GoFrameDistributed TracingGoFrameTracing

OpenTelemetry

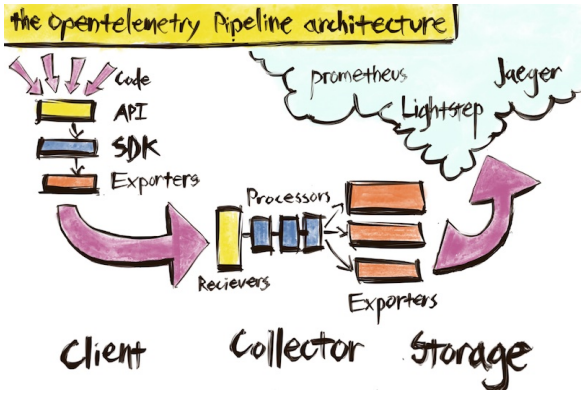
Distributed TracingGoogleTracingNetflixOpenTracingGoogleOpenCensusOpenTelemetry otel

1. OpenTracing
2. OpenTelemetry

TracingOpenTelemetryGolang

1. https://github.com/open-telemetry/opentelemetry-go
2. https://github.com/open-telemetry/opentelemetry-go-contrib

Jaeger/Prometheus/GrafanaOpenTelemetry



OpenTelemetryOpenTelemetry OpenTelemetry https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/trace/api.md

blocked URL

TracerProvider

TracerTracerProvider (NoopTracerProvider)TracerjaegerjaegerTracerProvider

Content Menu

```
        // InitJaeger initializes and registers jaeger to global
TracerProvider.
//
// The output parameter `flush` is used for waiting exported trace spans
to be uploaded,
// which is useful if your program is ending and you do not want to lose
recent spans.
func InitJaeger(serviceName, endpoint string) (flush func(), err error) {
        var endpointOption jaeger.EndpointOption
        if strings.HasPrefix(endpoint, "http") {
                // HTTP.
                endpointOption = jaeger.WithCollectorEndpoint(endpoint)
        } else {
                // UDP.
                endpointOption = jaeger.WithAgentEndpoint(endpoint)
        }
        return jaeger.InstallNewPipeline(
                endpointOption,
                jaeger.WithProcess(jaeger.Process{
                        ServiceName: serviceName,
                }),
                jaeger.WithSDK(&trace.Config{
                        DefaultSampler: trace.AlwaysSample(),
                }),
        )
}
```

Tracer

Tracertracerspan8spantracer:

```
                [Span A]  (the root span)
             |
     +------+------+
     |             |
 [Span B]      [Span C] (Span C is a `ChildOf` Span A)
     |             |
 [Span D]      +---+-------+
               |           |
           [Span E]    [Span F] >>> [Span G] >>> [Span H]



                         (Span G `FollowsFrom` Span F)
```

```
        --|-------|-------|-------|-------|-------|-------|-------|-> time

 [Span A·······························································]
   [Span B·················································]
      [Span D·····································]
    [Span C·································]
         [Span E·******]        [Span F··] [Span G··] [Span H··]
```

Tracer

```
        otel.Tracer(tracerName)
//
otel.GetTracerProvider().Tracer(tracerName)
//
gtrace.NewTracer(tracerName)
```

Span

Spanspanhttpspan:

- operation name
-
- K/VTags
- K/VLogs
- SpanContext

SpanSpan

```
        otel.Tracer().Start(ctx, spanName, opts ...)
//
otel.Tracer(tracerName).Start(ctx, spanName, opts ...)
//
gtrace.NewSpan(ctx, spanName, opts...)
```

Attributes

AttributesK/V http.method="GET",http.status_code=200keyvalue spanAttributes SpanContextspan Attributes

```
        span.SetAttributes(
        label.String("http.remote", conn.RemoteAddr().String()),
        label.String("http.local", conn.LocalAddr().String()),
)
```

Events

EventsAttributesK/VAttributesEventsEventsEventsEventskeyvalue

```
        span.AddEvent("http.request", trace.WithAttributes(
        label.Any("http.request.header", headers),
        label.Any("http.request.baggage", gtrace.GetBaggageMap(ctx)),
        label.String("http.request.body", bodyContent),
))
```

SpanContext

SpanContext

- spanspan_id, trace_id
- Baggage - K/VBaggageAttributesK/VAttributes
    - keyvalue
    - BaggagespanSpanContextspanBaggage - spanK,VCPU

Propagator

Propagator/ClientServerTraceIdSpanIdBaggagePropagator/OpenTelemetryTextMapPropagatorTextMapPropagator https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/context/api-propagators.md

GoFramegtraceOpenTelemetry

```
        // defaultTextMapPropagator is the default propagator for context
propagation between peers.
defaultTextMapPropagator = propagation.NewCompositeTextMapPropagator(
        propagation.TraceContext{},
        propagation.Baggage{},
)
```

TracingGoFrameOpenTelemetryGo APIOpenTelemetryGo APITracerProviderGoFrameTracingTracingTracingTracingHTTP/gRPC

Http Client

HTTPTracing

HTTP

```
        // MiddlewareClientTracing is a client middleware that enables
tracing feature using standards of OpenTelemetry.
func MiddlewareClientTracing(c *Client, r *http.Request) (*ClientResponse,
error)
```

Use

```
        client := g.Client().Use(ghttp.MiddlewareClientTracing)
```

HTTP ClientTracing

Http Server

HTTP/Tracing

```
        // MiddlewareServerTracing is a serer middleware that enables
tracing feature using standards of OpenTelemetry.
func MiddlewareServerTracing(r *Request)
```

Use

```
        s := g.Server()
s.Group("/", func(group *ghttp.RouterGroup) {
    group.Middleware(ghttp.MiddlewareServerTracing)
    // ...
})
```

HTTP ServerTracing

gRPC Client

gRPCUnaryStreamKatyushagRPCTracing

```
        // UnaryTracing is an unary interceptor for adding tracing feature
for gRPC client using OpenTelemetry.
func (c *krpcClient) UnaryTracing(ctx context.Context, method string, req,
reply interface{}, cc *grpc.ClientConn, invoker grpc.UnaryInvoker, opts ...
grpc.CallOption) error

// StreamTracing is a stream interceptor for adding tracing feature for
gRPC client using OpenTelemetry.
func (c *krpcClient) StreamTracing(ctx context.Context, desc *grpc.
StreamDesc, cc *grpc.ClientConn, method string, streamer grpc.
Streamer,        callOpts ...grpc.CallOption) (grpc.ClientStream, error)
```

```
        grpcClientOptions := make([]grpc.DialOption, 0)
grpcClientOptions = append(
        grpcClientOptions,
        grpc.WithInsecure(),
        grpc.WithBlock(),
        grpc.WithChainUnaryInterceptor(
                krpc.Client.UnaryTracing,
        ),
)

conn, err := grpc.Dial(":8000", grpcClientOptions...)
// ...
```

gRPC Server

gRPCUnaryStreamKatyushagRPCTracing

```
        // UnaryTracing is an unary interceptor for adding tracing feature
for gRPC server using OpenTelemetry.
func (s *krpcServer) UnaryTracing(ctx context.Context, req interface{},
info *grpc.UnaryServerInfo, handler grpc.UnaryHandler) (interface{},
error)

// StreamTracing is a stream unary interceptor for adding tracing feature
for gRPC server using OpenTelemetry.
func (s *krpcServer) StreamTracing(srv interface{}, ss grpc.ServerStream,
info *grpc.StreamServerInfo, handler grpc.StreamHandler) error
```

```
        s := grpc.NewServer(
        grpc.ChainUnaryInterceptor(
                krpc.Server.UnaryTracing,
        ),
)
```

Logging

`TraceIdTracingTracingTraceIdglogCtxcontext.Contextcontext.ContextTraceId`

Orm

`OrmTracingTracing`

Redis

`RedisRedisTracingTracing`

Utils

`Tracinggtrace`https://godoc.org/github.com/gogf/gf/net/gtrace

- -
- -HTTP
- -GRPC

- https://opentracing.io
- https://opencensus.io
- https://opentelemetry.io
- https://github.com/open-telemetry/opentelemetry-specification/tree/main/specification