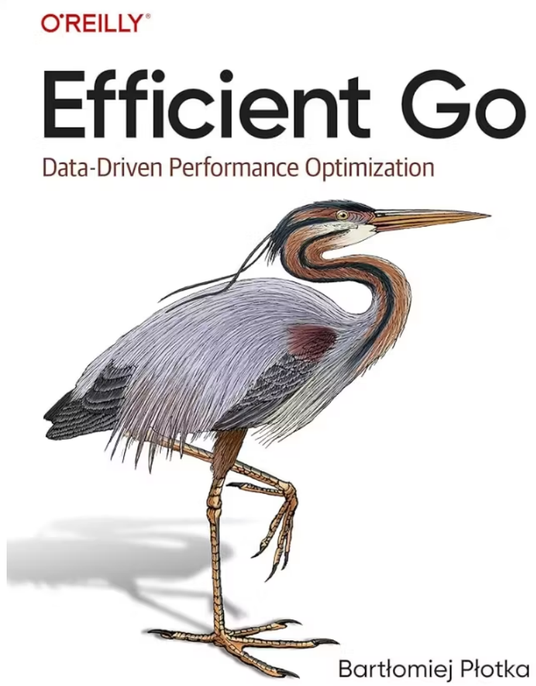
Go-ing Under the Hood: Breaking and optimizing(?) the Promtheus Parser

Hi, I’m Manik

Intern @TurboML,
Contributor @GSoC/Prometheus



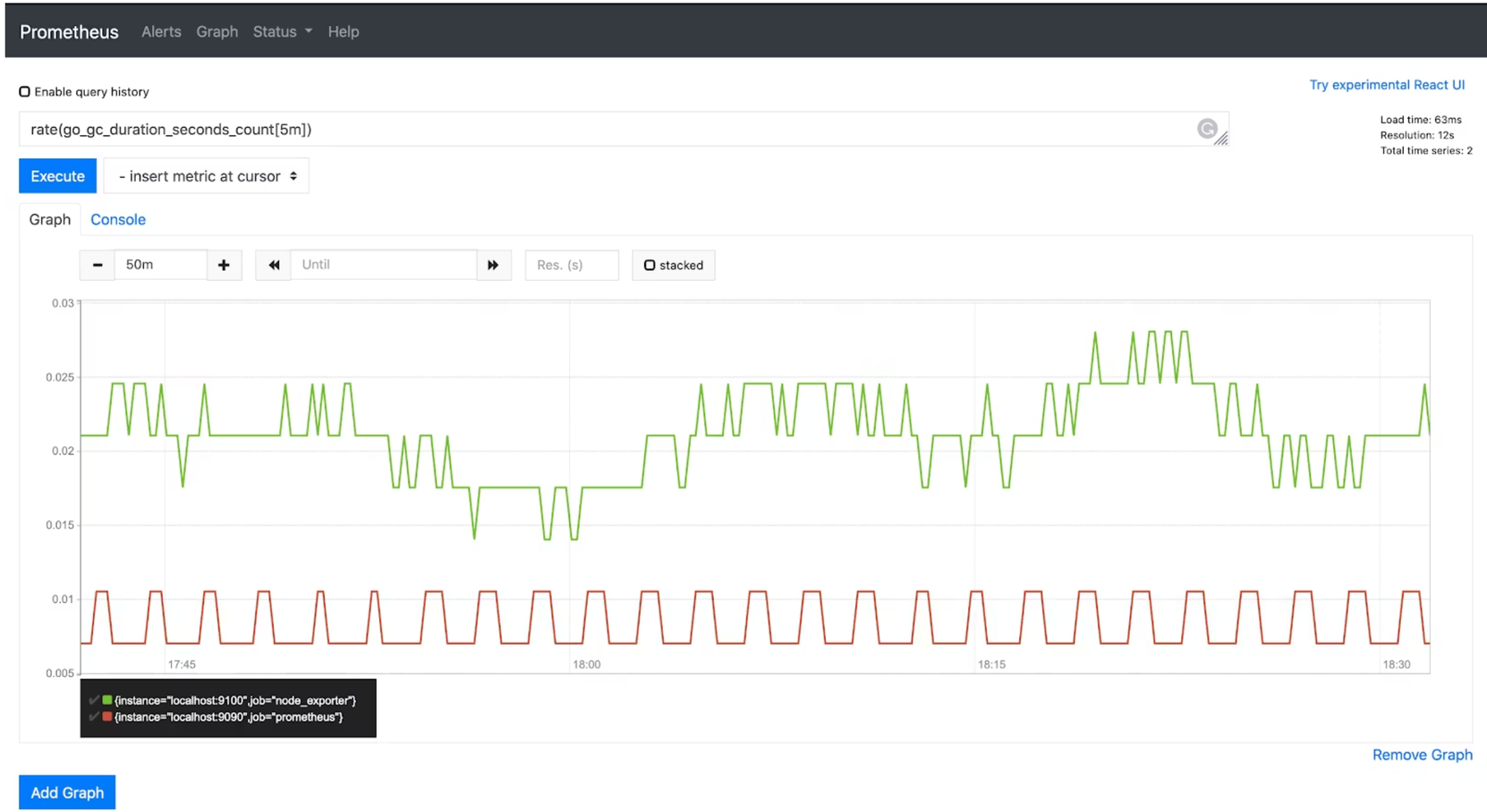
https://www.manikrana.dev/

Agenda

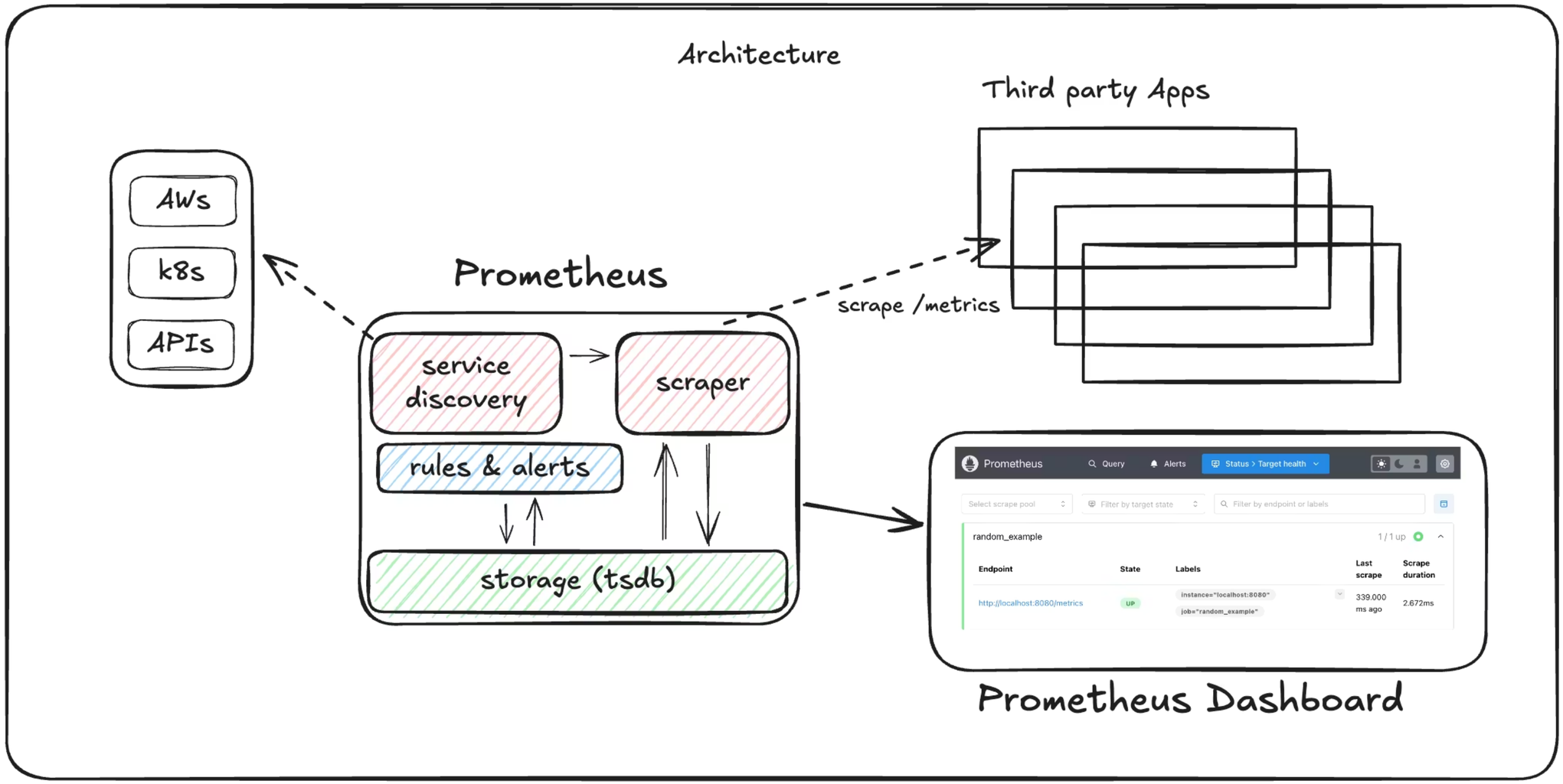
- What is Prometheus
- I’m given a task idk how to accomplish
- I ship something but its bad… really bad
- I figure out what’s causing the issue and fix
- Build your own parser

What is Prometheus?

Prometheus is a monitoring tool that collects and stores **metrics** as time series data, comprising timestamped information alongside optional key-value pairs called labels. This data is used to provide insights into system performance, enabling users to make data-driven decisions and optimize their systems.



What is Prometheus?



My task → Created Timestamp Support

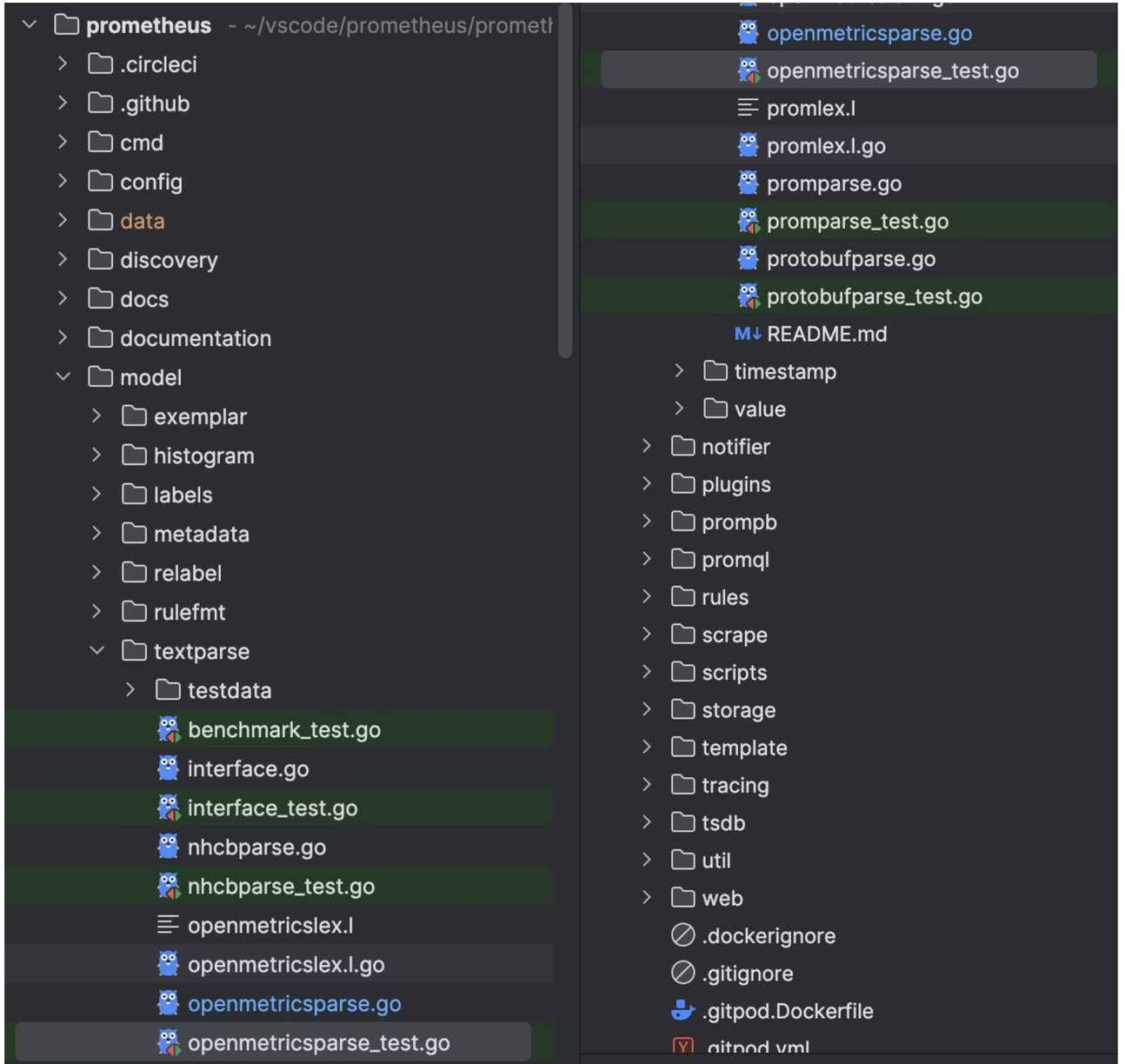
Created timestamps in Prometheus are a special type of timestamp metadata that tracks when a specific time series was first created or observed by the Prometheus system. This is different from the regular timestamps associated with individual samples in a time series.

When Prometheus first observes a new time series, it associates a "created" timestamp with that series. This timestamp marks when Prometheus first saw this particular series.

```
# HELP foo Counter with and without labels to certify CT is parsed for both cases
# TYPE foo counter
foo_total 17.0 1520879607.789 # {id="counter-test"} 5
foo_created 1520872607.123
foo_total{a="b"} 17.0 1520879607.789 # {id="counter-test"} 5
foo_created{a="b"} 1520872607.123
foo_total{le="c"} 21.0
foo_created{le="c"} 1520872621.123
foo_total{le="1"} 10.0
```

I’ve never worked with parsers what do I do?

Prior to GSoC, I never worked with parsers and I had no idea how to proceed with the task. Moreover, the Prometheus codebase was massive and it all got overwhelming until I learned this one simple trick: debugging



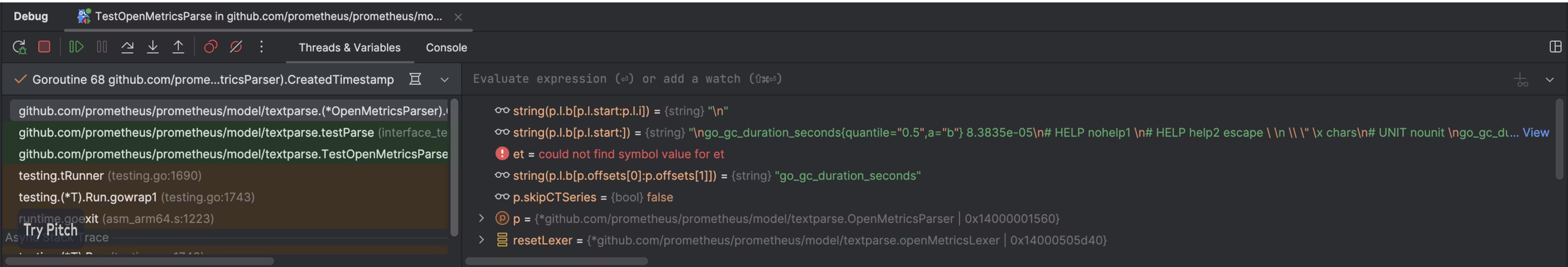
I've never worked with parsers what do I do?

I knew where I had to work but there were too many moving parts. Luckily, Prometheus has extensive testing and all I had to do was follow the breadcrumbs.

I start with adding breakpoints in 2 places:

1. the function I had to work on
2. the entry point for a unit test that would eventually call the aforementioned function

We then start the debugger and hit continue till we find where the function was called.
Based off the stacktrace we can figure out where and when the our breakpoints were reached.



Implementing Created Timestamps

KEY TECHNICAL DETAILS

• It utilizes an ephemeral deep copy of the main parser to "peek ahead" at upcoming lines without altering the original parser's state.
The function verifies if the "_created" line belongs to the same metric series by comparing labels.
• The CreatedTimestamp() function specifically targets metrics of type counter, summary, and histogram.

https://github.com/prometheus/prometheus/pull/14356

It was the process of multiple syncs and pair programming sessions

```go
func deepCopy(p *OpenMetricsParser) OpenMetricsParser {
    newB := make([]byte, len(p.l.b))
    copy(newB, p.l.b)

    newLexer := &openMetricsLexer{
        b:     newB,
        i:     p.l.i,
        start: p.l.start,
        err:   p.l.err,
        state: p.l.state,
    }

    newParser := OpenMetricsParser{
        l:             newLexer,
        builder:       p.builder,
        mtype:         p.mtype,
        val:           p.val,
        skipCTSeries: false,
    }
    return newParser
}
```

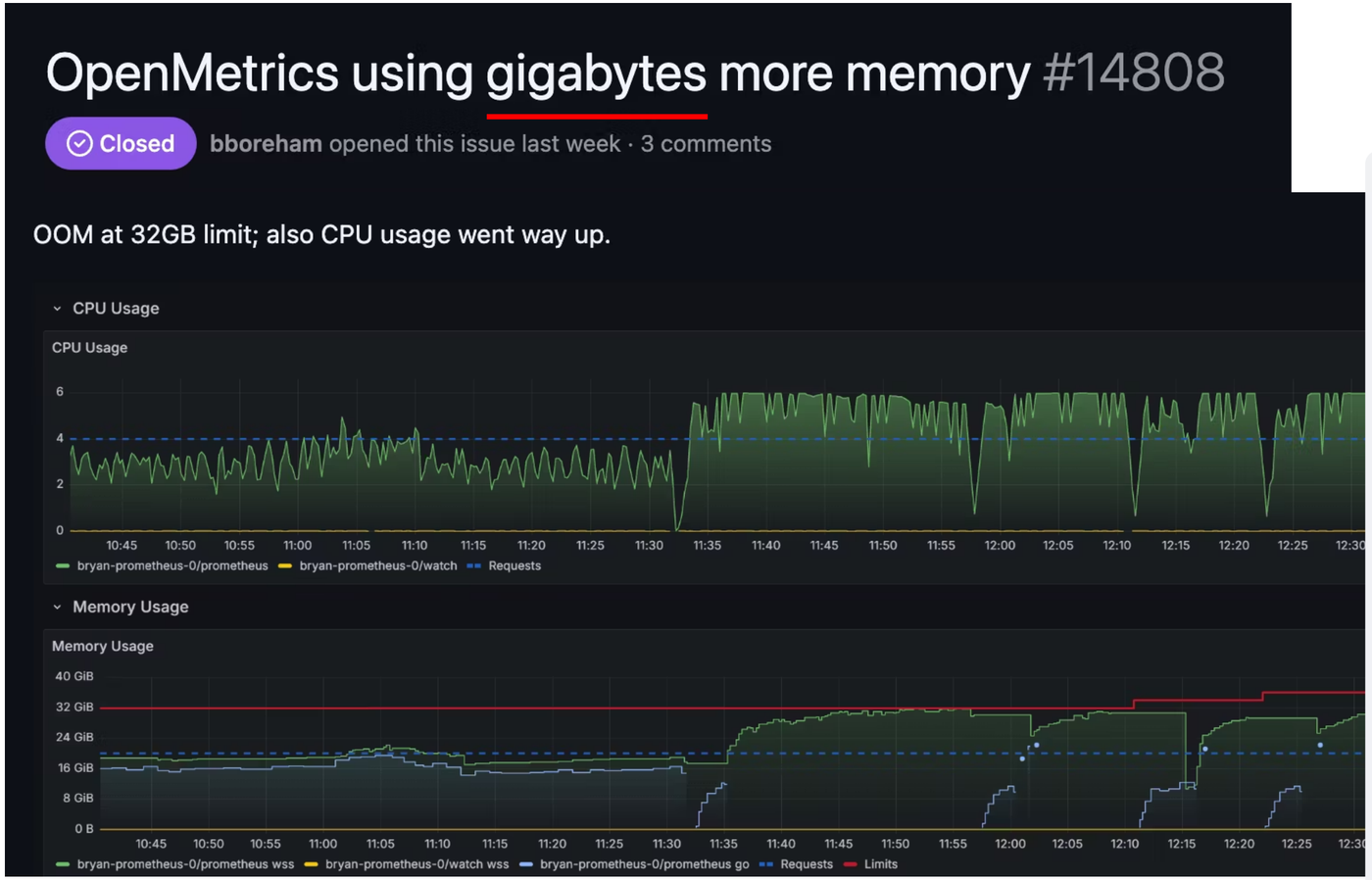
This is not ideal

```
func deepCopy(p *OpenMetricsParser) OpenMetricsParser {
    newB := make([]byte, len(p.l.b))
    copy(newB, p.l.b)

    newLexer := &openMetricsLexer{
        b:     newB,
        i:     p.l.i,
        start: p.l.start,
        err:   p.l.err,
        state: p.l.state,
    }

    newParser := OpenMetricsParser{
        l:             newLexer,
        builder:       p.builder,
        mtype:         p.mtype,
        val:           p.val,
        skipCTSeries: false,
    }
    return newParser
}
```

As we learned the hard way



https://github.com/prometheus/prometheus/issues/14808

oops 🤭

This is not ideal

```go
func deepCopy(p *OpenMetricsParser) OpenMetricsParser {
    newB := make([]byte, len(p.l.b))
    copy(newB, p.l.b)

    newLexer := &openMetricsLexer{
        b:     newB,
        i:     p.l.i,
        start: p.l.start,
        err:   p.l.err,
        state: p.l.state,
    }

    newParser := OpenMetricsParser{
        l:             newLexer,
        builder:       p.builder,
        mtype:         p.mtype,
        val:           p.val,
        skipCTSeries: false,
    }
    return newParser
}
```

Now how do we fix this? Start with benchmarks

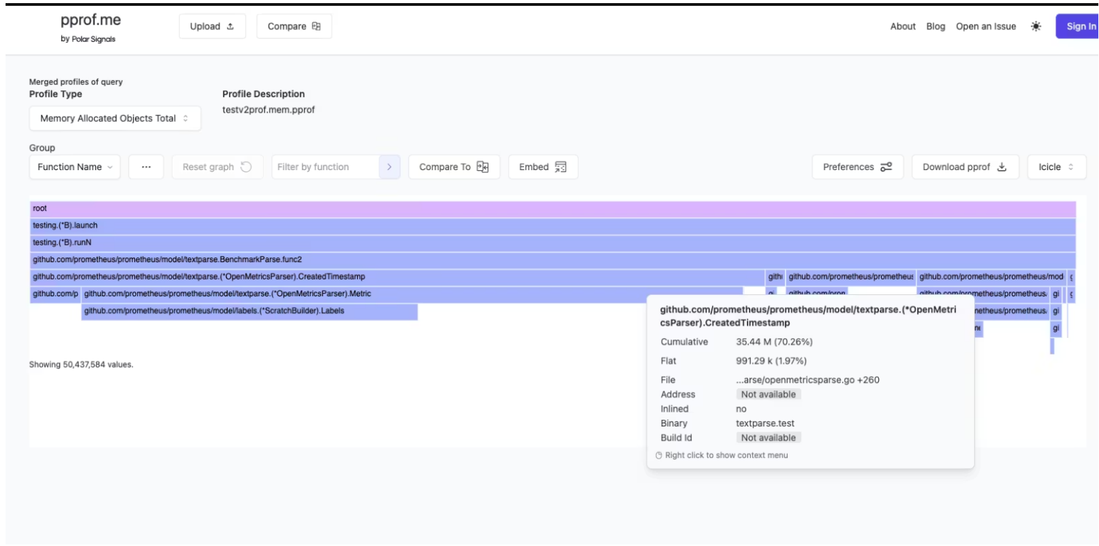
We knew something went wrong after my PR but we didn't know what exactly the issue was. For this we needed to extend the parser's micro benchmarks to include the new created timestamps syntax.

MICROBENCHMARKS

- Microbenchmarking involves measuring the performance of small, isolated code segments, such as individual functions or methods.
- This approach helps in identifying specific bottlenecks and understanding the impact of changes at a granular level.

GETTING MICROBENCHMARKS RIGHT

- Create realistic and representative microbenchmarks. Avoid synthetic benchmarks that may not accurately reflect real-world usage and lead to misleading conclusions.
- Use inputs and scenarios that closely mimic actual application behavior.
- When conducting microbenchmarks in Go, it's essential to address factors that can skew results, such as compiler optimizations and garbage collection



https://github.com/prometheus/prometheus/pull/14965
https://github.com/prometheus/prometheus/pull/15097
https://github.com/prometheus/prometheus/pull/15150

Now build it yourself

1. Define Token Types
2. Define Regex Patterns
3. Define Lexer States
4. Build Lexer Rules
5. Create Lexer Class
6. Define AST Structure
7. Define Parser Rules
8. Build Parser Class

```
class JSXLexer:
    # List of token names
    tokens = (
        "TAGSTART",  # <
        "TAGEND",  # >
        "CLOSETAG",  # </
        "SELFCLOSING",  # />
        "EQUALS",  # =
        "STRING",  # "hello" or 'hello'
        "IDENTIFIER",  # tag names, attribute names
        "TEXT",  # text content
    )


    # Tokens
    def t_ANY_error(self, t):
        t.lexer.skip(1)

    def t_INITIAL_SELFCLOSING(self, t):
        r"/>"
        return t

    def t_INITIAL_CLOSETAG(self, t):
        r"</"
        return t

    def t_INITIAL_TAGSTART(self, t):
        r"<(?!/)"
        return t
```

Thank you

https://www.manikrana.dev/

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It’s free)