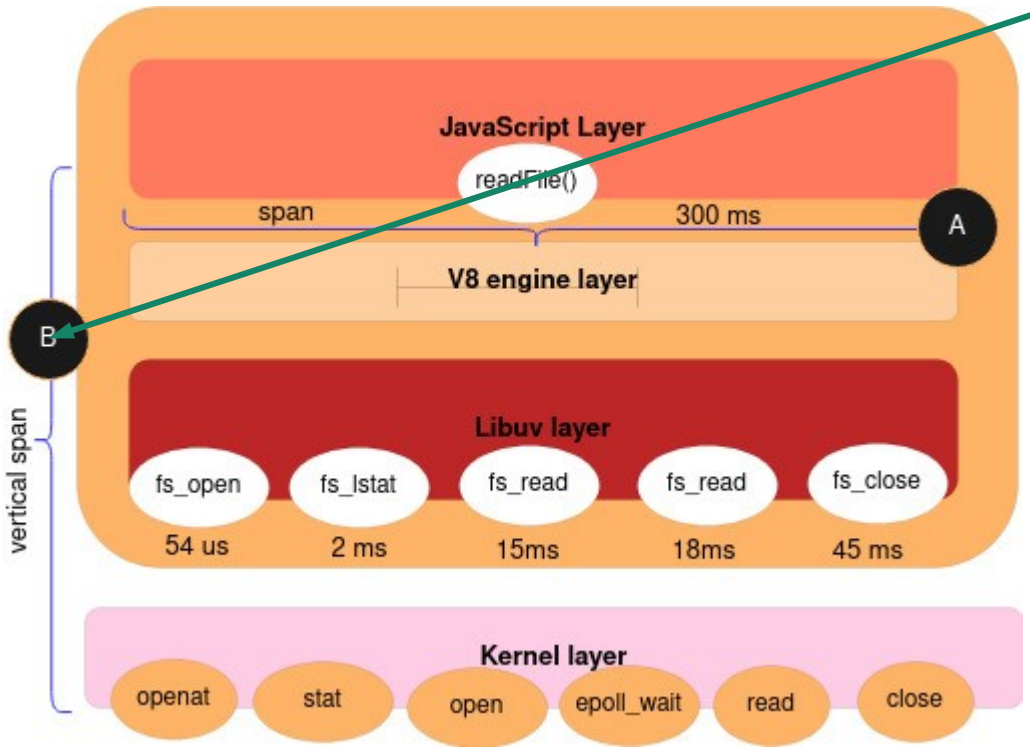


Introduction

NodeCompass approach

New paradigm:

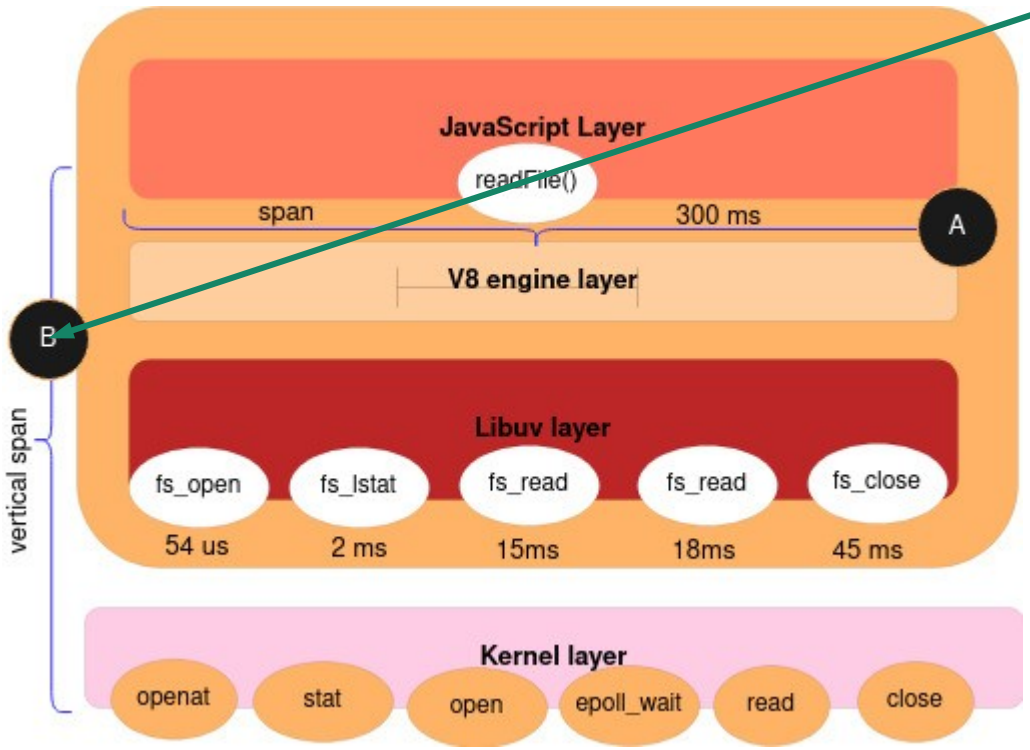
Introduce Vertical Span Model (VSM)



Introduction

NodeCompass approach

New paradigm:



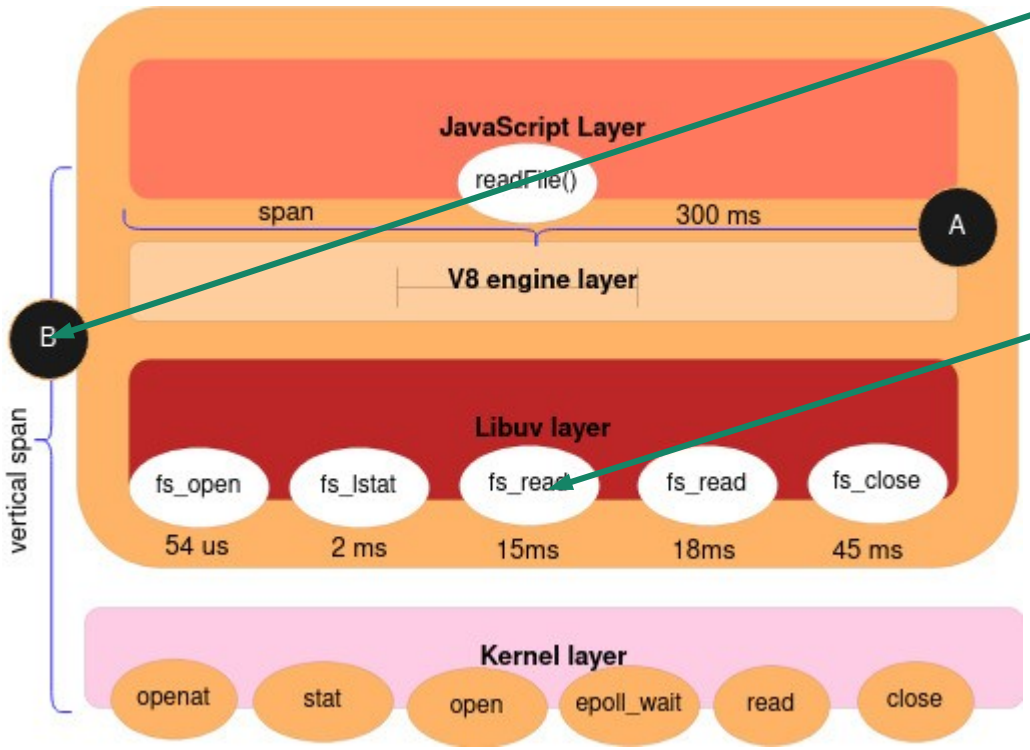
Introduce Vertical Span Model (VSM)

Consider atomic operations in Node

Introduction

NodeCompass approach

New paradigm:



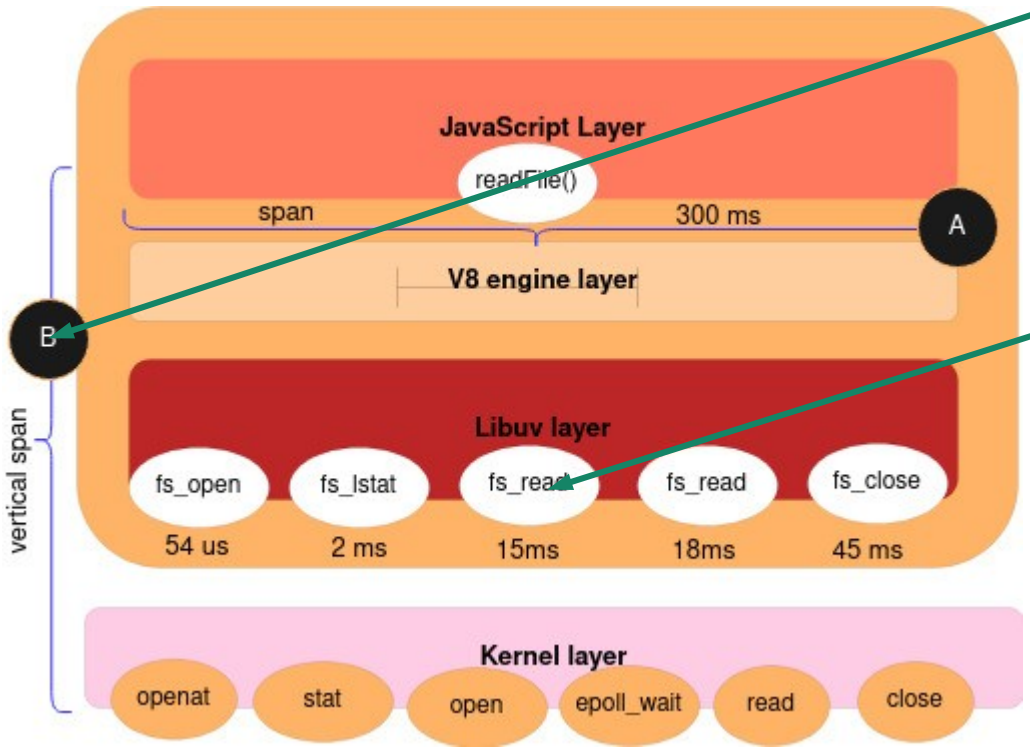
Introduce Vertical Span Model (VSM)

Consider atomic operations in Node

Introduction

NodeCompass approach

New paradigm:



Introduce Vertical Span Model (VSM)

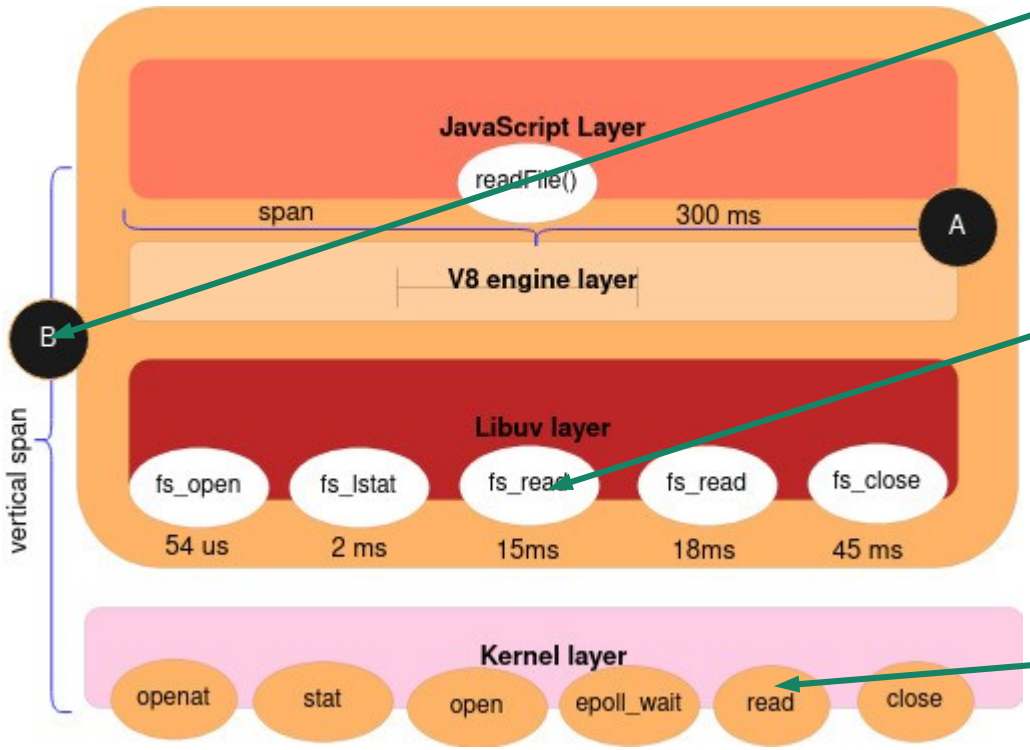
Consider atomic operations in Node

Map atomic operations with Syscalls

Introduction

NodeCompass approach

New paradigm:



Introduce Vertical Span Model (VSM)

Consider atomic operations in Node

Map atomic operations with Syscalls

Introduction

The Vertical Span Model: The new paradigm

Introduction

The Vertical Span Model: The new paradigm

Represent a request with all its underlying atomic operations

Introduction

The Vertical Span Model: The new paradigm

Represent a request with all its underlying atomic operations

Expose the application flow and the request underlying paths

Introduction

The Vertical Span Model: The new paradigm

Represent a request with all its underlying atomic operations

Expose the application flow and the request underlying paths

Expose sequence of events in each layer with its latency

Introduction

The Vertical Span Model: The new paradigm

Represent a request with all its underlying atomic operations

Expose the application flow and the request underlying paths

Expose sequence of events in each layer with its latency

Performance analysis with high granularity degree

Introduction

The Vertical Span Model: The new paradigm

Represent a request with all its underlying atomic operations

Expose the application flow and the request underlying paths

Expose sequence of events in each layer with its latency

Performance analysis with high granularity degree

Pinpoint performance bottlenecks, system bugs, Root cause

Introduction

The Vertical Span Model: The new paradigm

Represent a request with all its underlying atomic operations

Expose the application flow and the request underlying paths

Expose sequence of events in each layer with its latency

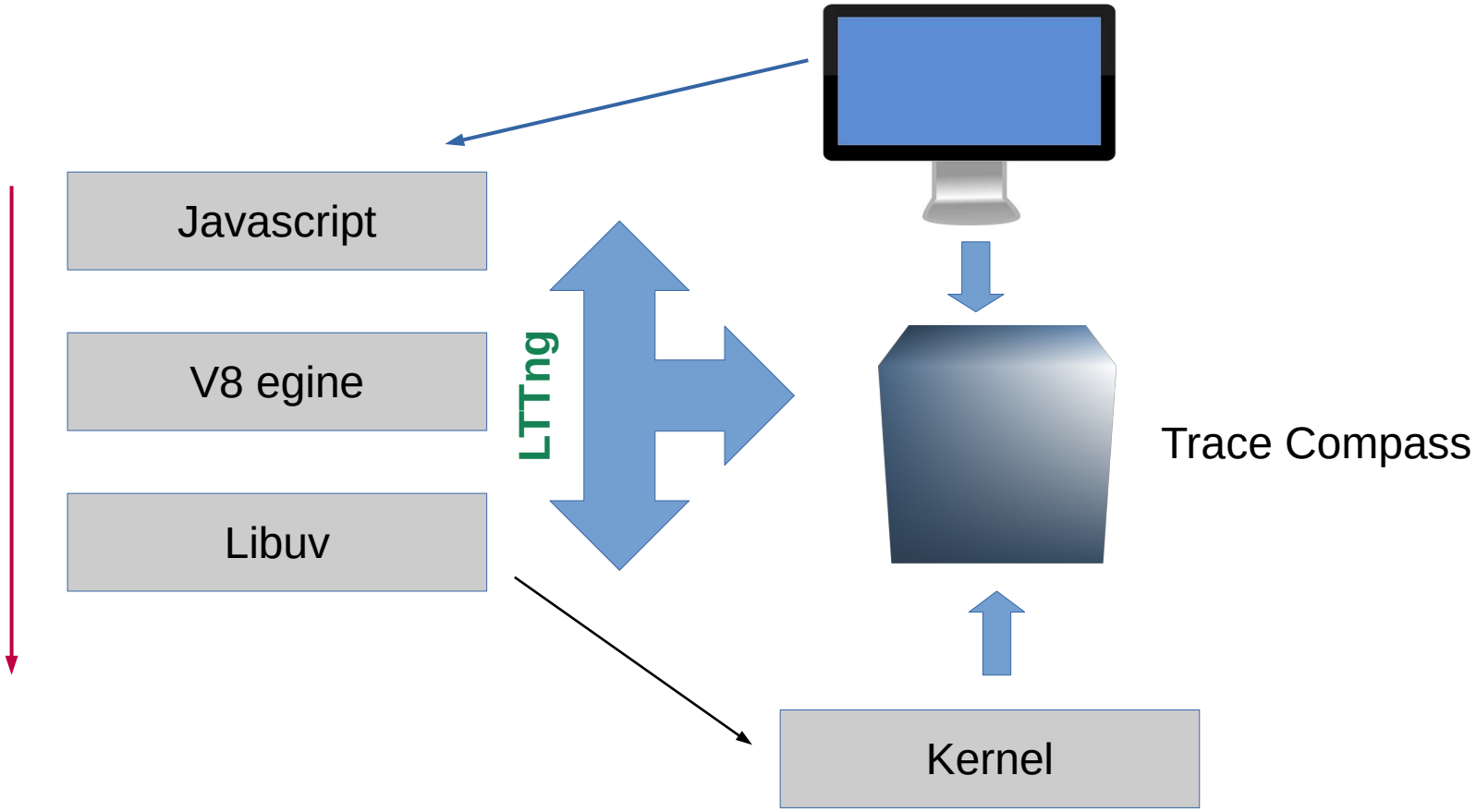
Performance analysis with high granularity degree

Pinpoint performance bottlenecks, system bugs, Root cause

Expose race Conditions in Node.js

Introduction

- **System Architecture**



Architecture

User level-Instrumentation Example

```
calcula                                  Aa ab .*  7 of 12
565     }
566     function traceSendingRequest(message) {
567         if (trace === Trace.Off || !tracer) {
568             return;
569         }
570         if (traceFormat === TraceFormat.Text) {
571             let data = undefined;
572             if (trace === Trace.Verbose && message.params) {
573                 data = `Params: ${JSON.stringify(message.params, null, 4)}\n\n`;
574             }
575             //tracer.log(`Sending request '${message.method} - (${message.id})'.`, data);
576             calculate.send_event(message.id, async_hooks.executionAsyncId(), "js_open_" + message.method + '$server');
577         }
578         else {
579             logLSPMessage('send-request', message);
580         }
581     }
```

Architecture

User level-Instrumentation Example

```
calcula                                  Aa ab .*  7 of 12
565     }
566     function traceSendingRequest(message) {
567         if (trace === Trace.Off || !tracer) {
568             return;
569         }
570         if (traceFormat === TraceFormat.Text) {
571             let data = undefined;
572             if (trace === Trace.Verbose && message.params) {
573                 data = `Params: ${JSON.stringify(message.params, null, 4)}\n\n`;
574             }
575             //tracer.log(`Sending request '${message.method} - (${message.id})'.`, data);
576             calculate.send_event(message.id, async_hooks.executionAsyncId(), "js_open_" + message.method + '$server');
577         }
578         else {
579             logLSPMessage('send-request', message);
580         }
581     }
```

Architecture

User level-Instrumentation Example

```
565     }
566     function traceSendingRequest(message) {
567         if (trace === Trace.Off || !tracer) {
568             return;
569         }
570         if (traceFormat === TraceFormat.Text) {
571             let data = undefined;
572             if (trace === Trace.Verbose && message.params) {
573                 data = `Params: ${JSON.stringify(message.params, null, 4)}\n\n`;
574             }
575             //tracer.log(`Sending request '${message.method} - (${message.id})'.`, data);
576             calculate.send_event(message.id, async_hooks.executionAsyncId(), "js_open_" + message.method + '$server');
577         }
578         else {
579             logLSPMessage('send-request', message);
580         }
581     }
```

calcula Aa ab .* 7 of 12

Entry point (RPC)

Architecture

User level-Instrumentation Example

```
565     }
566     function traceSendingRequest(message) {
567         if (trace === Trace.Off || !tracer) {
568             return;
569         }
570         if (traceFormat === TraceFormat.Text) {
571             let data = undefined;
572             if (trace === Trace.Verbose && message.params) {
573                 data = `Params: ${JSON.stringify(message.params, null, 4)}\n\n`;
574             }
575             //tracer.log(`Sending request '${message.method} - (${message.id})'.`, data);
576             calculate.send_event(message.id, async_hooks.executionAsyncId(), "js_open_" + message.method + '$server');
577         }
578         else {
579             logLSPMessage('send-request', message);
580         }
581     }
```

calcula Aa ab .* 7 of 12

Entry point (RPC)

Architecture

User-Level Instrumentation Example

```
664    function traceReceivedResponse(message, responsePromise) {
665        if (trace === Trace.Off || !tracer) {
666            return;
667        }
668        if (traceFormat === TraceFormat.Text) {
669            let data = undefined;
670            if (trace === Trace.Verbose) {
671                if (message.error && message.error.data) {
672                    data = `Error data: ${JSON.stringify(message.error.data, null, 4)}\n\n`;
673                }
674                else {
675                    if (message.result) {
676                        data = `Result: ${JSON.stringify(message.result, null, 4)}\n\n`;
677                    }
678                    else if (message.error === void 0) {
679                        data = 'No result returned.\n\n';
680                    }
681                }
682            }
683            if (responsePromise) {
684                let error = message.error ? ` Request failed: ${message.error.message} (${message.error.code}).` : '';
685                //tracer.log(`Received response '${responsePromise.method} - (${message.id})' in ${Date.now() - responseProm
686                calculate.send_event(message.id, async_hooks.executionAsyncId(), 'js_exit$server');
687            }
```

Architecture

User-Level Instrumentation Example

```
664    function traceReceivedResponse(message, responsePromise) {
665        if (trace === Trace.Off || !tracer) {
666            return;
667        }
668        if (traceFormat === TraceFormat.Text) {
669            let data = undefined;
670            if (trace === Trace.Verbose) {
671                if (message.error && message.error.data) {
672                    data = `Error data: ${JSON.stringify(message.error.data, null, 4)}\n\n`;
673                }
674                else {
675                    if (message.result) {
676                        data = `Result: ${JSON.stringify(message.result, null, 4)}\n\n`;
677                    }
678                    else if (message.error === void 0) {
679                        data = 'No result returned.\n\n';
680                    }
681                }
682            }
683            if (responsePromise) {
684                let error = message.error ? ` Request failed: ${message.error.message} (${message.error.code}).` : '';
685                //tracer.log(`Received response '${responsePromise.method} - (${message.id})' in ${Date.now() - responseProm
686                calculate.send_event(message.id, async_hooks.executionAsyncId(), 'js_exit$server');
687            }
```

Exit Point (RPC)

Architecture

User-Level Instrumentation Example

```
664    function traceReceivedResponse(message, responsePromise) {
665        if (trace === Trace.Off || !tracer) {
666            return;
667        }
668        if (traceFormat === TraceFormat.Text) {
669            let data = undefined;
670            if (trace === Trace.Verbose) {
671                if (message.error && message.error.data) {
672                    data = `Error data: ${JSON.stringify(message.error.data, null, 4)}\n\n`;
673                }
674                else {
675                    if (message.result) {
676                        data = `Result: ${JSON.stringify(message.result, null, 4)}\n\n`;
677                    }
678                    else if (message.error === void 0) {
679                        data = 'No result returned.\n\n';
680                    }
681                }
682            }
683            if (responsePromise) {
684                let error = message.error ? ` Request failed: ${message.error.message} (${message.error.code}).` : '';
685                //tracer.log(`Received response '${responsePromise.method} - (${message.id})' in ${Date.now() - responseProm
686                calculate.send_event(message.id, async_hooks.executionAsyncId(), 'js_exit$server');
687            }
```

Exit Point (RPC)

Architecture

User-Level Instrumentation Example

```
664    function traceReceivedResponse(message, responsePromise) {
665        if (trace === Trace.Off || !tracer) {
666            return;
667        }
668        if (traceFormat === TraceFormat.Text) {
669            let data = undefined;
670            if (trace === Trace.Verbose) {
671                if (message.error && message.error.data) {
672                    data = `Error data: ${JSON.stringify(message.error.data, null, 4)}\n\n`;
673                }
674                else {
675                    if (message.result) {
676                        data = `Result: ${JSON.stringify(message.result, null, 4)}\n\n`;
677                    }
678                    else if (message.error === void 0) {
679                        data = 'No result returned.\n\n';
680                    }
681                }
682            }
683            if (responsePromise) {
684                let error = message.error ? ` Request failed: ${message.error.message} (${message.error.code}).` : '';
685                //tracer.log(`Received response '${responsePromise.method} - (${message.id})' in ${Date.now() - responseProm
686                calculate.send_event(message.id, async_hooks.executionAsyncId(), 'js_exit$server');
687            }
```

Exit Point (RPC)

Use case 1

Pinpointing process.NexTick Performance Issues

Use case 1

Pinpointing process.NexTick Performance Issues

Postpone function execution to the Event-loop next tick

Use case 1

Pinpointing process.NexTick Performance Issues

Postpone function execution to the Event-loop next tick

Queue must be exhausted before the Event-loop continues to next phases

Use case 1

Pinpointing process.NexTick Performance Issues

Postpone function execution to the Event-loop next tick

Queue must be exhausted before the Event-loop continues to next phases

Must be used with caution since it can slow or block the Event-loop

Nextick Performance degradation scenario

Use case 1

Pinpointing process.NexTick Performance Issues

Postpone function execution to the Event-loop next tick

Queue must be exhausted before the Event-loop continues to next phases

Must be used with caution since it can slow or block the Event-loop

Nextick Performance degradation scenario

An express server contacted on the rout /nexttick

Use case 1

Pinpointing process.NexTick Performance Issues

Postpone function execution to the Event-loop next tick

Queue must be exhausted before the Event-loop continues to next phases

Must be used with caution since it can slow or block the Event-loop

Nextick Performance degradation scenario

An express server contacted on the rout /nexttick

A client ***A*** contacting continuously the server at the route /nexttick

Use case 1

Pinpointing process.NexTick Performance Issues

Postpone function execution to the Event-loop next tick

Queue must be exhausted before the Event-loop continues to next phases

Must be used with caution since it can slow or block the Event-loop

Nextick Performance degradation scenario

An express server contacted on the rout /nexttick

A client ***A*** contacting continuously the server at the route /nexttick

A second client ***B*** contacting continuously the server at the route /checkState

Use case 1

Pinpointing process.NexTick Performance Issues

Postpone function execution to the Event-loop next tick

Queue must be exhausted before the Event-loop continues to next phases

Must be used with caution since it can slow or block the Event-loop

Nextick Performance degradation scenario

An express server contacted on the rout /nexttick

A client ***A*** contacting continuously the server at the route /nexttick

A second client ***B*** contacting continuously the server at the route /checkState

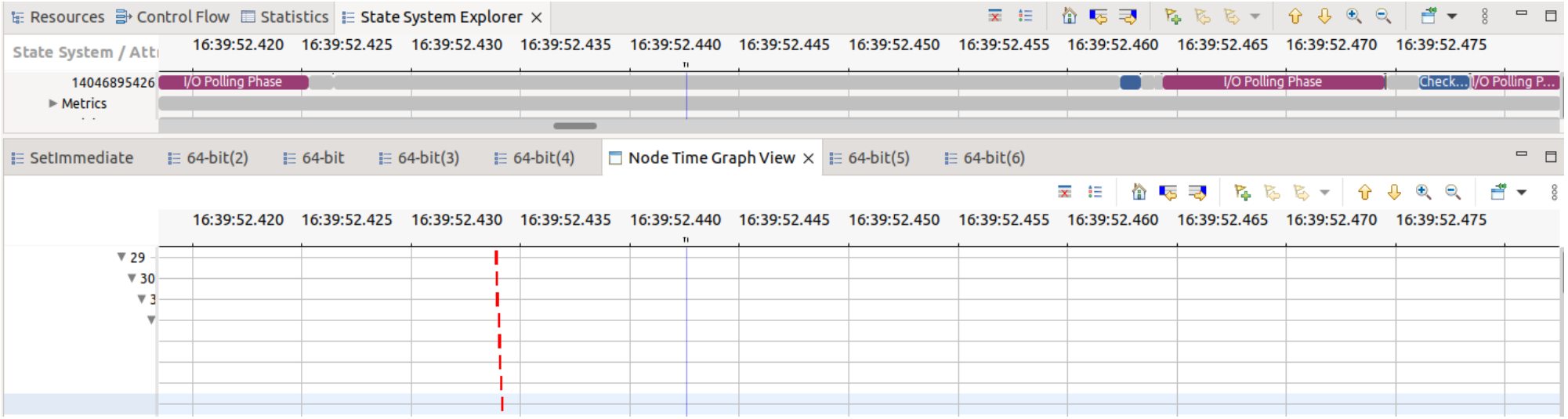
After a few seconds the server stops responding to B. A performance issue appears

Use case 1

Resulting flow:

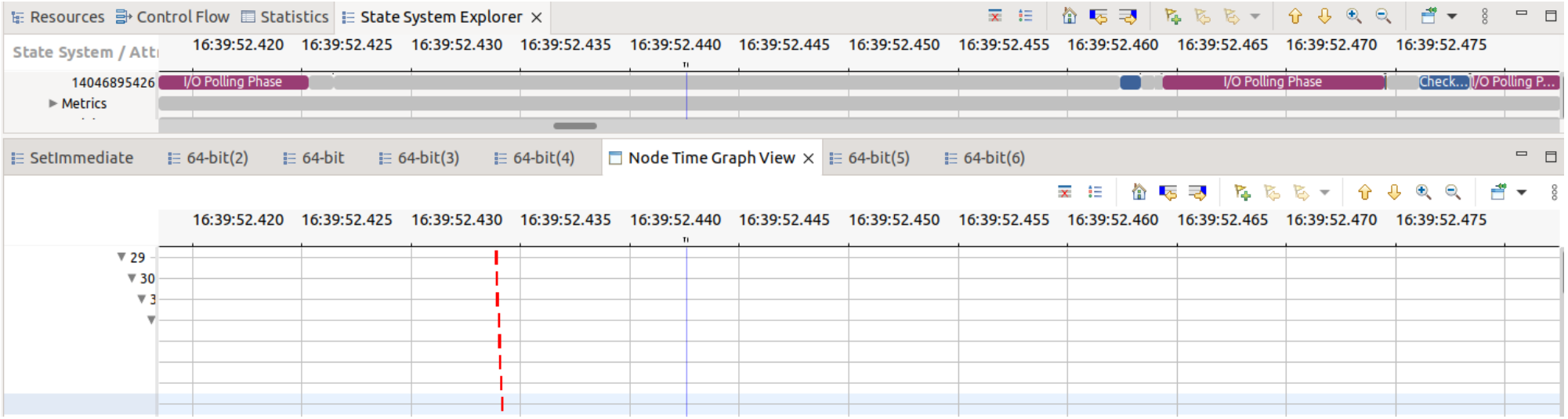
Use case 1

Resulting flow:



Use case 1

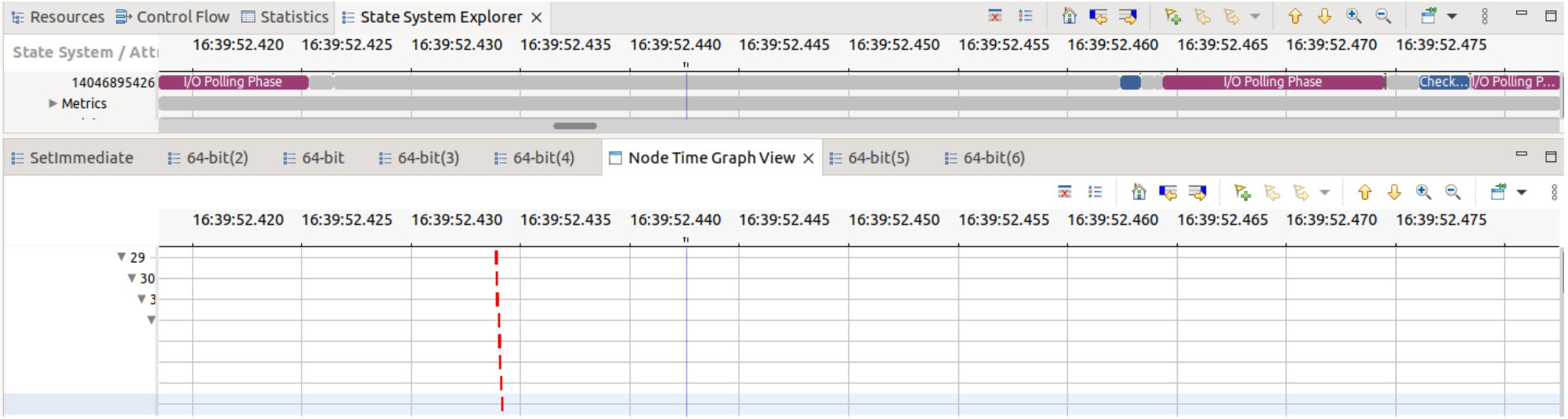
Resulting flow:



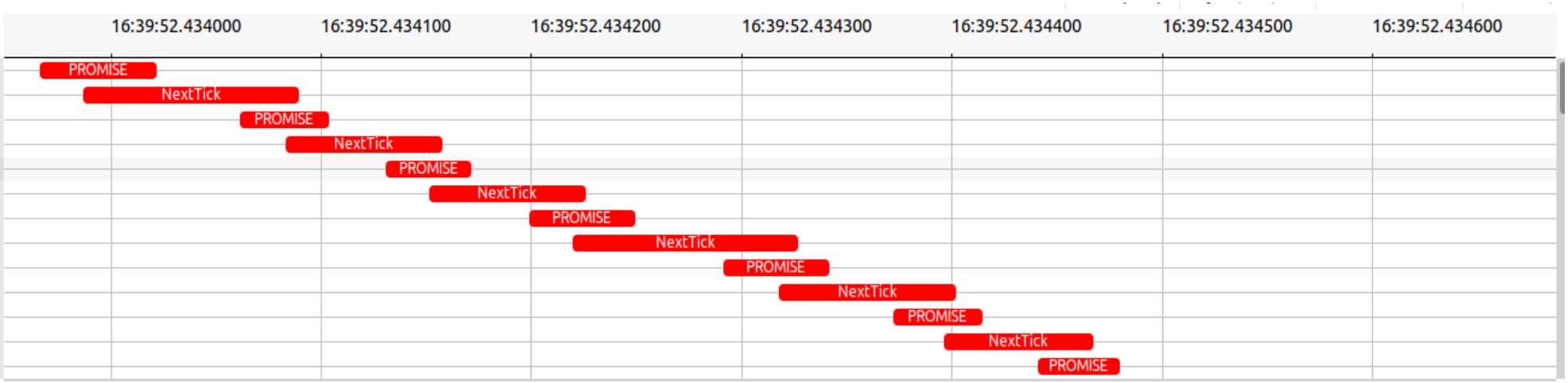
Let zoom on the view

Use case 1

Resulting flow:



Let zoom on the view

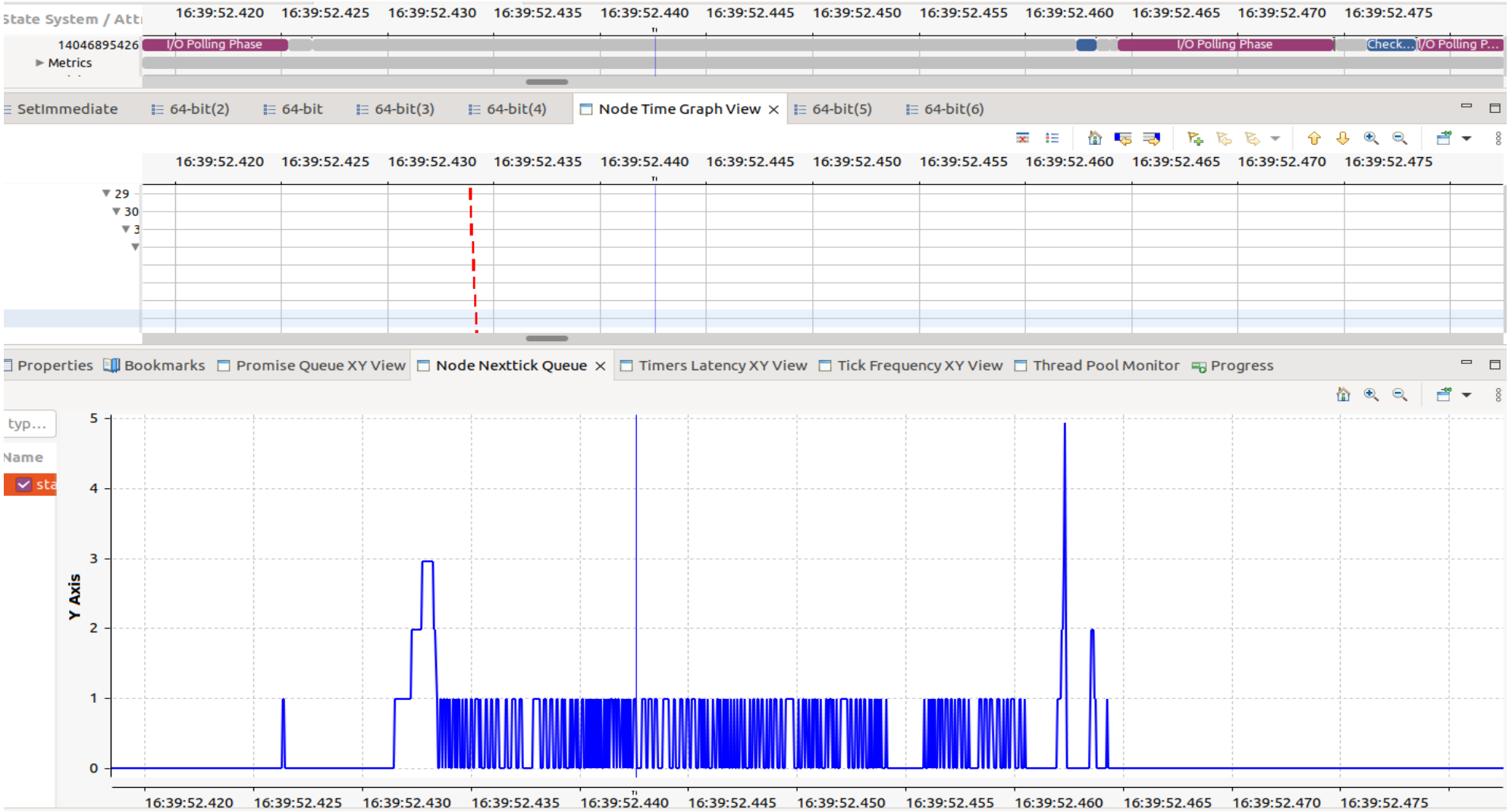


Use case 1

Results Interpretation

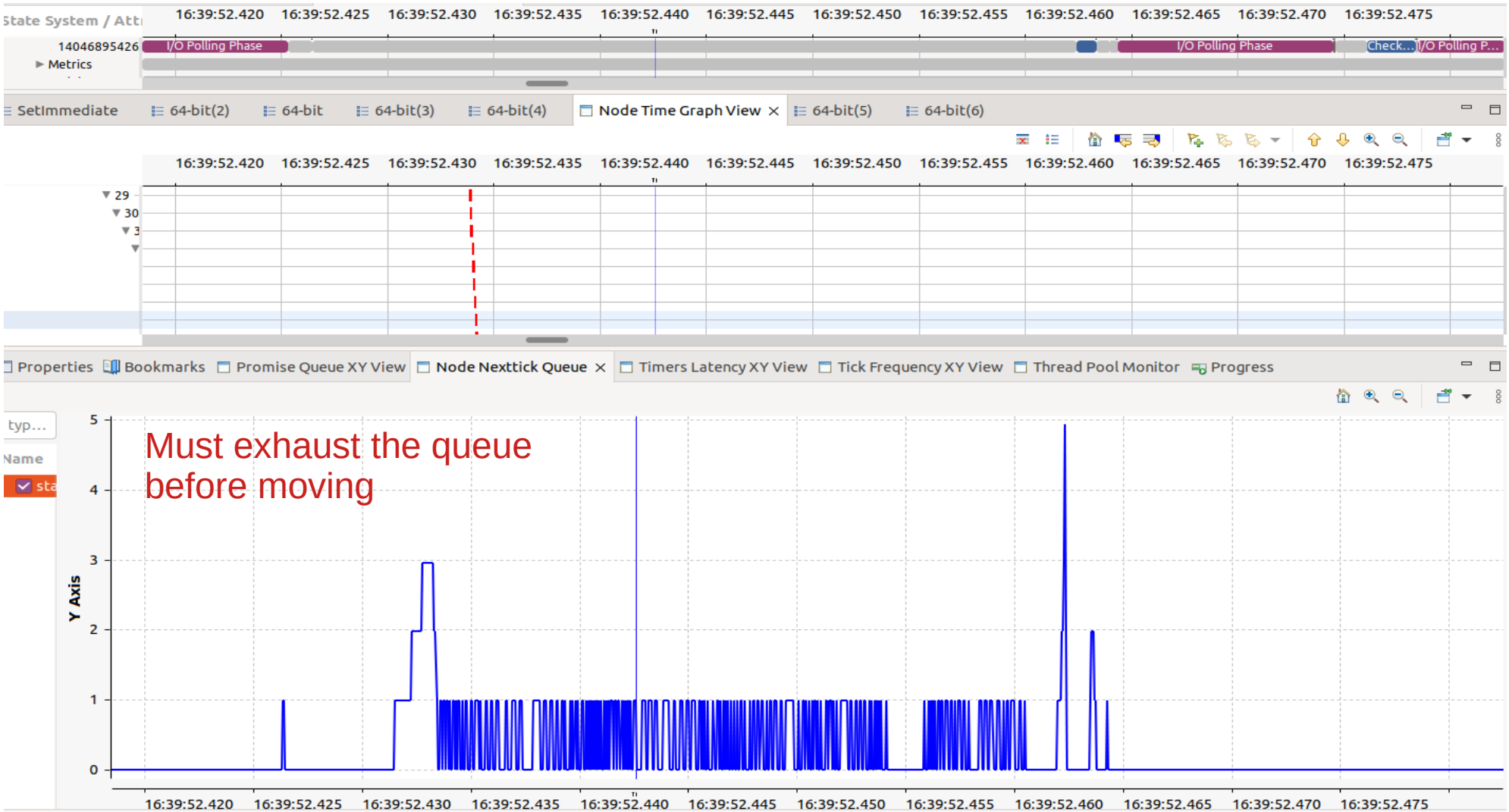
Use case 1

Results Interpretation



Use case 1

Results Interpretation

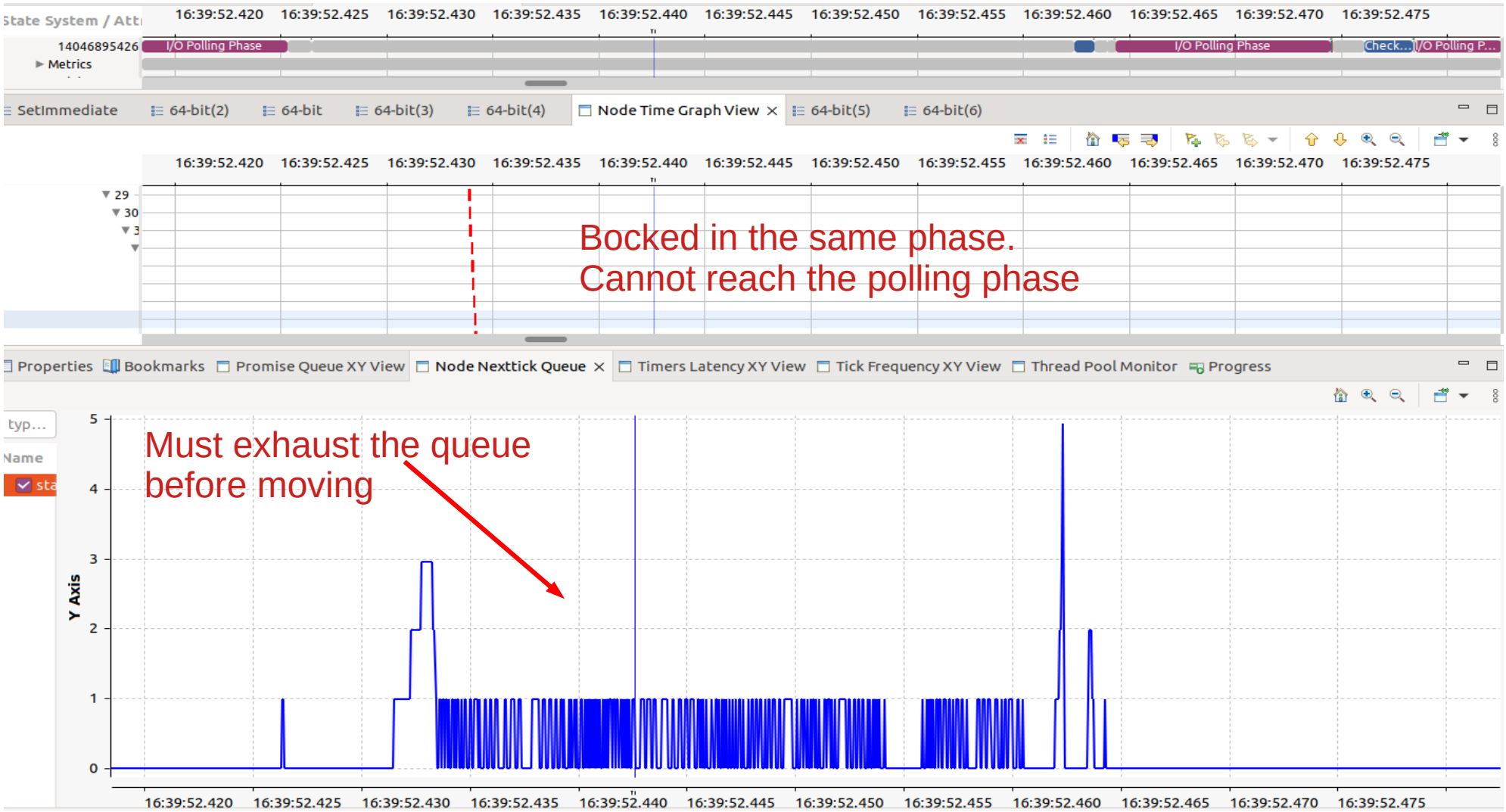


Use case 1

Results Interpretation

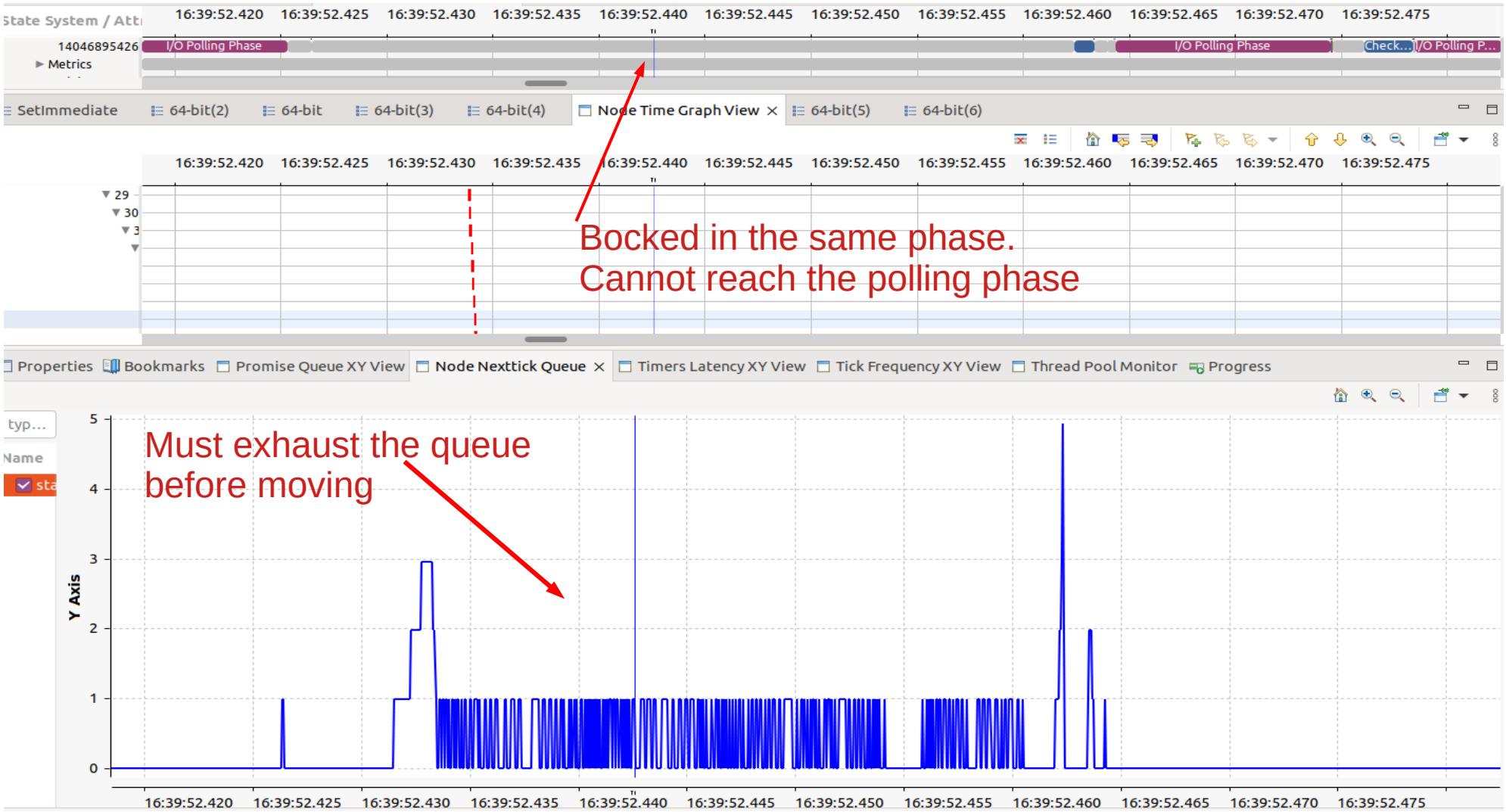
Use case 1

Results Interpretation



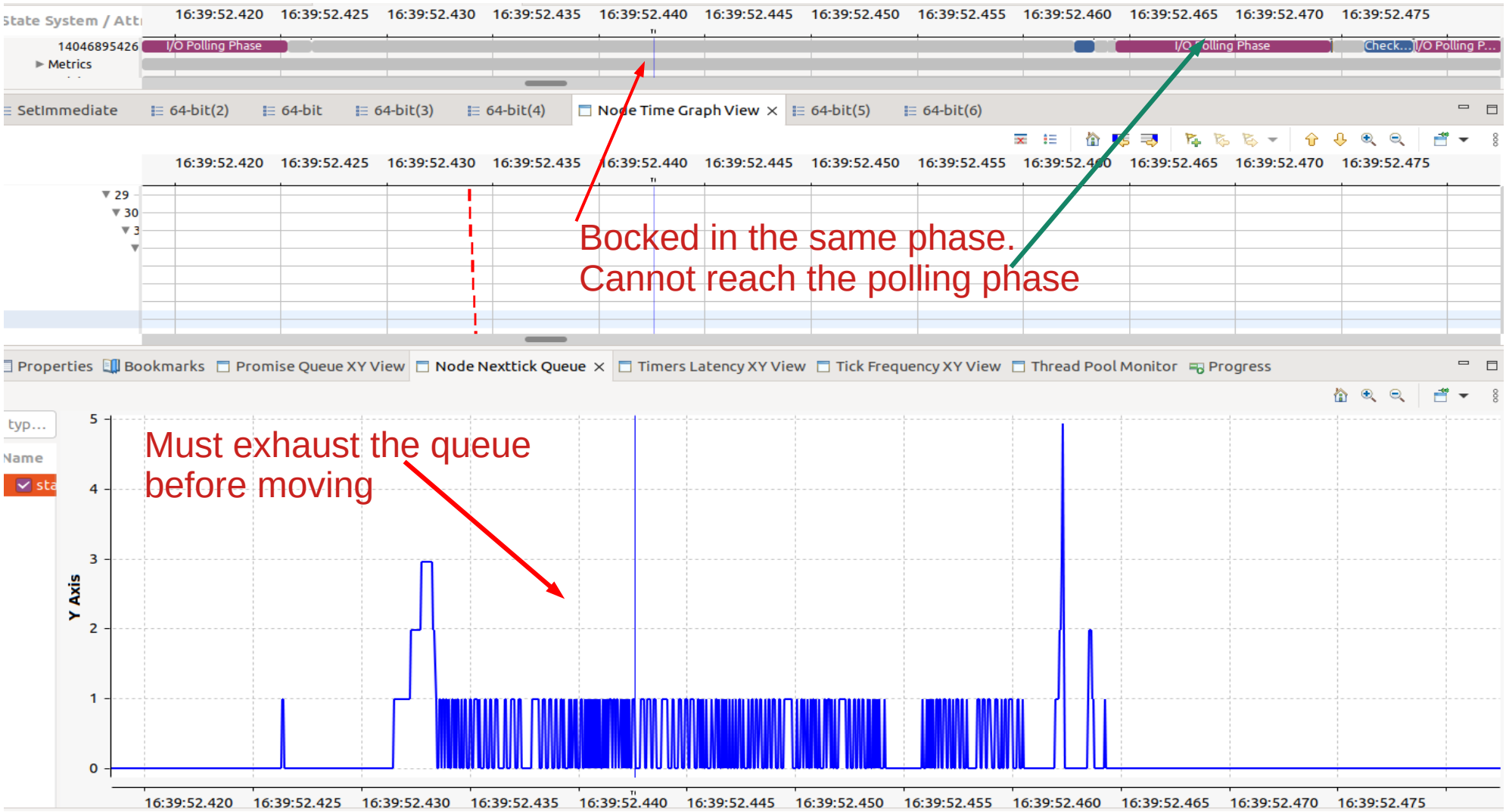
Use case 1

Results Interpretation



Use case 1

Results Interpretation



Use case 2

Asynchronous resources destruction bug

Use case 2

Asynchronous resources destruction bug

Each asynchronous resource has a lifetime

Use case 2

Asynchronous resources destruction bug

Each asynchronous resource has a lifetime

Internal mechanisms in Node.js track async. resources and destroy them after used

Use case 2

Asynchronous resources destruction bug

Each asynchronous resource has a lifetime

Internal mechanisms in Node.js track async. resources and destroy them after used

“Destroy” callback is invoked when cleaning the system from the resource

Test scenario

Use case 2

Asynchronous resources destruction bug

Each asynchronous resource has a lifetime

Internal mechanisms in Node.js track async. resources and destroy them after used

“Destroy” callback is invoked when cleaning the system from the resource

Test scenario

An express server contacted at the route /test

Use case 2

Asynchronous resources destruction bug

Each asynchronous resource has a lifetime

Internal mechanisms in Node.js track async. resources and destroy them after used

“Destroy” callback is invoked when cleaning the system from the resource

Test scenario

An express server contacted at the route /test

A client ***A*** contacting continuously the server on the route /test

Use case 2

Asynchronous resources destruction bug

Each asynchronous resource has a lifetime

Internal mechanisms in Node.js track async. resources and destroy them after used

“Destroy” callback is invoked when cleaning the system from the resource

Test scenario

An express server contacted at the route /test

A client ***A*** contacting continuously the server on the route /test

When contacted by A, the server calls the ***fs.promise.readFile()*** of the file system API

Use case 2

Resulting Flow

Use case 2

Resulting Flow

Resources | Control Flow | Statistics | State System Explorer

State Syst

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

14 — Checking Phase

Met

SetImmediate | 64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5)

CPU	Event type	Contents
<srch>	<srch>	id=9544
3	uv_provider:uv_send_event	id=7923, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7906, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7905, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7912, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_

Properties | Bookmarks | Node Time Graph Vie | Promise Queue XY Vie | Node Nexttick Queue | Timers Latency XY Vie | Tick Frequency XY Vie | Thread Pool Monitor | Progress

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

PROMISE

PROMISE

PR...

fs_read

Trace	9545
Date	2022-05-04
Start Time	16:47:40.694818529
Stop Time	16:47:40.699026325
Duration	0.004207796s (4,2 ms)

Use case 2

Resulting Flow

Resources | Control Flow | Statistics | State System Explorer

State Syst

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

14 — Checking Phase

SetImmediate | 64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5)

CPU	Event type	Contents
<srch>	<srch>	id=9544
3	uv_provider:uv_send_event	id=7923, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7906, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7905, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7912, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_

Properties | Bookmarks | Node Time Graph Vie | Promise Queue XY Vie | Node Nexttick Queue | Timers Latency XY Vie | Tick Frequency XY Vie | Thread Pool Monitor | Progress

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

PROMISE

PROMISE

PR...

fs_read

Trace	9545
Date	2022-05-04
Start Time	16:47:40.694818529
Stop Time	16:47:40.699026325
Duration	0.004207796s (4,2 ms)

Atomic read operation

Use case 2

Resulting Flow

Resources | Control Flow | Statistics | State System Explorer

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

Checking Phase

SetImmediate | 64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5)

CPU	Event type	Contents
<srch>	<srch>	id=9544
3	uv_provider:uv_send_event	id=7923, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7906, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7905, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7912, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_

Properties | Bookmarks | Node Time Graph Vie | Promise Queue XY Vie | Node Nexttick Queue | Timers Latency XY Vie | Tick Frequency XY Vie | Thread Pool Monitor | Progress

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

PROMISE

PROMISE

PR...

fs_read

Trace	9545
Date	2022-05-04
Start Time	16:47:40.694818529
Stop Time	16:47:40.699026325
Duration	0.004207796s (4,2 ms)

Atomic read operation

Use case 2

Results interpretation

Use case 2

Results interpretation

Resources | Control Flow | Statistics | State System Explorer

State Syst

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

14 — Checking Phase

Met

SetImmediate | 64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5)

CPU	Event type	Contents
<srch>	<srch>	id=9544
3	uv_provider:uv_send_event	id=7923, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7906, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7905, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7912, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_

Properties | Bookmarks | Node Time Graph Vie | Promise Queue XY Vie | Node Nexttick Queue | Timers Latency XY Vie | Tick Frequency XY Vie | Thread Pool Monitor | Progress

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

PROMISE

PROMISE

PR...

fs_read

Trace	9545
Date	2022-05-04
Start Time	16:47:40.694818529
Stop Time	16:47:40.699026325
Duration	0.004207796s (4,2 ms)

Use case 2

Results interpretation

Resources | Control Flow | Statistics | State System Explorer

State Syst

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

14 — Checking Phase

Met

SetImmediate | 64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5)

CPU	Event type	Contents
<srch>	<srch>	id=9544
3	uv_provider:uv_send_event	id=7923, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7906, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7905, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7912, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_

Properties | Bookmarks | Node Time Graph Vie | Promise Queue XY Vie | Node Nexttick Queue | Timers Latency XY Vie | Tick Frequency XY Vie | Thread Pool Monitor | Progress

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

PROMISE

PROMISE

PR...

fs_read

Trace	9545
Date	2022-05-04
Start Time	16:47:40.694818529
Stop Time	16:47:40.699026325
Duration	0.004207796s (4,2 ms)

Abnormal behavior

Use case 2

Results interpretation

Resources | Control Flow | Statistics | State System Explorer

State Syst

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

14 — Checking Phase

Met

SetImmediate | 64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5)

CPU	Event type	Contents
<srch>	<srch>	id=9544
3	uv_provider:uv_send_event	id=7923, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7906, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7905, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7912, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_

Properties | Bookmarks | Node Time Graph Vie | Promise Queue XY Vie | Node Nexttick Queue | Timers Latency XY Vie | Tick Frequency XY Vie | Thread Pool Monitor | Progress

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

PROMISE

PROMISE

PR...

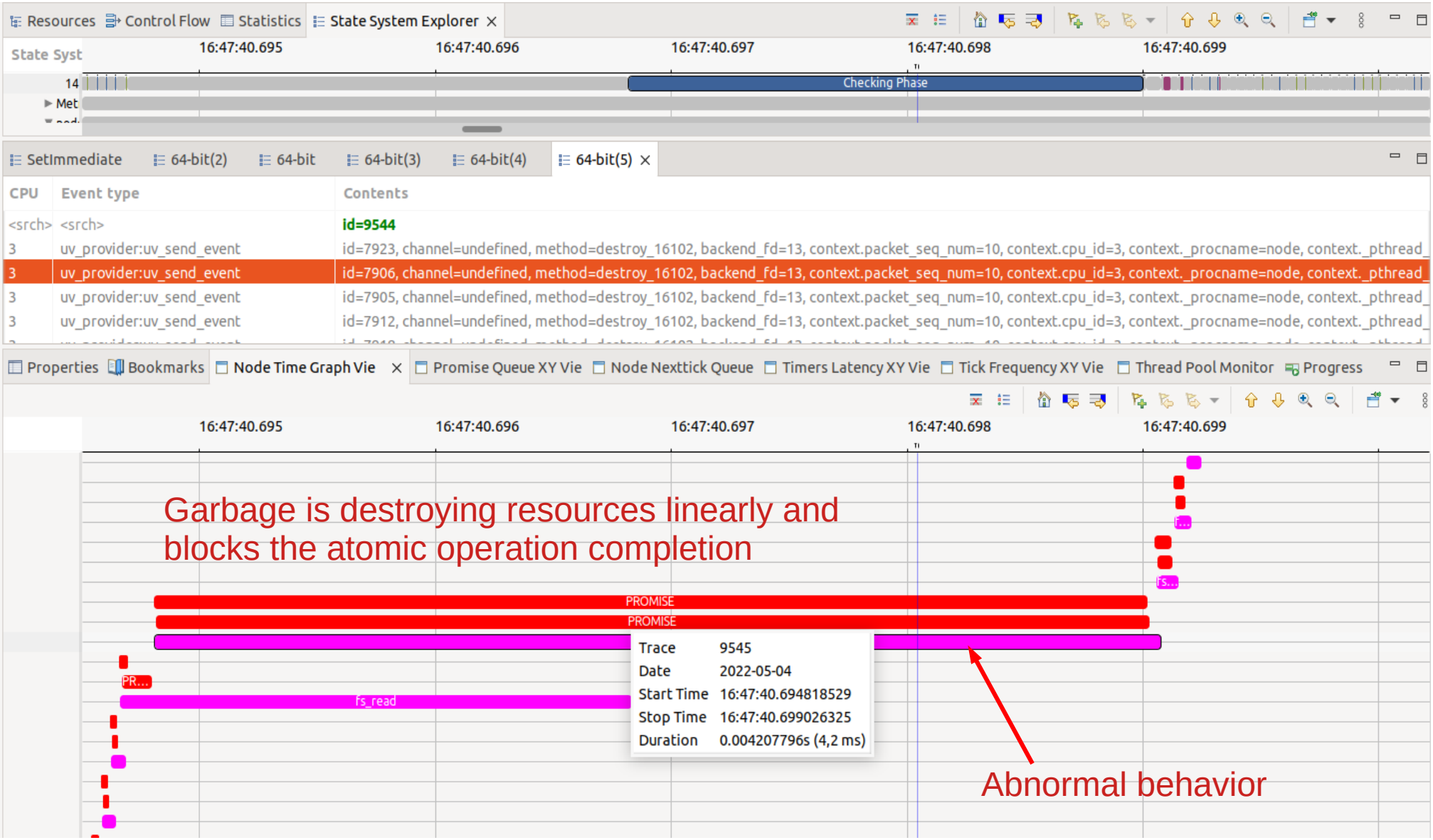
fs_read

Trace	9545
Date	2022-05-04
Start Time	16:47:40.694818529
Stop Time	16:47:40.699026325
Duration	0.004207796s (4,2 ms)

Abnormal behavior

Use case 2

Results interpretation



Use case 2

Results interpretation

Resources | Control Flow | Statistics | State System Explorer

State Syst

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

14 — Checking Phase

Met

SetImmediate | 64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5)

CPU	Event type	Contents
<srch>	<srch>	id=9544
3	uv_provider:uv_send_event	id=7923, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7906, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7905, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_
3	uv_provider:uv_send_event	id=7912, channel=undefined, method=destroy_16102, backend_fd=13, context.packet_seq_num=10, context.cpu_id=3, context._procname=node, context._pthread_

Properties | Bookmarks | Node Time Graph Vie | Promise Queue XY Vie | Node Nexttick Queue | Timers Latency XY Vie | Tick Frequency XY Vie | Thread Pool Monitor | Progress

16:47:40.695 16:47:40.696 16:47:40.697 16:47:40.698 16:47:40.699

Garbage is destroying resources linearly and blocks the atomic operation completion

PROMISE

PROMISE

PR...

fs_read

Trace	9545
Date	2022-05-04
Start Time	16:47:40.694818529
Stop Time	16:47:40.699026325
Duration	0.004207796s (4,2 ms)

Abnormal behavior

Use case 3

Race conditions detection

Use case 3

Race conditions detection

Atomic operation may access the same resource

Use case 3

Race conditions detection

Atomic operation may access the same resource

Two or more processes may access a shared resource

Use case 3

Race conditions detection

Atomic operation may access the same resource

Two or more processes may access a shared resource

The resource may be altered, locked, deleted or modified before the other accesses it

Test scenario

Use case 3

Race conditions detection

Atomic operation may access the same resource

Two or more processes may access a shared resource

The resource may be altered, locked, deleted or modified before the other accesses it

Test scenario

An express server contacted at the route /race

Use case 3

Race conditions detection

Atomic operation may access the same resource

Two or more processes may access a shared resource

The resource may be altered, locked, deleted or modified before the other accesses it

Test scenario

An express server contacted at the route /race

A client ***A*** contacting continuously the server on the route /race

Use case 3

Race conditions detection

Atomic operation may access the same resource

Two or more processes may access a shared resource

The resource may be altered, locked, deleted or modified before the other accesses it

Test scenario

An express server contacted at the route /race

A client ***A*** contacting continuously the server on the route /race

When contacted, the server calls a function that creates the file “test.txt”, then calls a function to read the file, and call another function to delete the file

Use case 3

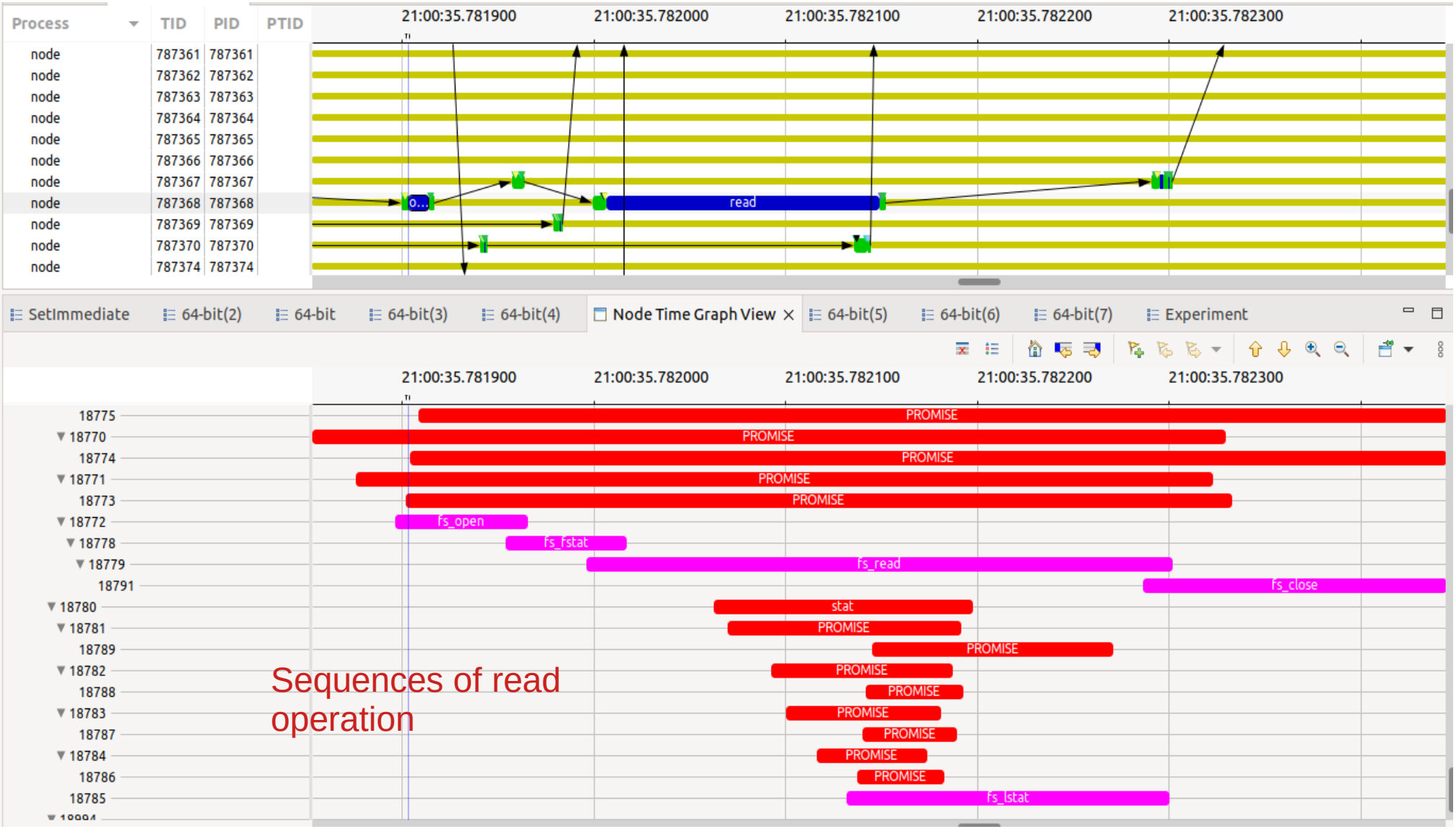
Resulting flow

Use case 3

Resulting flow

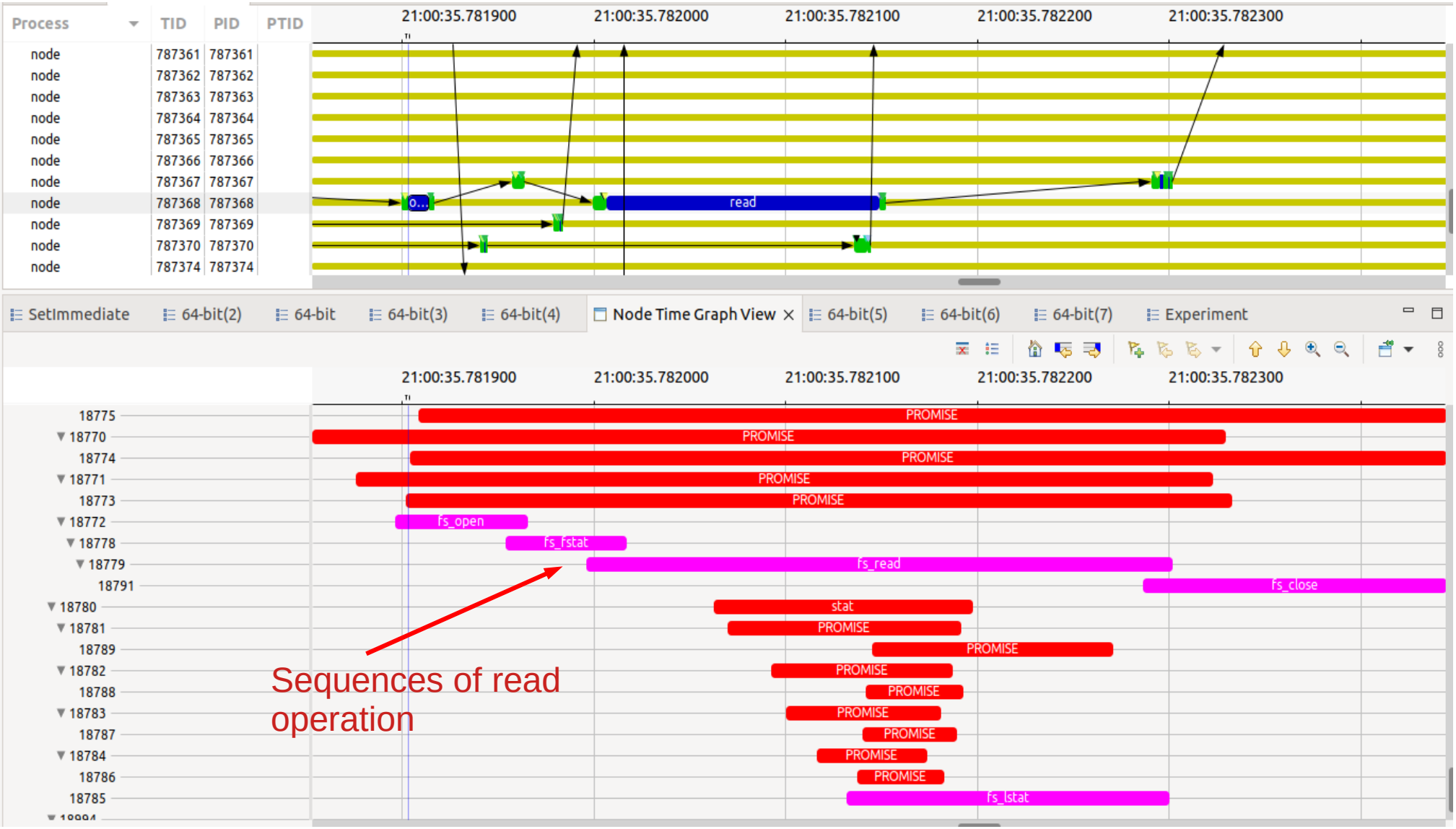
Use case 3

Resulting flow



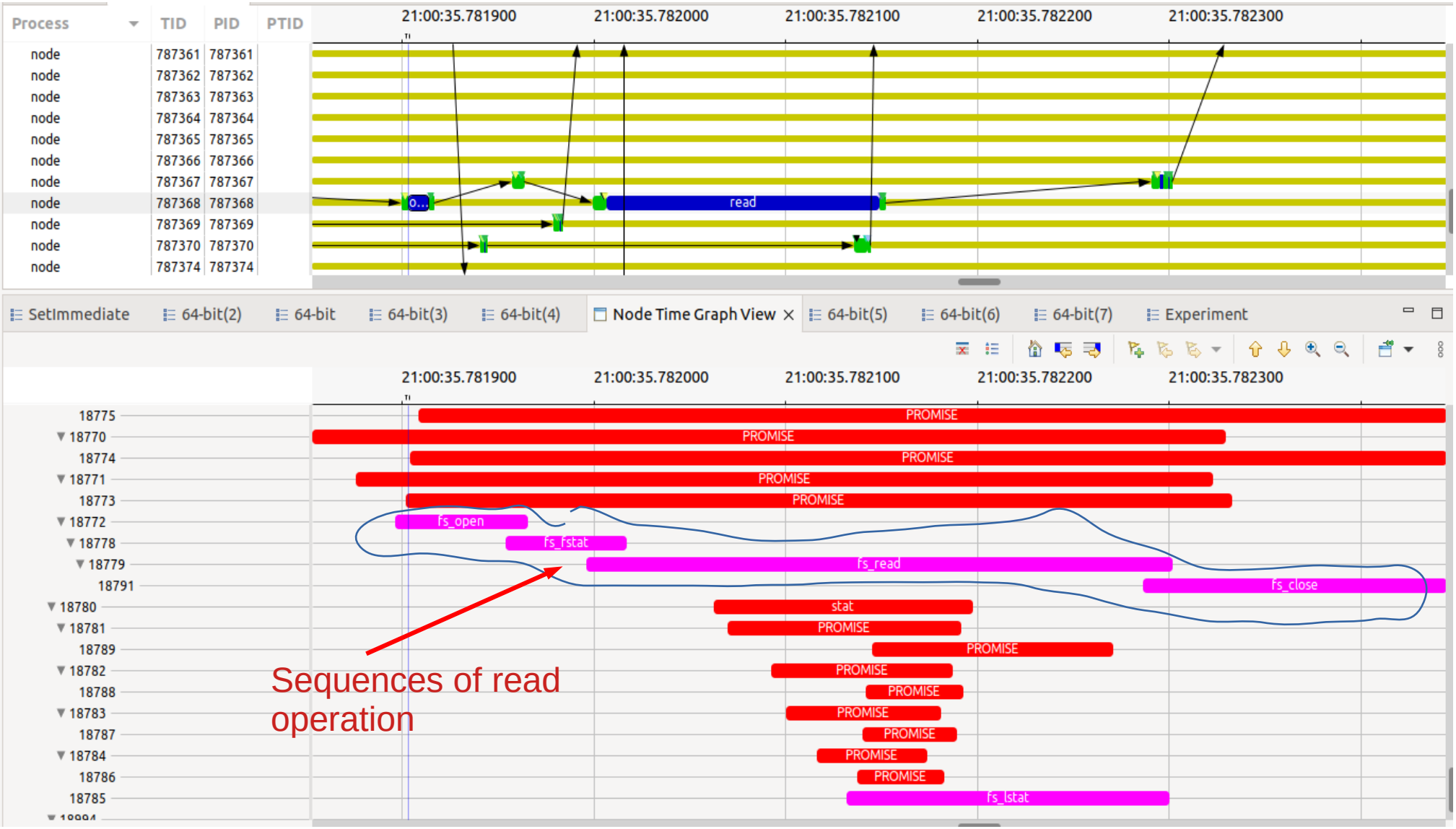
Use case 3

Resulting flow



Use case 3

Resulting flow



Use case 3

Results interpretation

Use case 3

Results interpretation

State System Explorer | Node Time Graph View

20:20:08.180000 | 20:20:08.180500 | 20:20:08.181000 | 20:20:08.181500

fs_write
fs_close
PROMISE
fs_open
fs_unlink

Trace	91
Date	2022-05-12
Start Time	20:20:08.179950359
Stop Time	20:20:08.181058121
Duration	0.001107762s (1,1 ms)

it(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5) | 64-bit(6) | 64-bit(7) | 64-bit(8) | Experiment | 64-bit(13)

hannel	CPU	Event type	Contents
srch>	<srch>	<srch>	<srch>
hannel0_3	3	uv_provider:uv_send_event	id=87, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.21200713404852, channel=85, method=js_open_main, backend_fd=12, context.packet_seq_num=0, contex
hannel0_3	3	uv_provider:uv_send_event	id=88, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.480561237982693, channel=6.21200713404852, method=js_open_create, backend_fd=14, context.packet_
hannel0_3	3	uv_provider:uv_send_event	id=89, channel=85, method=FSREQCALLBACK.29685, backend_fd=19, context.packet_seq_num=0, context.cpu_
hannel0_3	3	uv_provider:uv_fs_open_event	fd=29685, path=0xffffffffbe1eed68, backend_fd=2032454154, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_async_file_event	id=1649709016, backend_fd=29685, oper=fs_open, context.packet_seq_num=0, context.cpu_id=3, context._pro
hannel0_3	3	uv_provider:uv_work_submit_event	add_loop=0xffffffffbe1eed08, add_work=0xffffffffbe1eecf8, add_w=0xffffffffbe1eed00, add_done=0xffffffffb

s | Promise Queue XY View | Timers Latency XY View | Tick Frequency XY View | Thread Pool Monitor | Progress | Critical Flow View

Use case 3

Results interpretation

State System Explorer | Node Time Graph View

20:20:08.180000 | 20:20:08.180500 | 20:20:08.181000 | 20:20:08.181500

fs_write
fs_close
PROMISE
fs_open
fs_unlink

Trace	91
Date	2022-05-12
Start Time	20:20:08.179950359
Stop Time	20:20:08.181058121
Duration	0.001107762s (1,1 ms)

Race condition

it(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5) | 64-bit(6) | 64-bit(7) | 64-bit(8) | Experiment | 64-bit(13)

hannel	CPU	Event type	Contents
srch>	<srch>	<srch>	<srch>
hannel0_3	3	uv_provider:uv_send_event	id=87, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.21200713404852, channel=85, method=js_open_main, backend_fd=12, context.packet_seq_num=0, contex
hannel0_3	3	uv_provider:uv_send_event	id=88, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.480561237982693, channel=6.21200713404852, method=js_open_create, backend_fd=14, context.packet_
hannel0_3	3	uv_provider:uv_send_event	id=89, channel=85, method=FSREQCALLBACK.29685, backend_fd=19, context.packet_seq_num=0, context.cpu_
hannel0_3	3	uv_provider:uv_fs_open_event	fd=29685, path=0xffffffffbe1eed68, backend_fd=2032454154, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_async_file_event	id=1649709016, backend_fd=29685, oper=fs_open, context.packet_seq_num=0, context.cpu_id=3, context._pro
hannel0_3	3	uv_provider:uv_work_submit_event	add_loop=0xffffffffbe1eed08, add_work=0xffffffffbe1eecf8, add_w=0xffffffffbe1eed00, add_done=0xffffffffb

s | Promise Queue XY View | Timers Latency XY View | Tick Frequency XY View | Thread Pool Monitor | Progress | Critical Flow View

Use case 3

Results interpretation

State System Explorer | Node Time Graph View

20:20:08.180000 | 20:20:08.180500 | 20:20:08.181000 | 20:20:08.181500

fs_write
fs_close
PROMISE
fs_open
fs_unlink

Trace	91
Date	2022-05-12
Start Time	20:20:08.179950359
Stop Time	20:20:08.181058121
Duration	0.001107762s (1,1 ms)

Race condition

64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5) | 64-bit(6) | 64-bit(7) | 64-bit(8) | Experiment | 64-bit(13)

Channel	CPU	Event type	Contents
<srch>	<srch>	<srch>	<srch>
hannel0_3	3	uv_provider:uv_send_event	id=87, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.21200713404852, channel=85, method=js_open_main, backend_fd=12, context.packet_seq_num=0, contex
hannel0_3	3	uv_provider:uv_send_event	id=88, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.480561237982693, channel=6.21200713404852, method=js_open_create, backend_fd=14, context.packet_
hannel0_3	3	uv_provider:uv_send_event	id=89, channel=85, method=FSREQCALLBACK.29685, backend_fd=19, context.packet_seq_num=0, context.cpu_
hannel0_3	3	uv_provider:uv_fs_open_event	fd=29685, path=0xffffffffbe1eed68, backend_fd=2032454154, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_async_file_event	id=1649709016, backend_fd=29685, oper=fs_open, context.packet_seq_num=0, context.cpu_id=3, context._pro
hannel0_3	3	uv_provider:uv_work_submit_event	add_loop=0xffffffffbe1eed08, add_work=0xffffffffbe1eecf8, add_w=0xffffffffbe1eed00, add_done=0xffffffffb

Promise Queue XY View | Timers Latency XY View | Tick Frequency XY View | Thread Pool Monitor | Progress | Critical Flow View

Use case 3

Results interpretation

State System Explorer | Node Time Graph View

20:20:08.180000 | 20:20:08.180500 | 20:20:08.181000 | 20:20:08.181500

fs_write
fs_close
PROMISE
fs_open
fs_unlink

Trace	91
Date	2022-05-12
Start Time	20:20:08.179950359
Stop Time	20:20:08.181058121
Duration	0.001107762s (1,1 ms)

Race condition

Opening a file in reading mode while the file was already deleted

64-bit(2) | 64-bit | 64-bit(3) | 64-bit(4) | 64-bit(5) | 64-bit(6) | 64-bit(7) | 64-bit(8) | Experiment | 64-bit(13)

Channel	CPU	Event type	Contents
<srch>	<srch>	<srch>	<srch>
hannel0_3	3	uv_provider:uv_send_event	id=87, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.21200713404852, channel=85, method=js_open_main, backend_fd=12, context.packet_seq_num=0, contex
hannel0_3	3	uv_provider:uv_send_event	id=88, channel=85, method=PROMISE.29685, backend_fd=13, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_send_event	id=6.480561237982693, channel=6.21200713404852, method=js_open_create, backend_fd=14, context.packet_
hannel0_3	3	uv_provider:uv_send_event	id=89, channel=85, method=FSREQCALLBACK.29685, backend_fd=19, context.packet_seq_num=0, context.cpu_
hannel0_3	3	uv_provider:uv_fs_open_event	fd=29685, path=0xffffffffbe1eed68, backend_fd=2032454154, context.packet_seq_num=0, context.cpu_id=3, co
hannel0_3	3	uv_provider:uv_async_file_event	id=1649709016, backend_fd=29685, oper=fs_open, context.packet_seq_num=0, context.cpu_id=3, context._pro
hannel0_3	3	uv_provider:uv_work_submit_event	add_loop=0xffffffffbe1eed08, add_work=0xffffffffbe1eecf8, add_w=0xffffffffbe1eed00, add_done=0xffffffffb

Promise Queue XY View | Timers Latency XY View | Tick Frequency XY View | Thread Pool Monitor | Progress | Critical Flow View

Conclusion

Conclusion

- NodeCompass a powerful tool for Node.js applications performance analysis

Conclusion

- NodeCompass a powerful tool for Node.js applications performance analysis

- Brings a high level of granularity in the Performance Analysis

Conclusion

- NodeCompass a powerful tool for Node.js applications performance analysis

- Brings a high level of granularity in the Performance Analysis

- Helps in understanding the application flow

Conclusion

- NodeCompass a powerful tool for Node.js applications performance analysis

- Brings a high level of granularity in the Performance Analysis

- Helps in understanding the application flow

- Allows pinpointing bottlenecks, bugs, errors, race conditions

Conclusion

- NodeCompass a powerful tool for Node.js applications performance analysis

- Brings a high level of granularity in the Performance Analysis

- Helps in understanding the application flow

- Allows pinpointing bottlenecks, bugs, errors, race conditions

- Can be used to instrument functions in the way distributed tracers work

Bibliography

[1] I. Beschastnikh, P. Wang, Y. Brun, M. D. Ernst, Debugging distributed systems, ACM-Queue (2015).

[2] J. Hoglund, An analysis of a distributed tracing systems effect on performance. jaeger and opentracing api, UMEA University (2020).

[3] S. Tilkov, S. Vinoski, Node.js: Using javascript to build high-performance network programs, IEEE INTERNET COMPUTING (2010).

[4] Cloud desktop ide platform.

URL https://kubernetes.io/fr/

[5] Visual studio code.

URL https://code.visualstudio.com/

[6] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, A. Vahdat, Exploiting a natural network effect for scalable, fine-grained clock synchronization, 2018.

2007, pp. 171–180.

Thank you