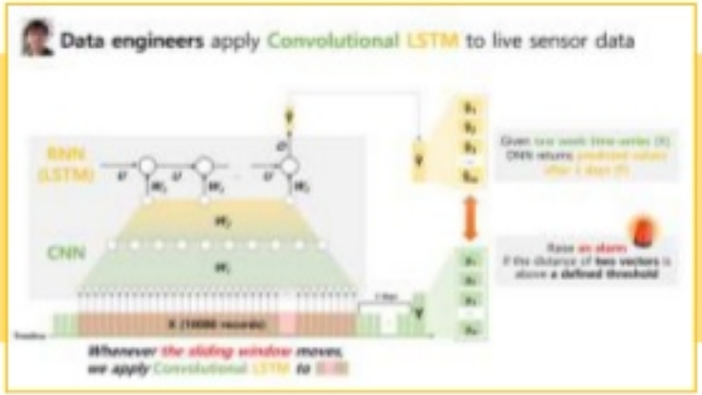
Real-time driving score service using Flink

Dongwon Kim
SK telecom

My talks @FlinkForward

Flink Forward 2015
A Comparative Performance Evaluation of Flink

Flink Forward 2017
Predictive Maintenance with Deep Learning and Flink



Flink Forward 2018
Real-time driving score service using Flink

T map, a mobile **navigation** app by SK telecom

Waze

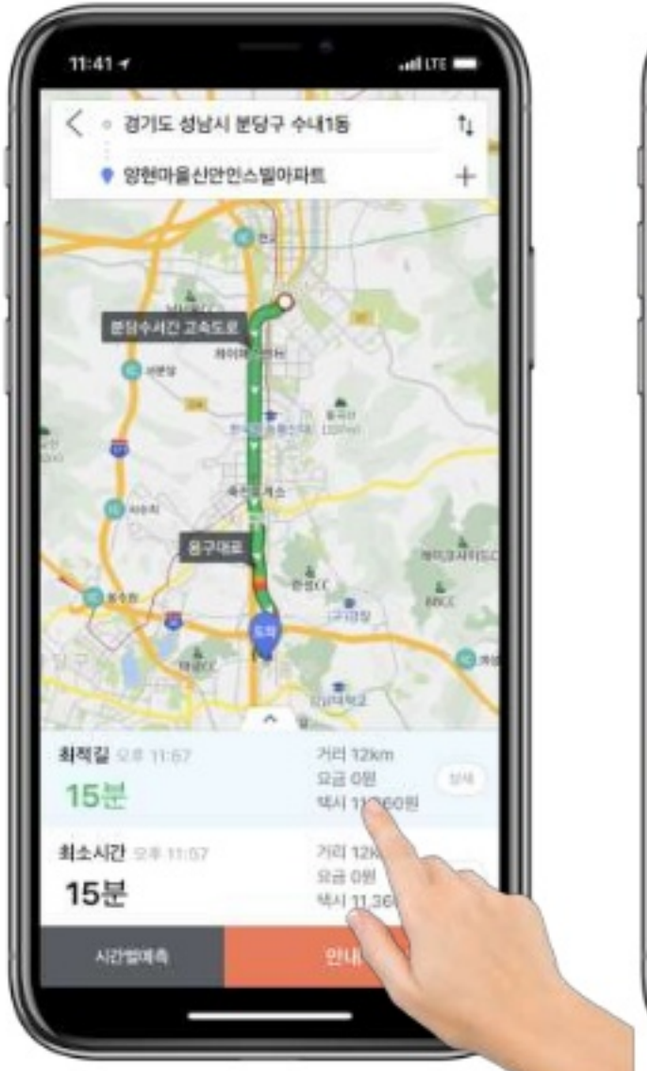
Google Maps

≈

Enter an address
or
a place name

Choose from
frequent locations

T map, a mobile **navigation** app by SK telecom



multiple route options

in driving mode

arriving at destination

Driving score service by T map

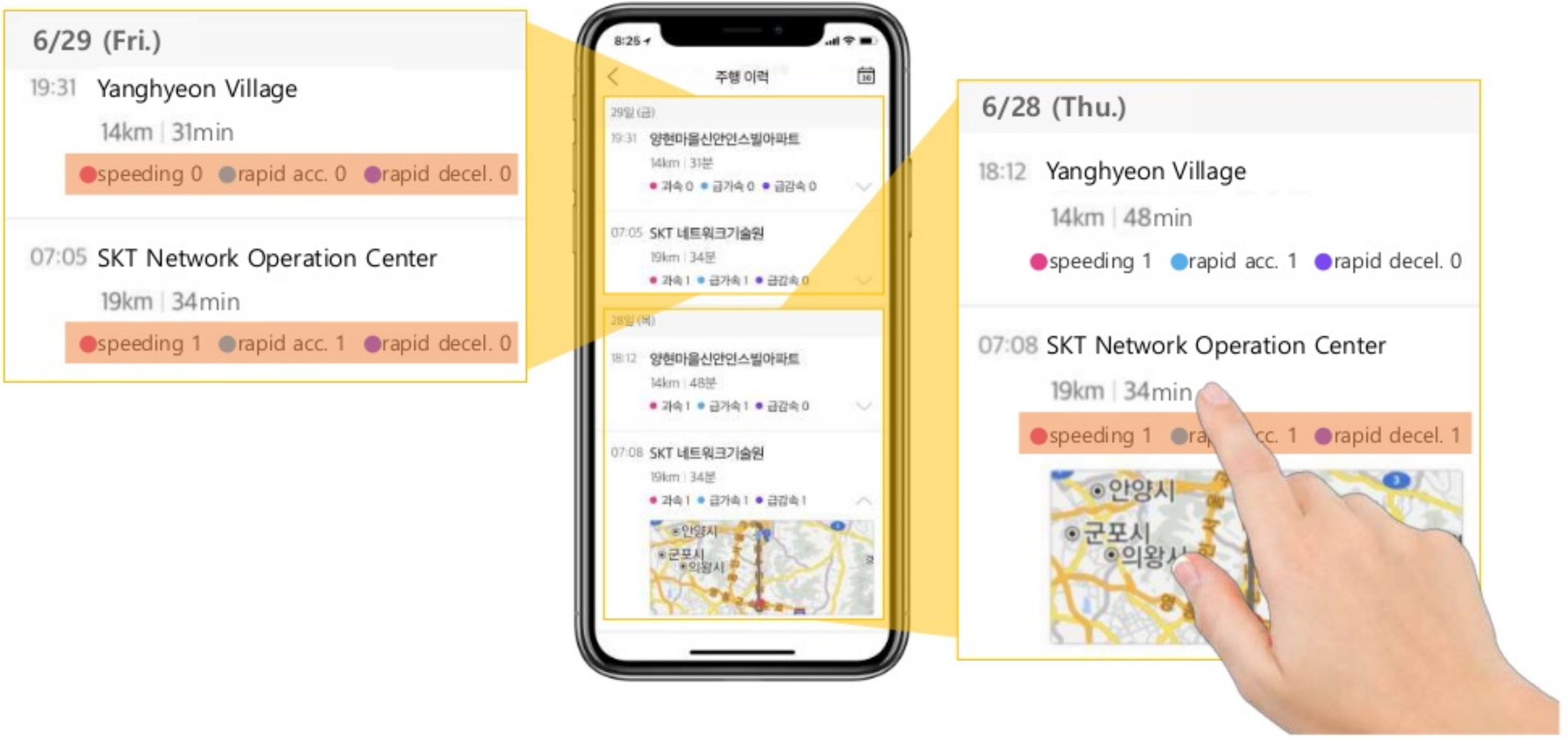
Car insurance discount for safe drivers

If you drive safely with map automobile insurance premiums go down.

KB Insurance	DB Insurance
10% discount	10% discount

Driving score is based on **three factors**

The three factors are calculated for each session

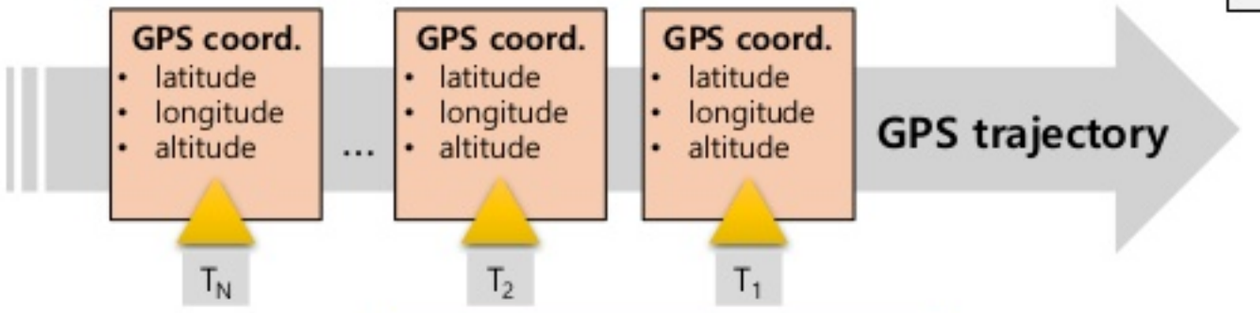


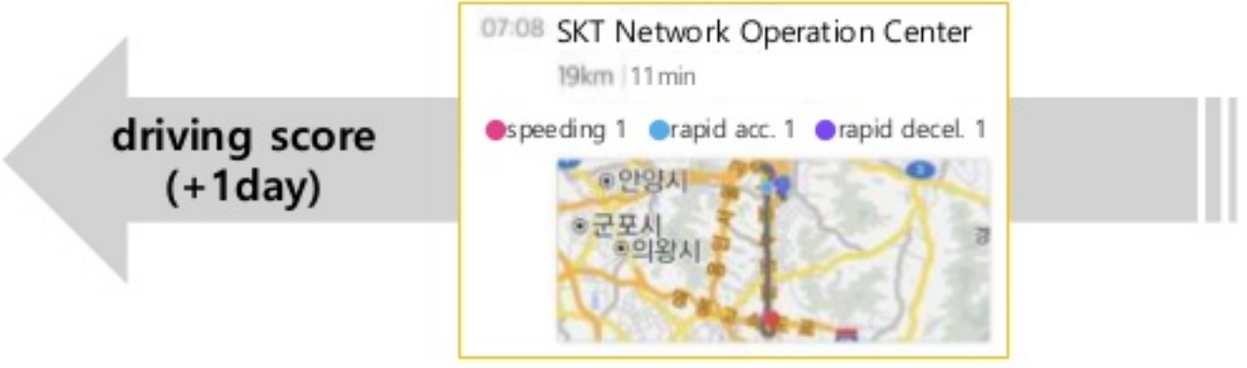
The three factors are calculated for each session

Current client-server architecture

A GPS trajectory is generated for each driving session

T map





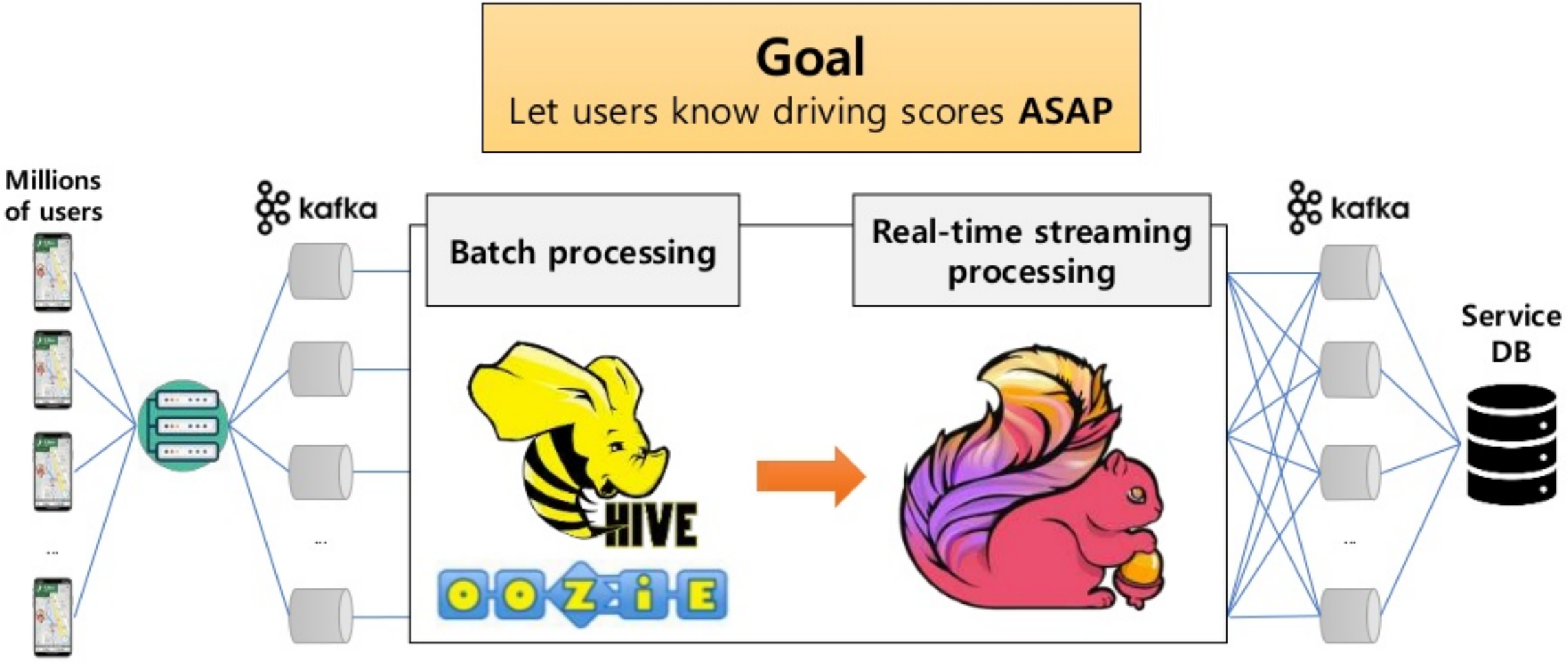
T map service server

Batch ETL jobs are executed twice a day to calculate three factors ●●● from trajectories

The main drawback

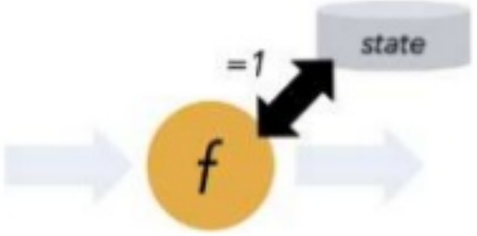
Users cannot see today's driving scores until tomorrow

Migration from **batch ETL** to **streaming processing**



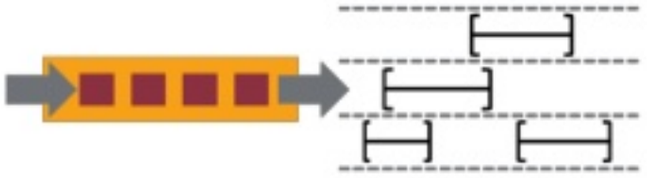
Why we choose to use Flink?

https://flink.apache.org/introduction.html#features-why-flink

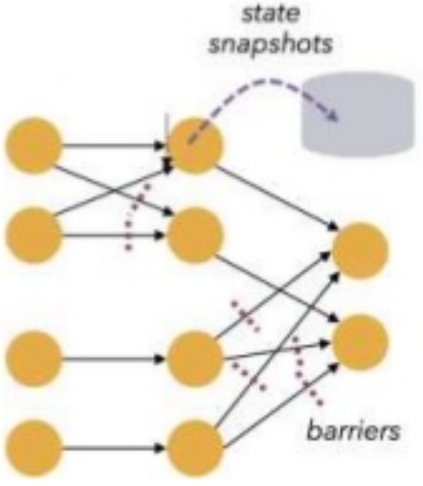


Exactly-once semantics for stateful computations

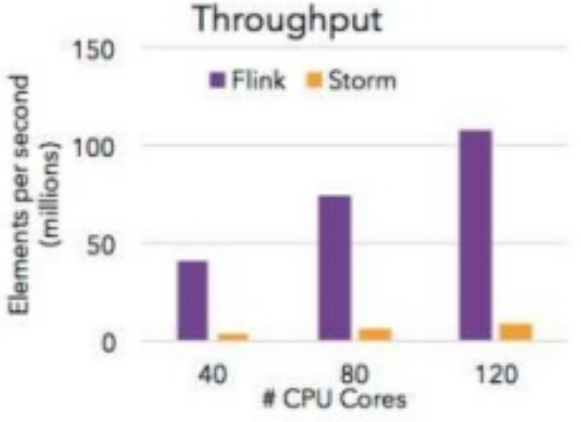
stream processing and windowing with event time semantics

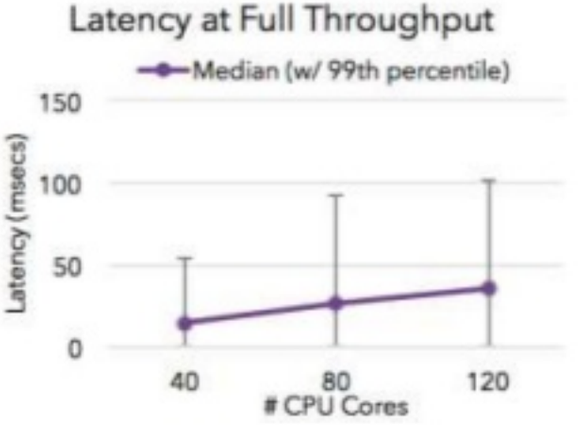


flexible windowing



light-weight fault-tolerance

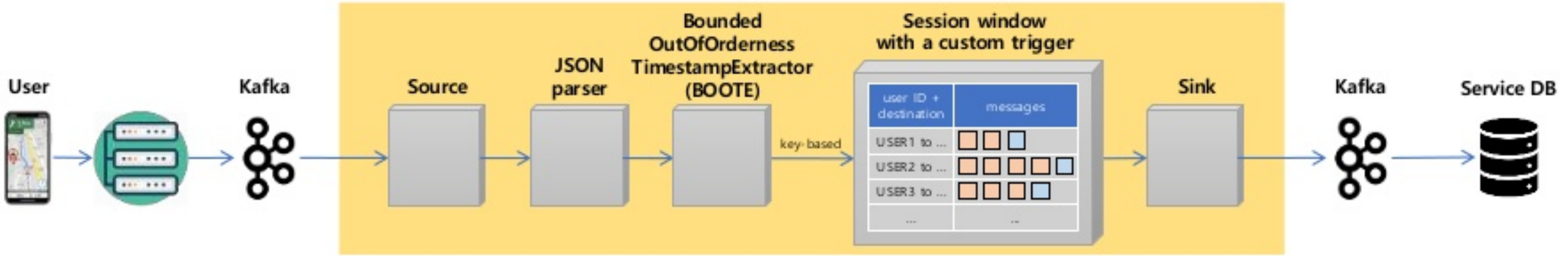




high throughput and low latency

Contents

- Dataflow design and trigger customization

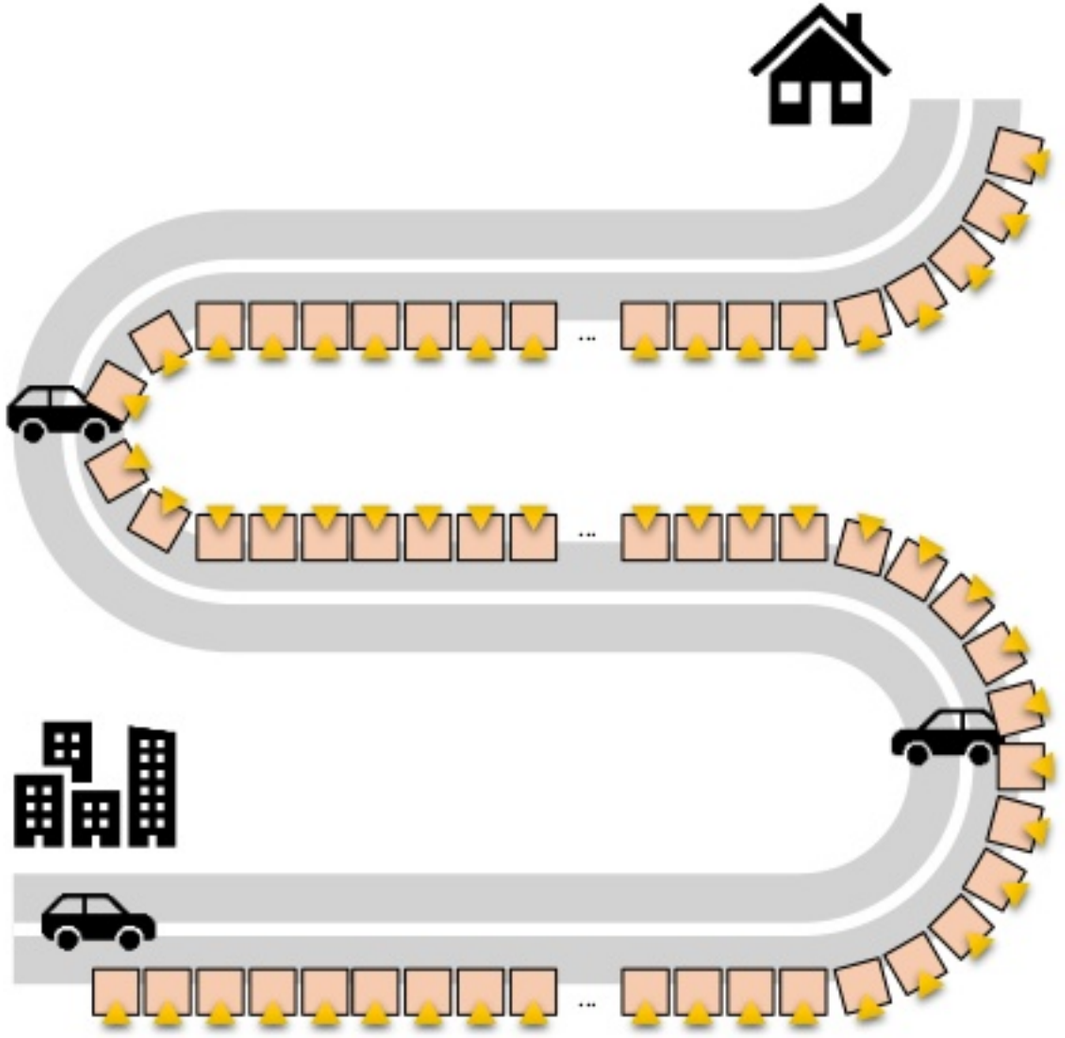


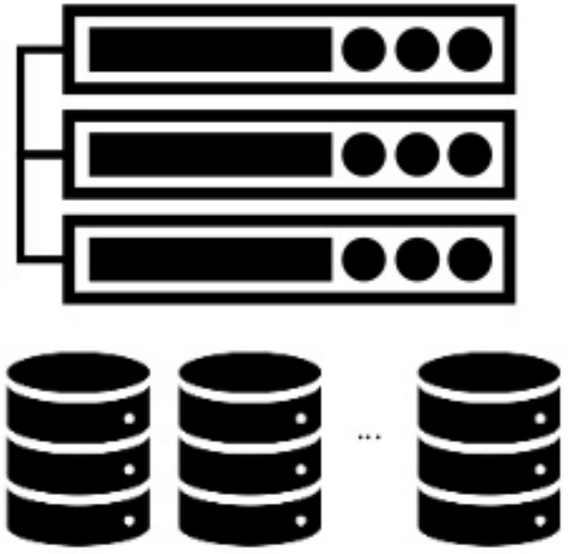
- Instrumentation with Prometheus

A 12-minute driving with 720 GPS coordinates

T map generates a GPS coordinate every second

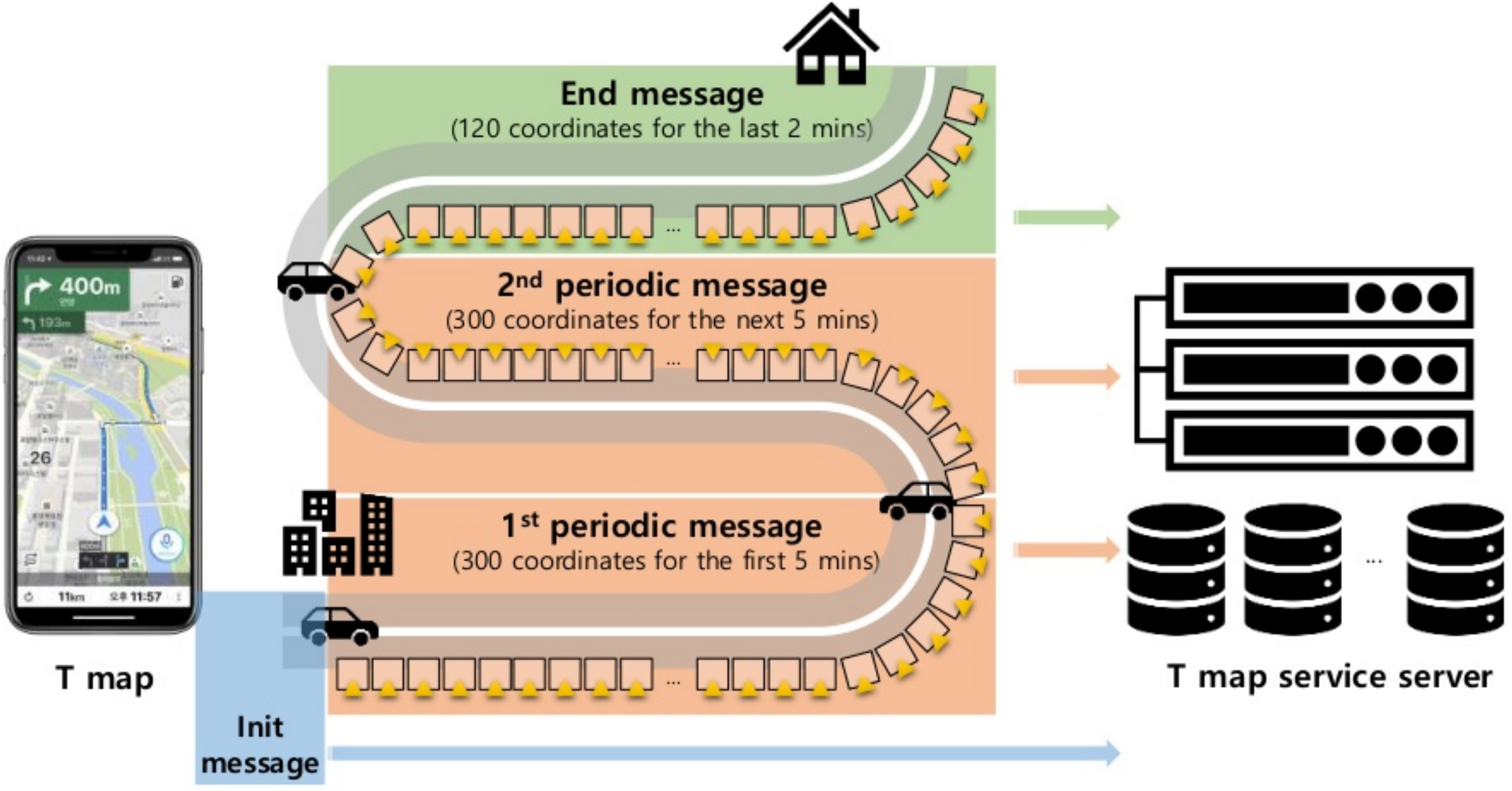
T map



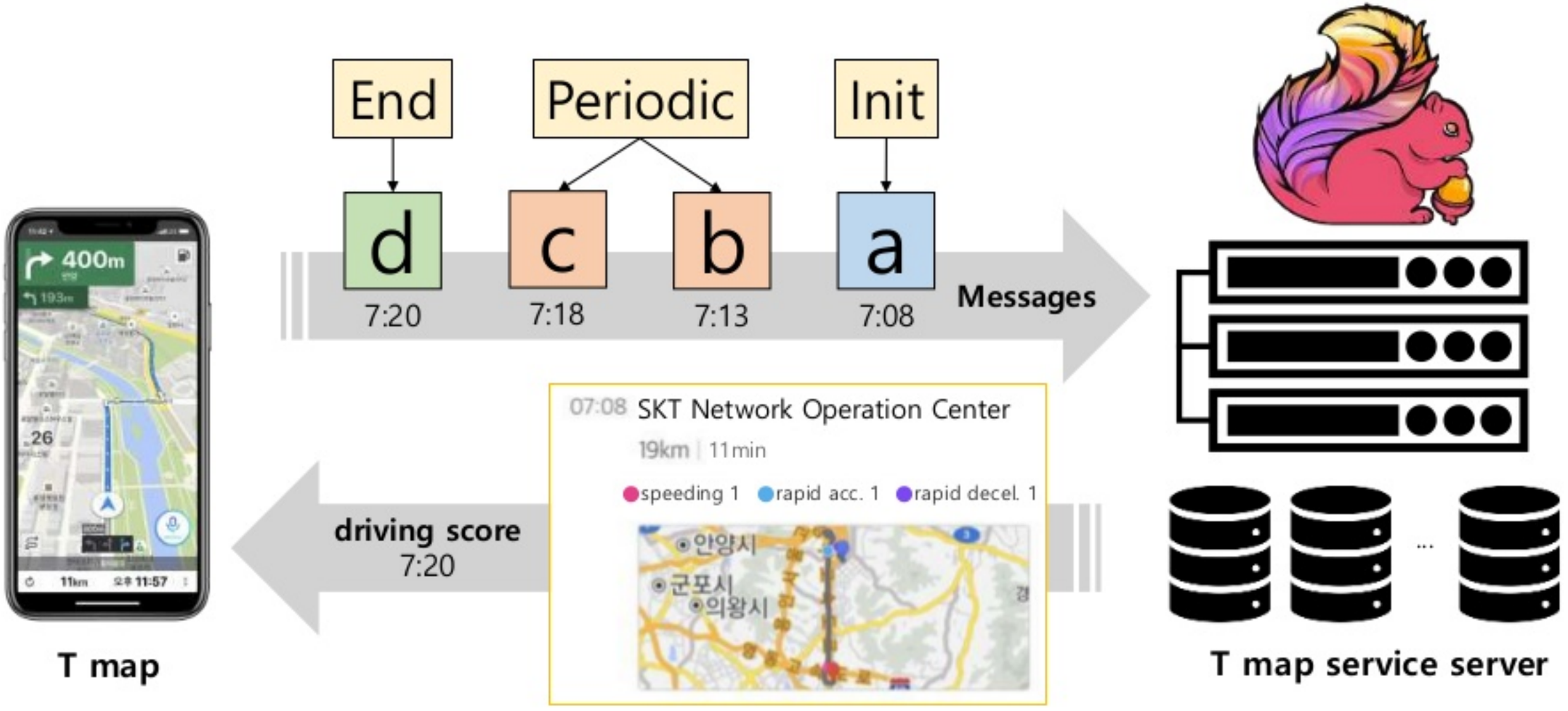


T map service server

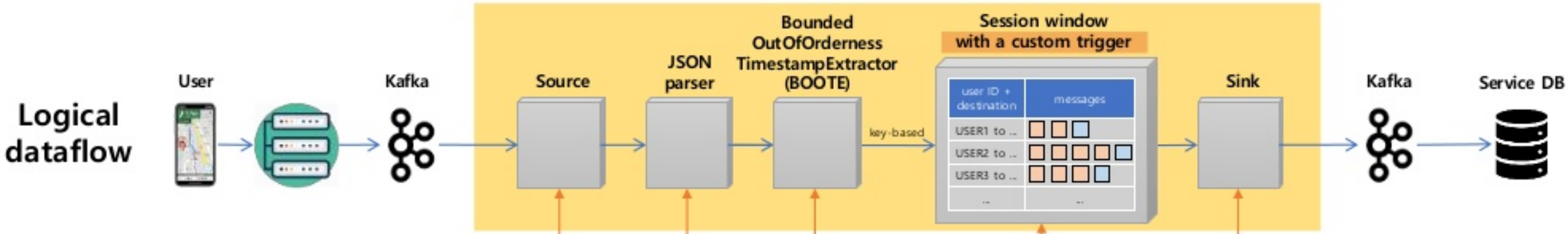
T map sends **4 messages** to the service server



Return scores right after receiving **end messages**



Real-time driving score dataflow using




```
val sourceStream = env
  .addSource(consumer)
  .setParallelism(sourceTasks)
  .process(jsonParser)
  .setParallelism(sourceTasks)
```

```
val mainDataStream = sourceStream
  .assignTimestampsAndWatermarks(
    new BoundedOutOfOrdernessTimestampExtractor[RouteRequest](Time.milliseconds(maxOutOfOrderness)) {
      override def extractTimestamp(req: RouteRequest): Long = req.eventTime
    }
  )
  .setParallelism(sourceTasks)
```

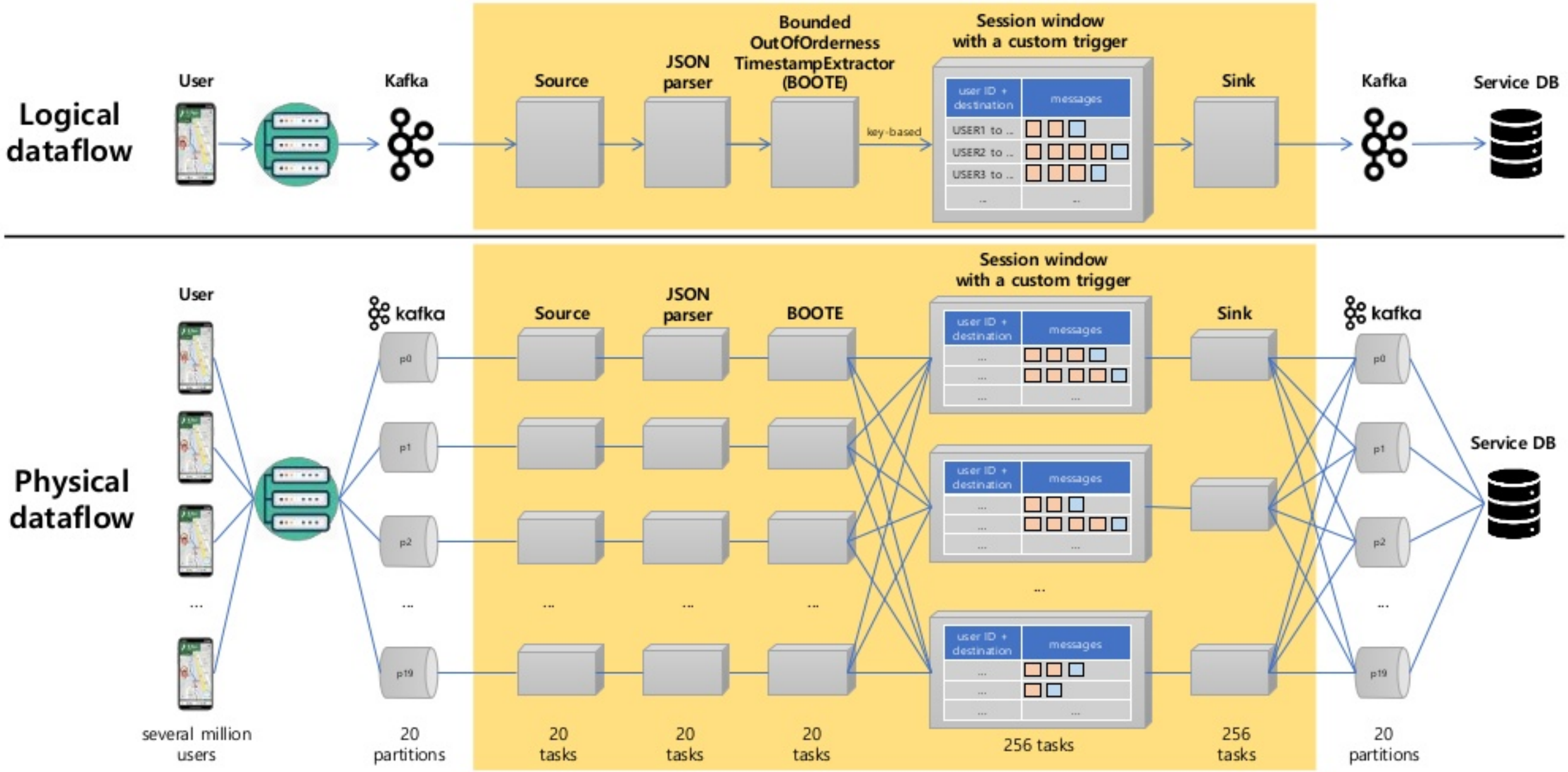
```
val sessionStream = mainDataStream
  .keyBy(keyExtractor)
  .window(EventTimeSessionWindows.withGap(Time.milliseconds(sessionGapInMillis)))
  .trigger(earlyResultTrigger)
  .aggregate(drivingScoreAggregator)
  .setParallelism(windowTasks)
```

session gap : 1 hour

```
sessionStream
  .addSink(producer)
  .setParallelism(sinkTasks)
```

at-least-once Kafka producer

Real-time driving score dataflow using



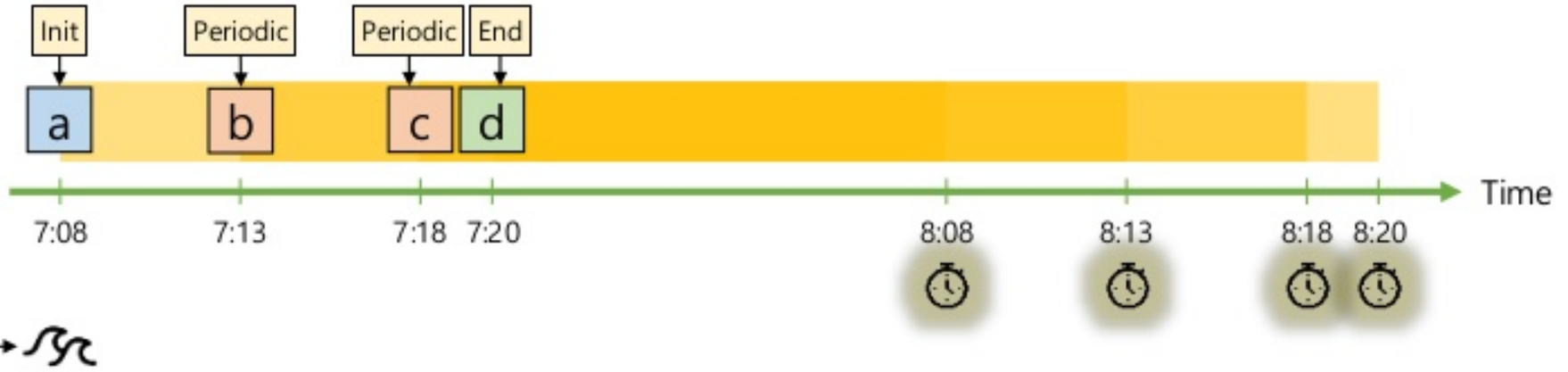
Session window (gap : 1 hour) with different triggers

The default EventTimeTrigger

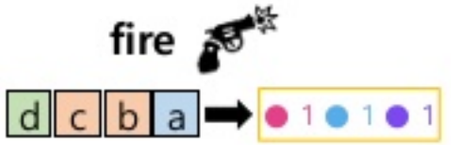
EarlyResultEventTimeTrigger

Session window (gap : 1 hour) with different triggers

The default EventTimeTrigger



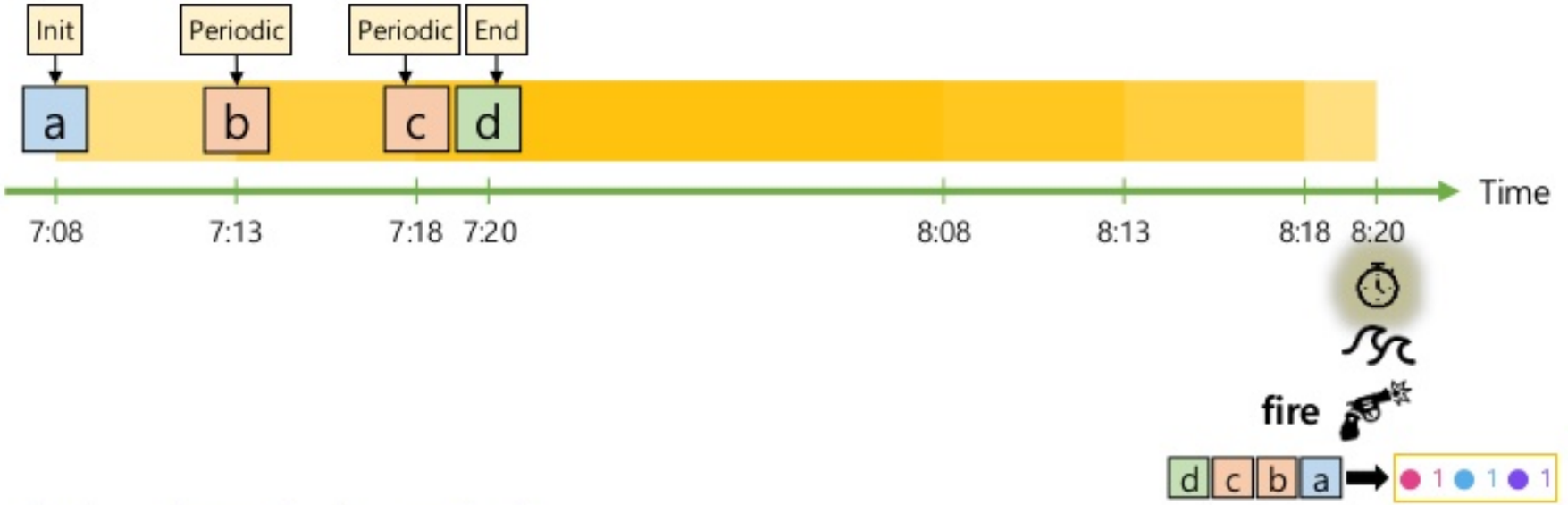
Watermark →



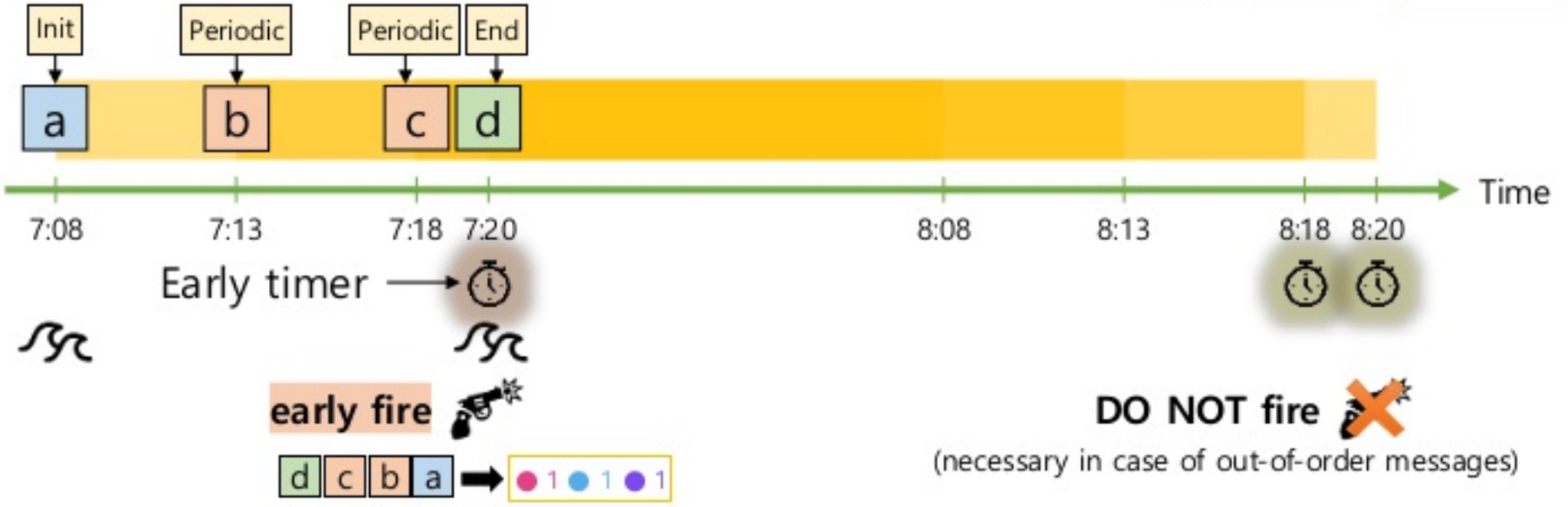
EarlyResultEventTimeTrigger

Session window (gap : 1 hour) with different triggers

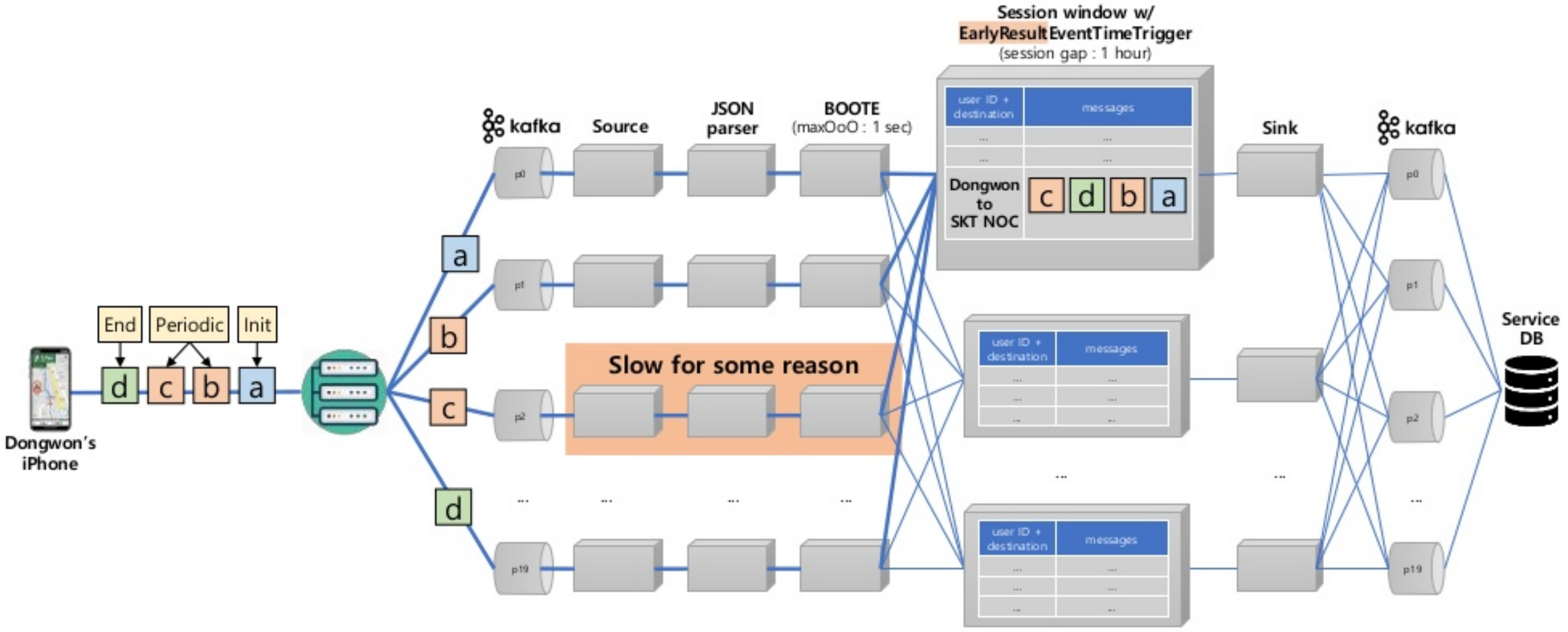
The default EventTimeTrigger



EarlyResultEventTimeTrigger



Out-or-order messages



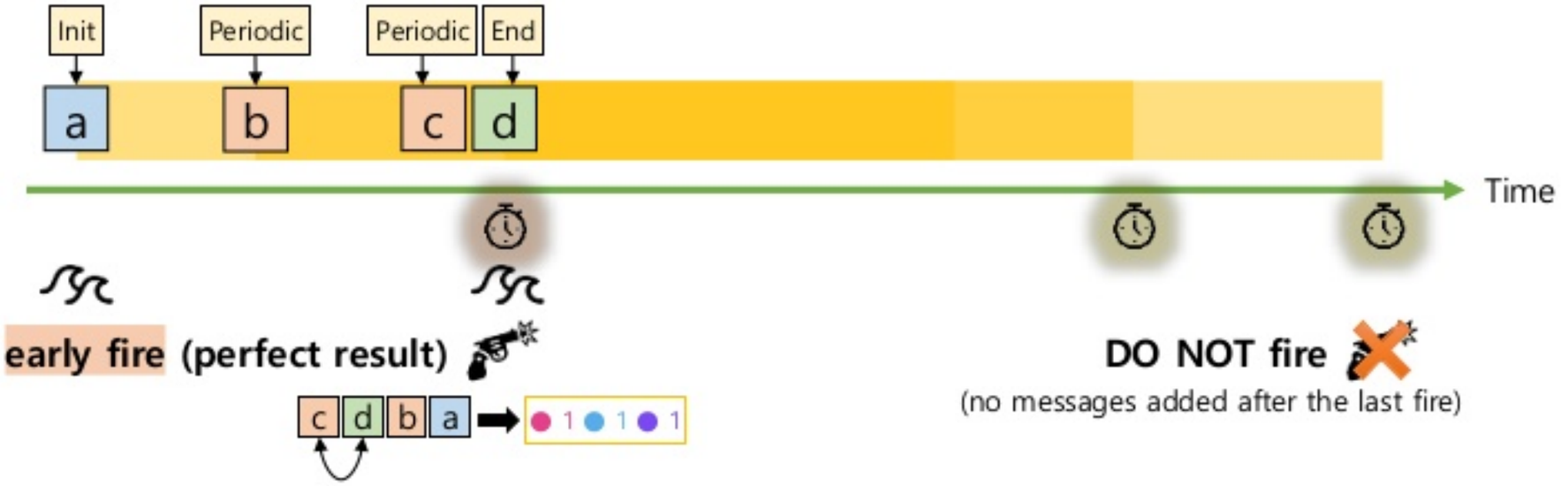
How **EarlyResult**EventTimeTrigger deals with **out-or-order messages**

[Case 1] c arrives
before the early timer expires

[Case 2] c arrives
after the early timer expires

How **EarlyResult**EventTimeTrigger deals with **out-or-order messages**

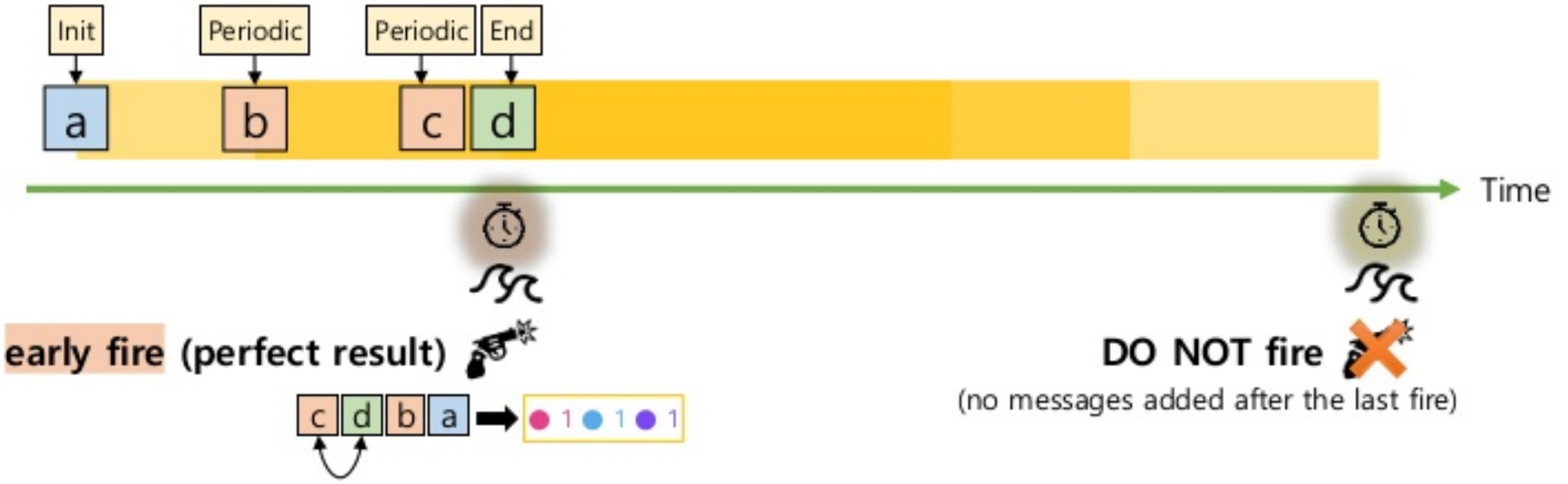
[Case 1] c arrives
before the early timer expires



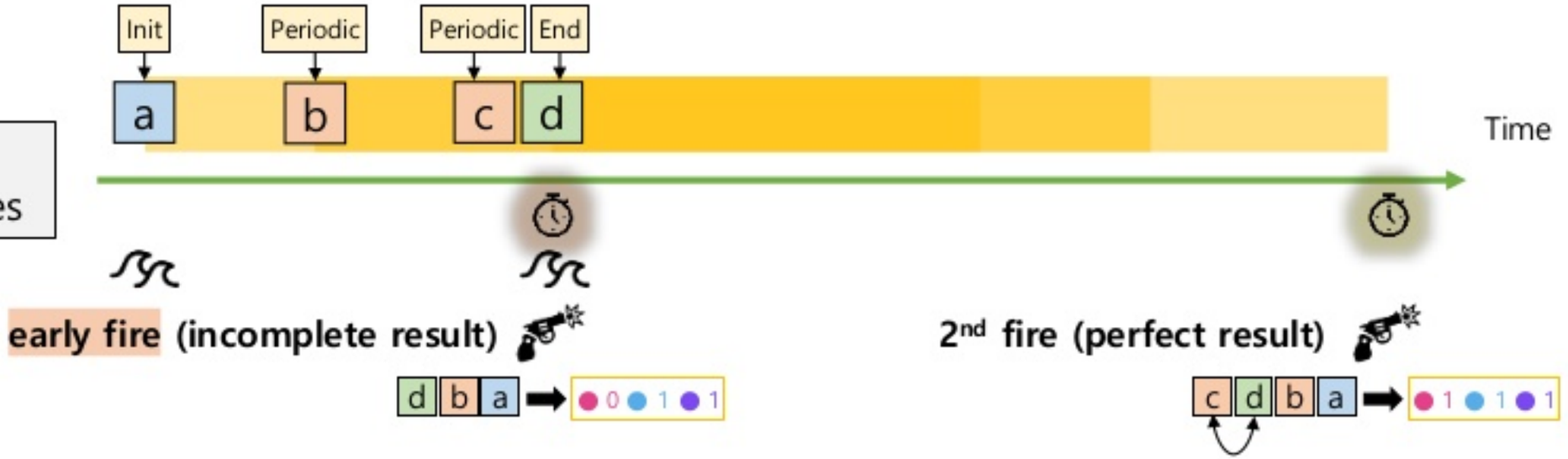
[Case 2] c arrives
after the early timer expires

How **EarlyResult**EventTimeTrigger deals with **out-or-order messages**

[Case 1] c arrives
before the early timer expires



[Case 2] c arrives
after the early timer expires



EarlyResultEventTimeTrigger

https://github.com/eastcirclek/flink-examples/blob/master/src/main/scala/com/github/eastcirclek/flink/trigger/EarlyResultEventTimeTrigger.scala

[Constructor]
Get an evaluator to determine **early firing**

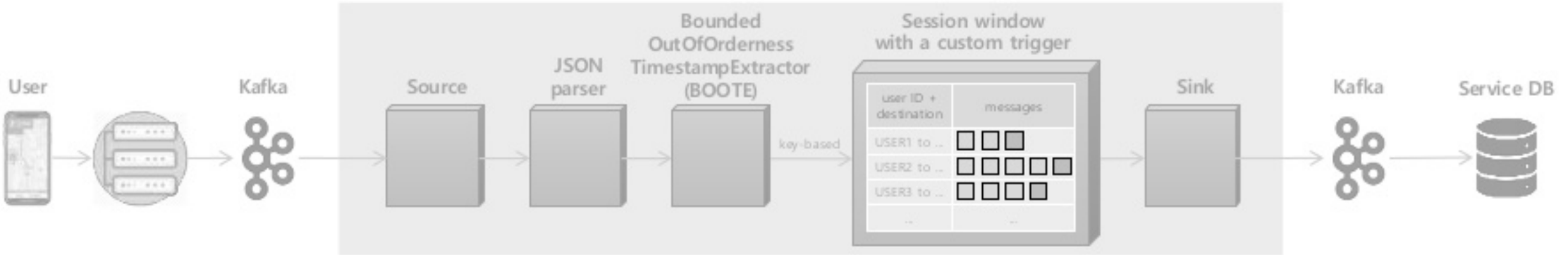
```
class EarlyResultEventTimeTrigger[T](eval: (T => Boolean)) extends Trigger[T, TimeWindow] {
  val timersDesc = new ListStateDescriptor[Long]("timers", classOf[Long])
  val countDesc = new AggregatingStateDescriptor("count", LongAdder.create(), classOf[Long])
  val lastCountWhenFiringDesc = new AggregatingStateDescriptor("lastCount", LongAdder.create(), classOf[Long])

  override def onElement(element: T, timestamp: Long, window: TimeWindow, ctx: Trigger.TriggerContext): TriggerResult = {
    ctx.getPartitionedState(countDesc).add(1)

    if (window.maxTimestamp <= ctx.getCurrentWatermark) {
      fireOrContinue(ctx)
    } else {
      if (eval(element)) {
        ctx.registerEventTimeTimer(timestamp)
        ctx.getPartitionedState(timersDesc).add(timestamp)
      }
      ctx.registerEventTimeTimer(window.maxTimestamp)
      TriggerResult.CONTINUE
    }
  }

  override def onEventTime(time: Long, window: TimeWindow, ctx: Trigger.TriggerContext): TriggerResult = {
    if (time < window.maxTimestamp) {
      ctx.deleteEventTimeTimer(time)

      val timers = ctx.getPartitionedState(timersDesc)
      timers.update(timers.get.asScala.filter(_ != time).toSeq.asJava)

      fireOrContinue(ctx)
    } else if (time == window.maxTimestamp) {
      fireOrContinue(ctx)
    } else {
      TriggerResult.CONTINUE
    }
  }
```

[onElement]
register **an early timer**
if the evaluator returns true
(e.g. when the end message comes in)

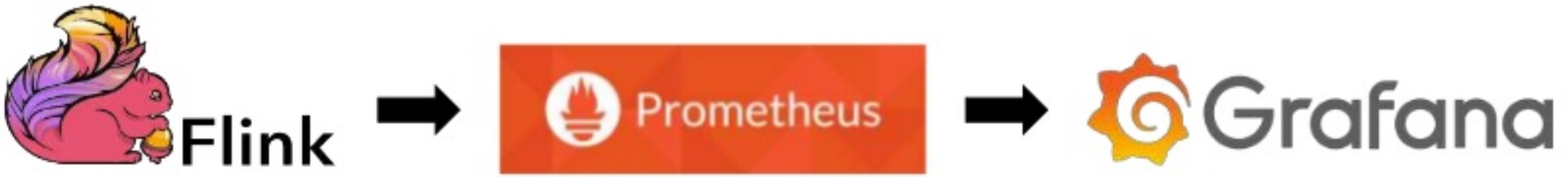
[onEventTime]
Fire if **the early timer** expires

Contents

- Dataflow design and trigger customization

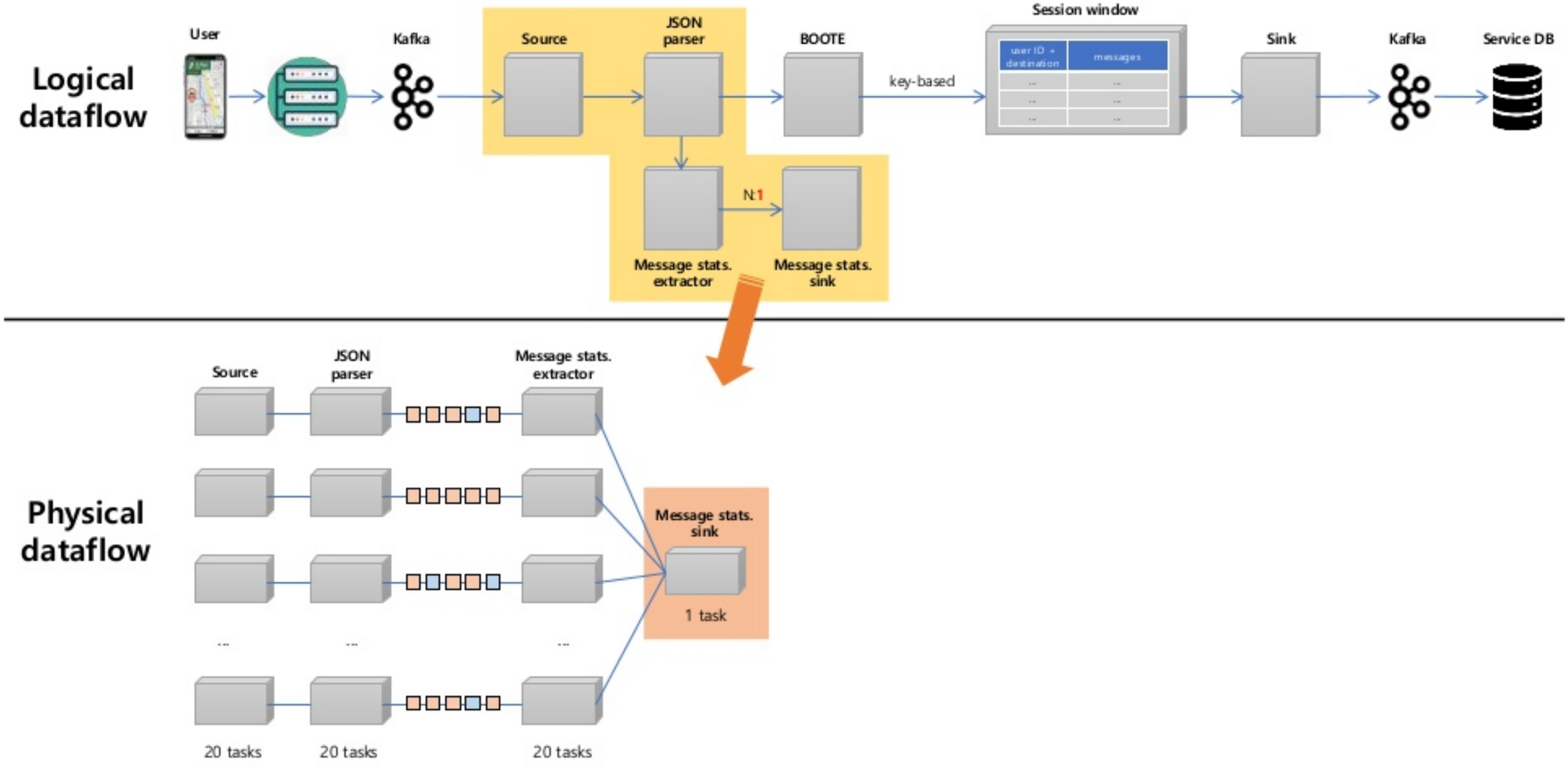


- Instrumentation with Prometheus

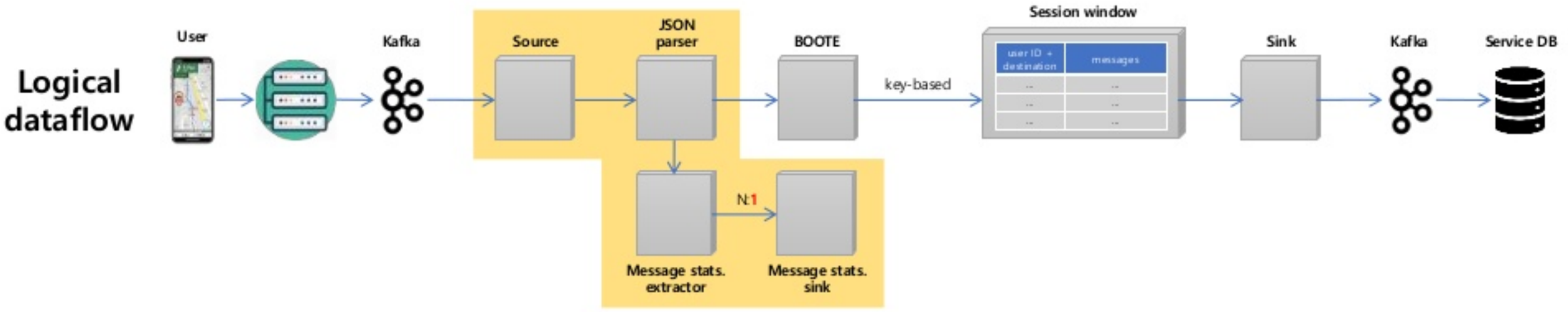


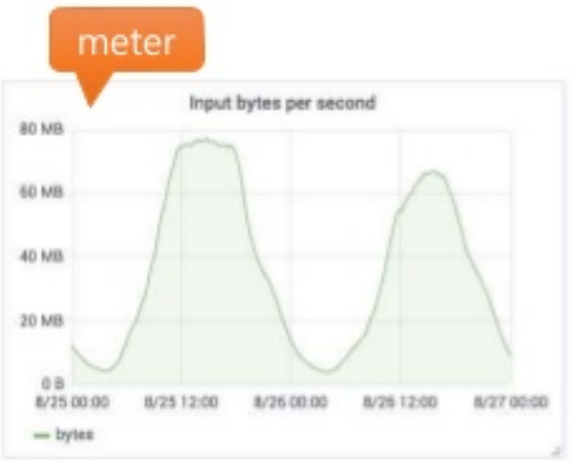
Define metrics → **Collect metrics** → **Plot metrics**

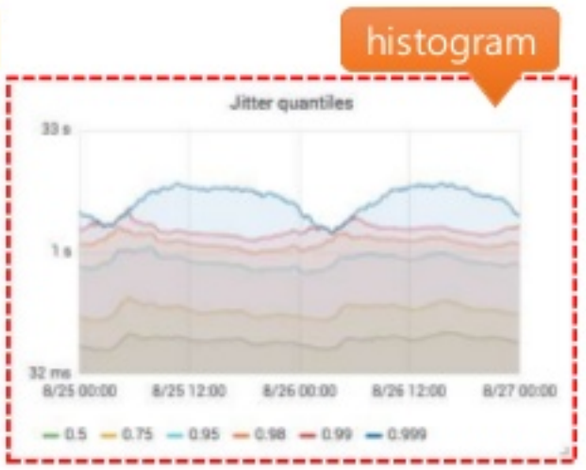
Individual message statistics



Individual message statistics



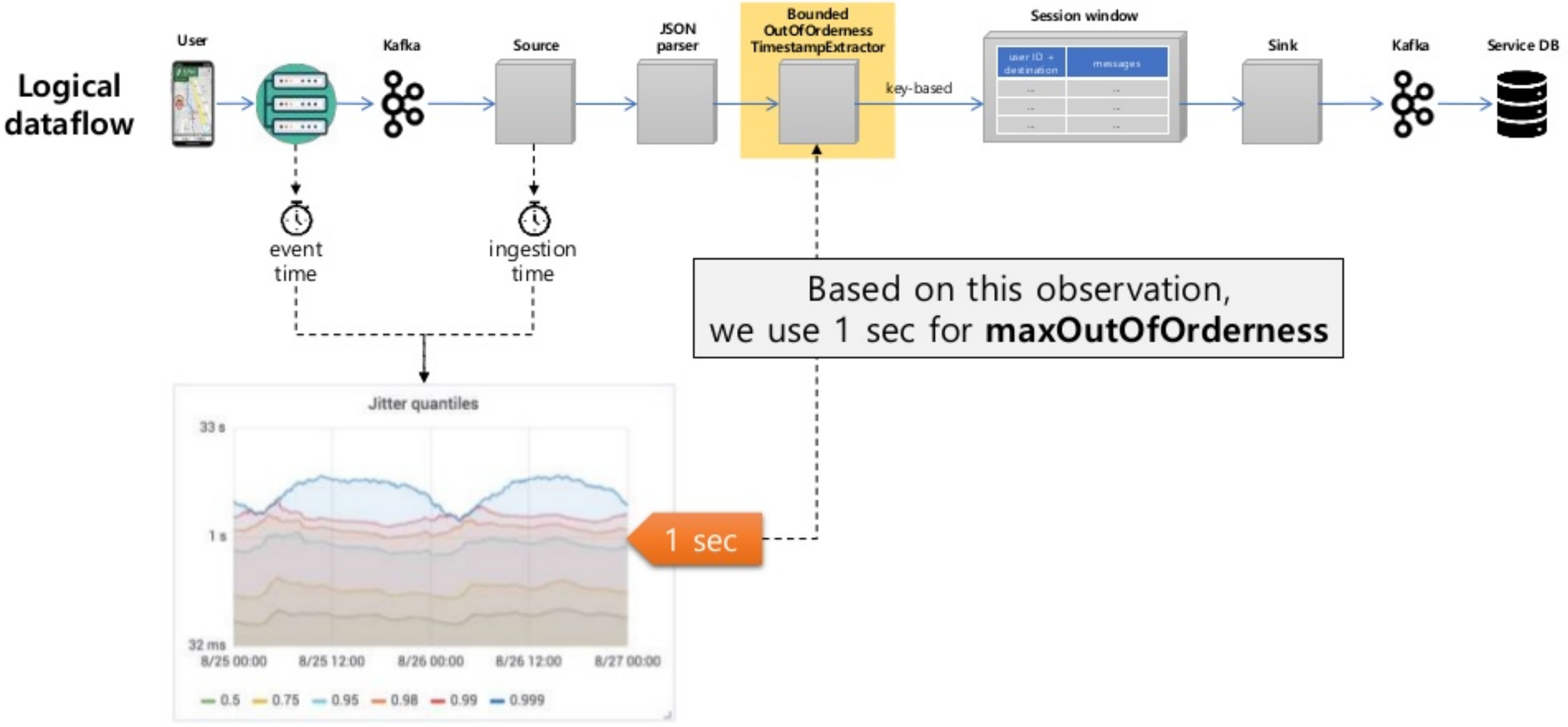




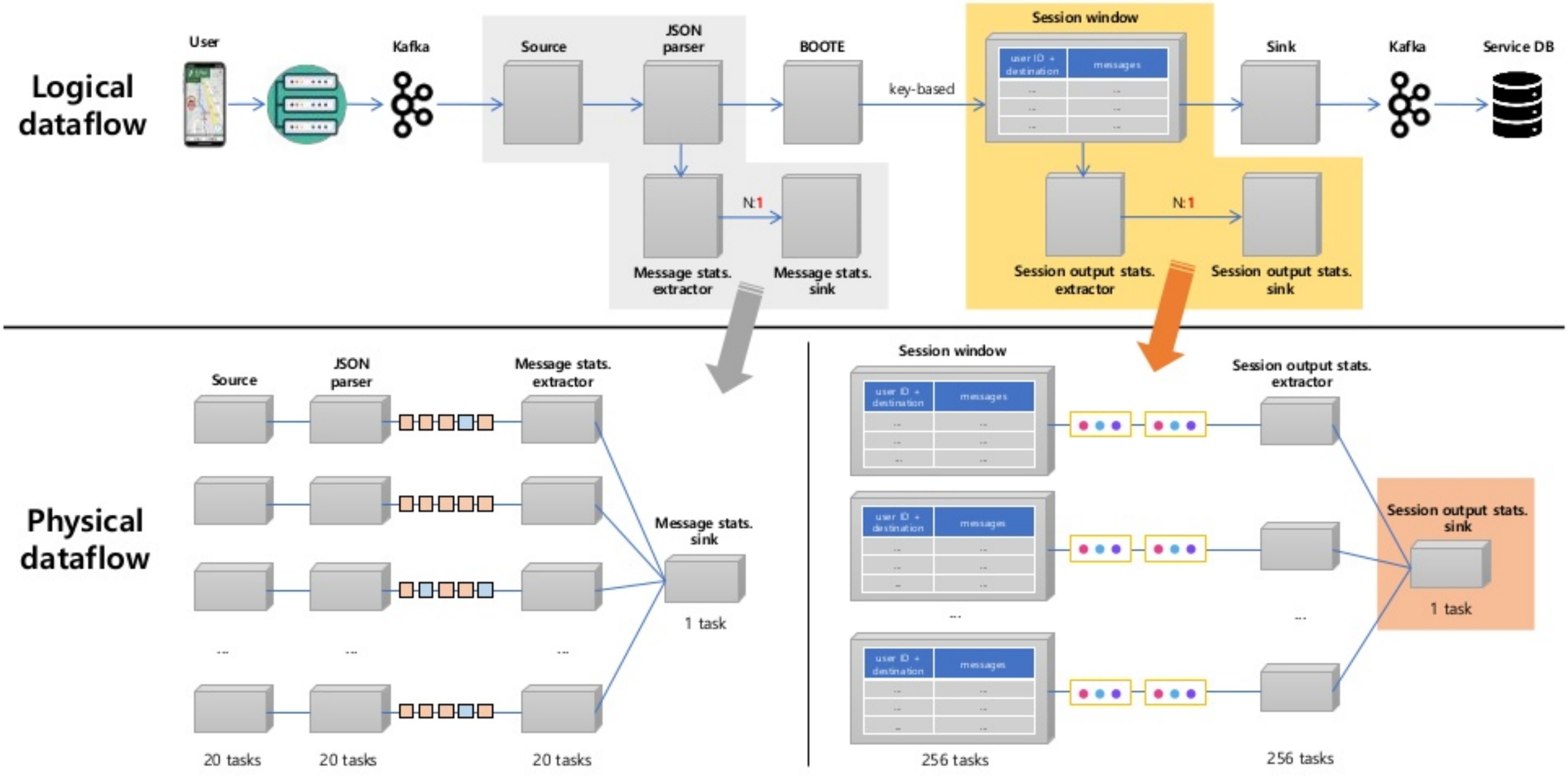
1K messages per second
100M messages per day

10s of MB per second
2 TB per day

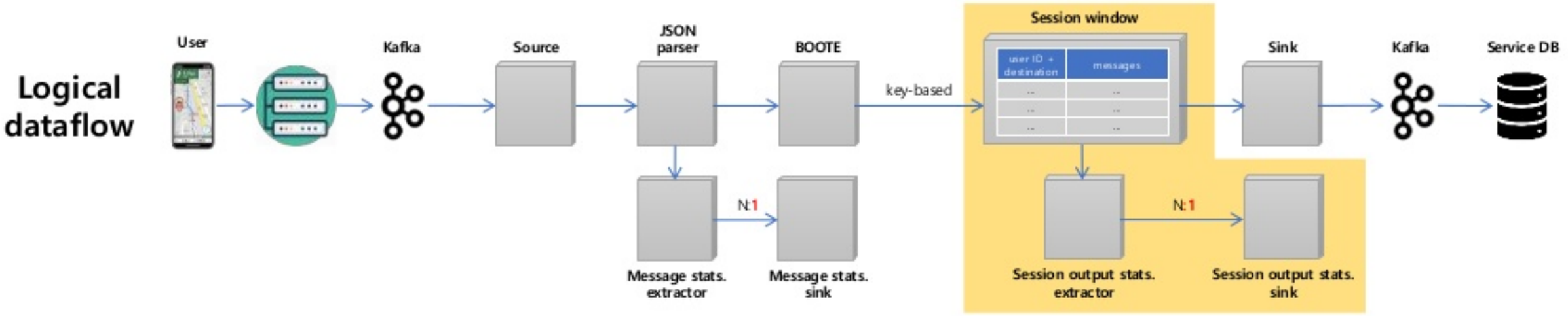
Jitter (ingestion time – event time)

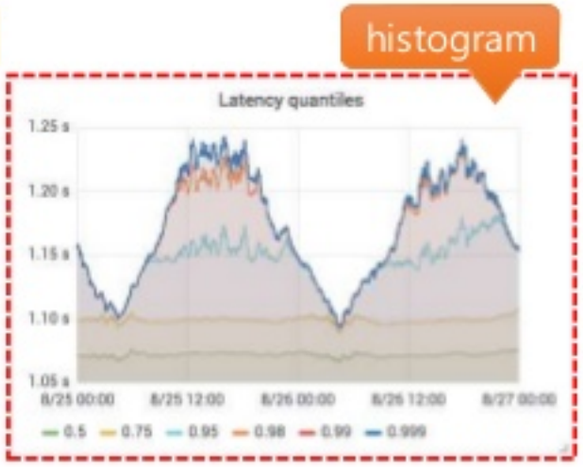


Session output statistics

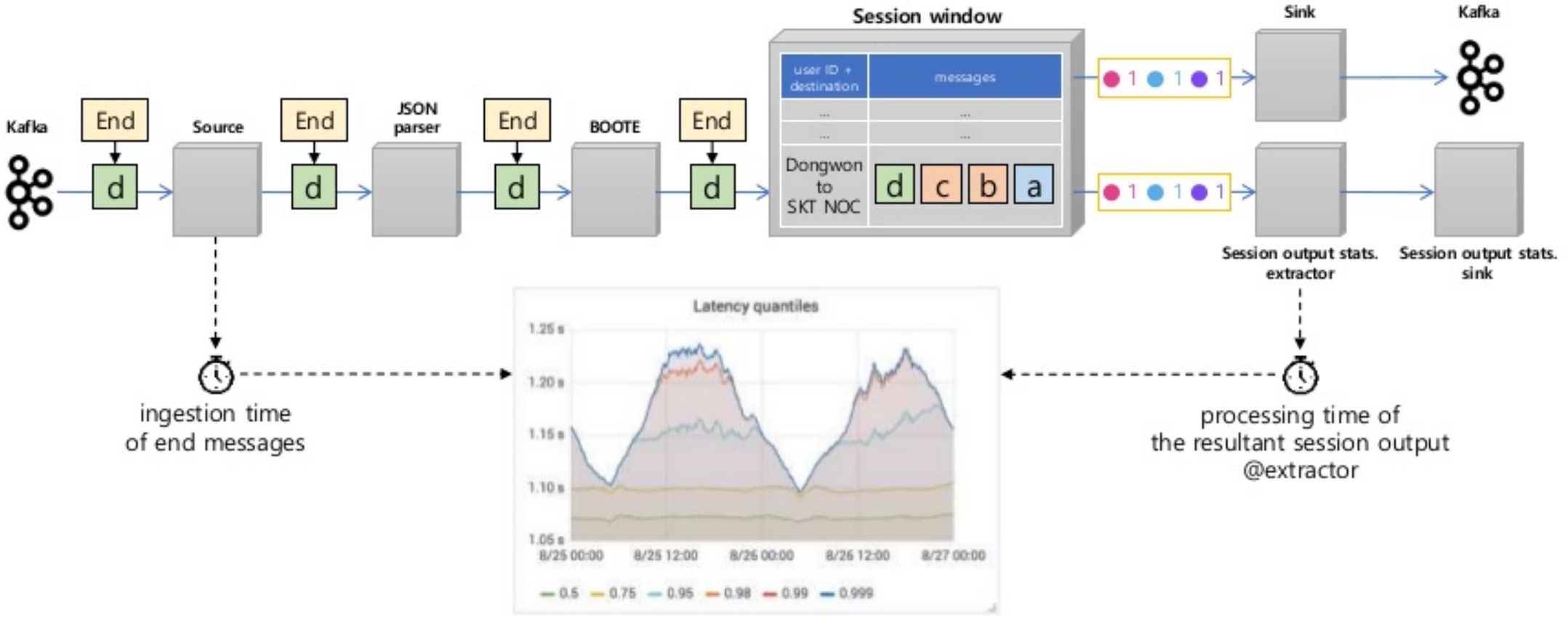


Session output statistics

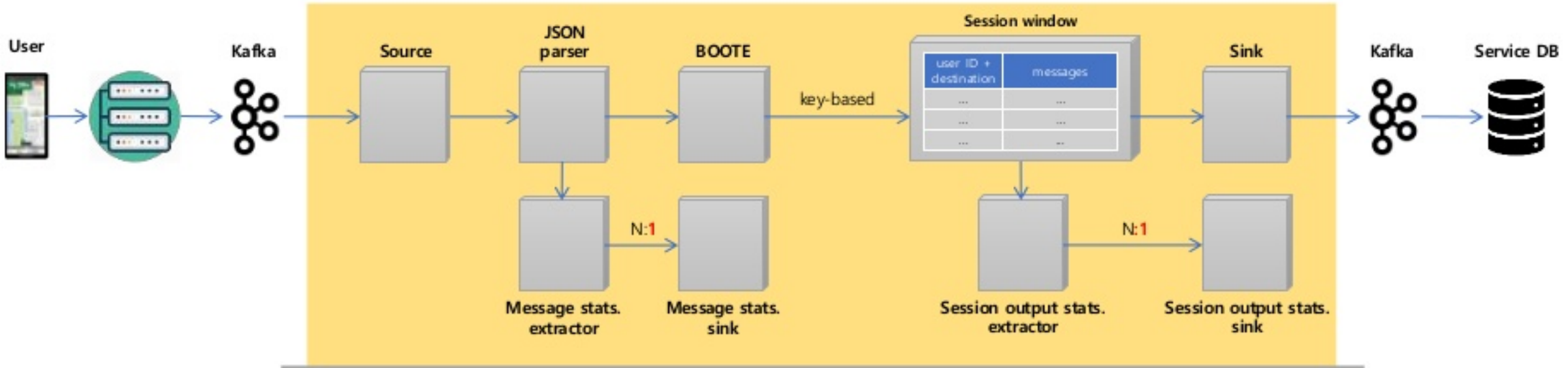




Our own definition of latency



Considering **maxOutOfOrderness** is 1 second,
Flink takes at most 250 milliseconds

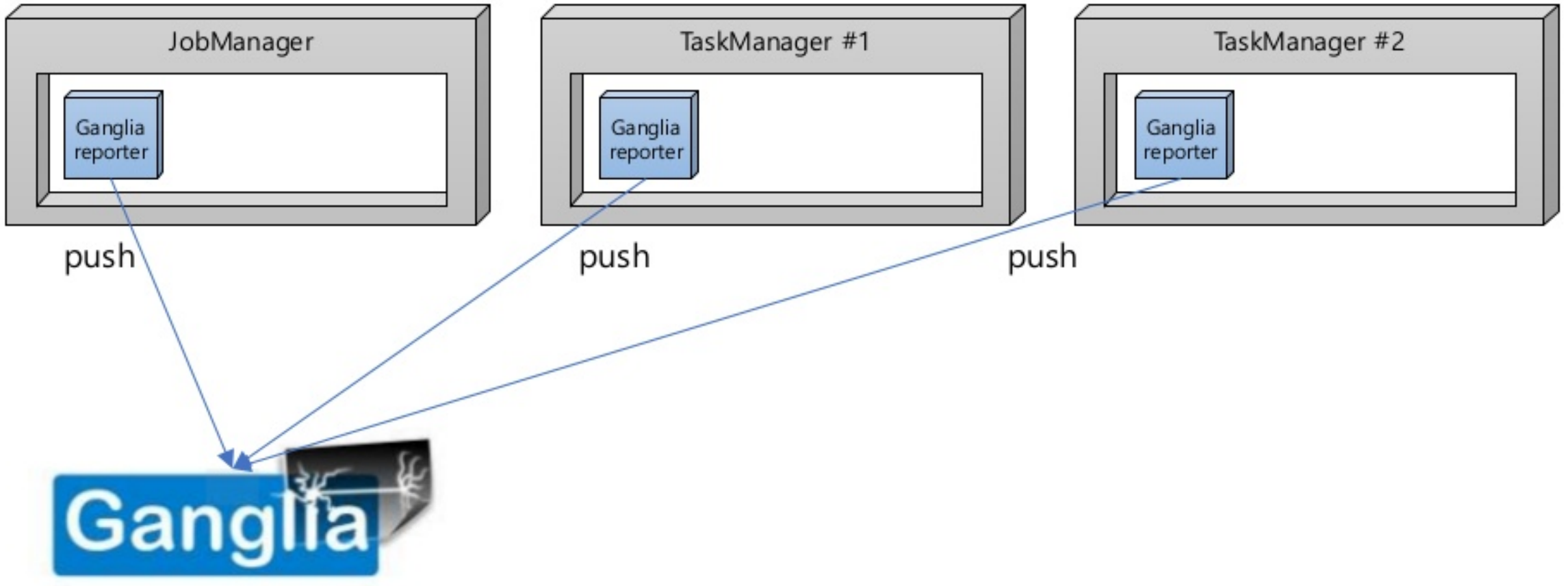


How to expose metrics to **Prometheus**?

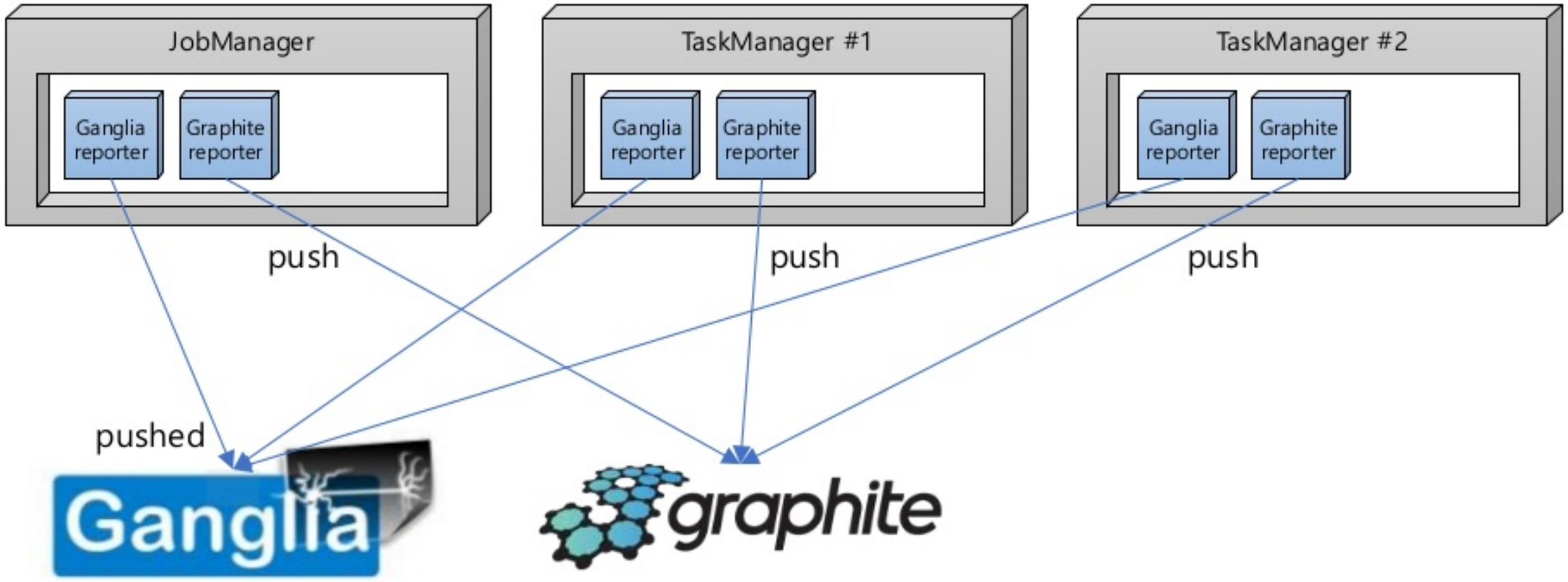
Flink metric reporters

Reporter
JMX (org.apache.flink.metrics.jmx.JMXReporter)
Ganglia (org.apache.flink.metrics.ganglia.GangliaReporter)
Graphite (org.apache.flink.metrics.graphite.GraphiteReporter)
Prometheus (org.apache.flink.metrics.prometheus.PrometheusReporter)
PrometheusPushGateway (org.apache.flink.metrics.prometheus.PrometheusPushGatewayReporter)
StatsD (org.apache.flink.metrics.statsd.StatsDReporter)
Datadog (org.apache.flink.metrics.datadog.DatadogHttpReporter)
Slf4j (org.apache.flink.metrics.slf4j.Slf4jReporter)

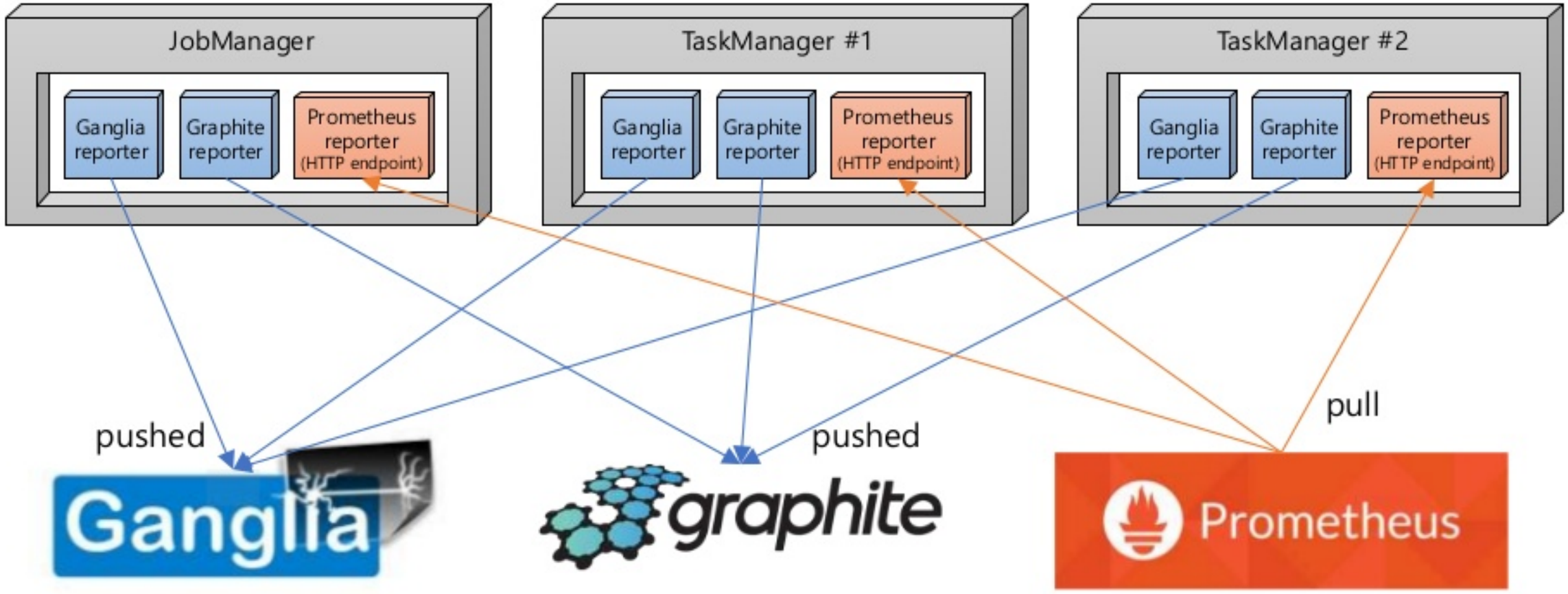
Push-model and pull-model



Push-model and pull-model



Push-model and pull-model



Reporter configuration

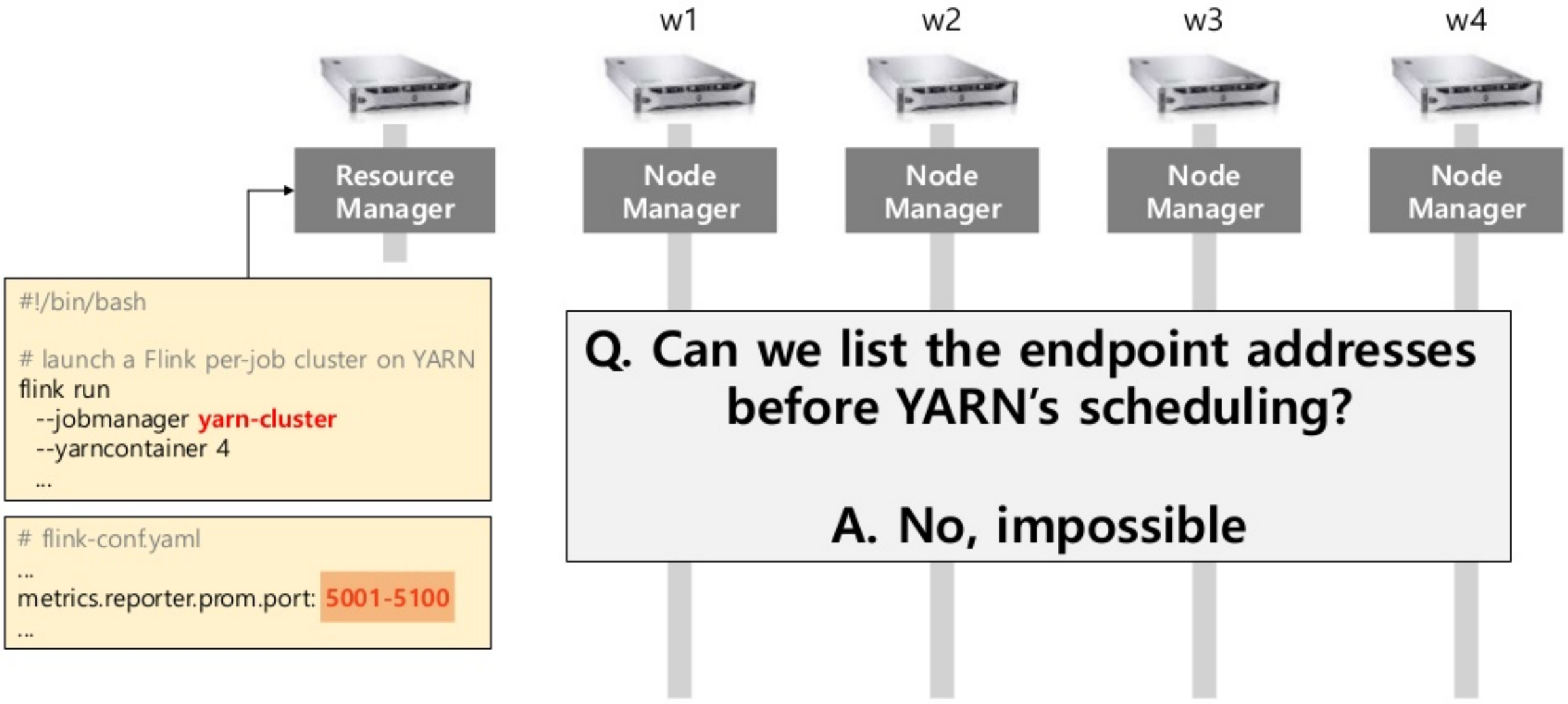
https://ci.apache.org/projects/flink/flink-docs-stable/monitoring/metrics.html#reporter

- `host` - the gmond host address configured under `udp_recv_channel.bind` in `gmond.conf`
- `port` - the gmond port configured under `udp_recv_channel.port` in `gmond.conf`
- `tmax` - soft limit for how long an old metric should be retained
- `dmax` - hard limit for how long an old metric should be retained
- `ttl` - time-to-live for transmitted UDP packets
- `addressingMode` - UDP addressing mode to use (UNICAST/MULTICAST)

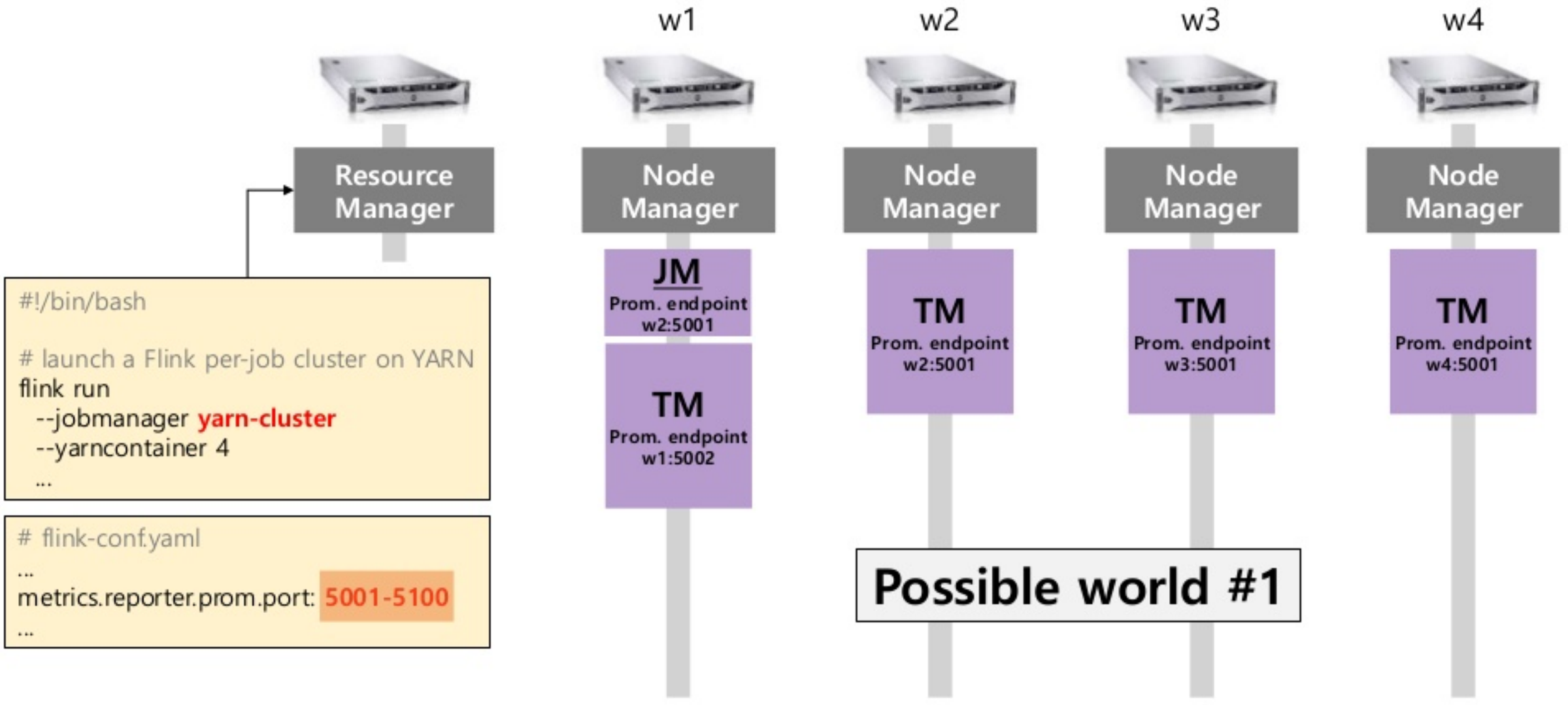
- `host` - the Graphite server host
- `port` - the Graphite server port
- `protocol` - protocol to use (TCP/UDP)

- `port` - (optional) the port the Prometheus exporter listens on, defaults to 9249. In order to be able to run several instances of the reporter on one host (e.g. when one TaskManager is colocated with the JobManager) it is advisable to use a port range like `9250-9260`.

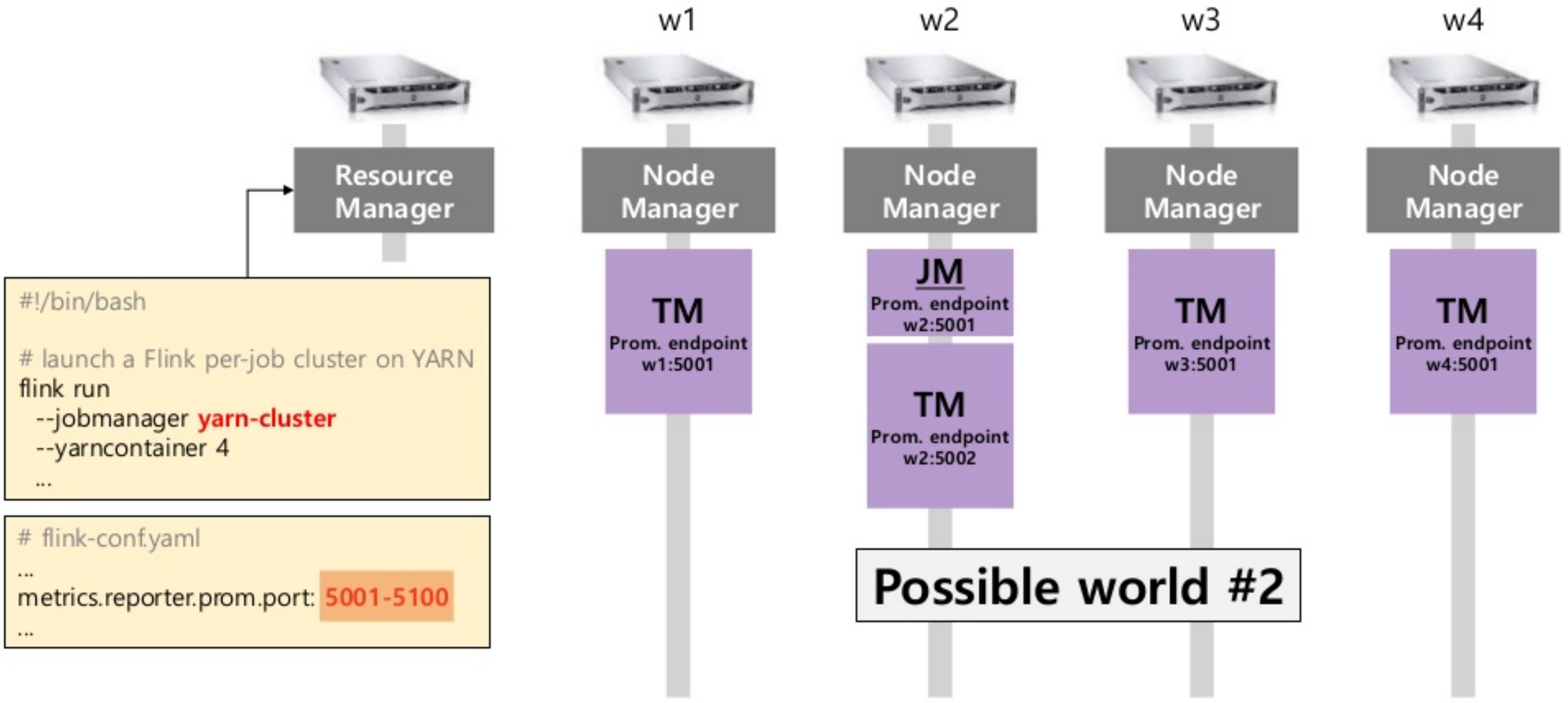
Endpoint addresses cannot be determined in advance



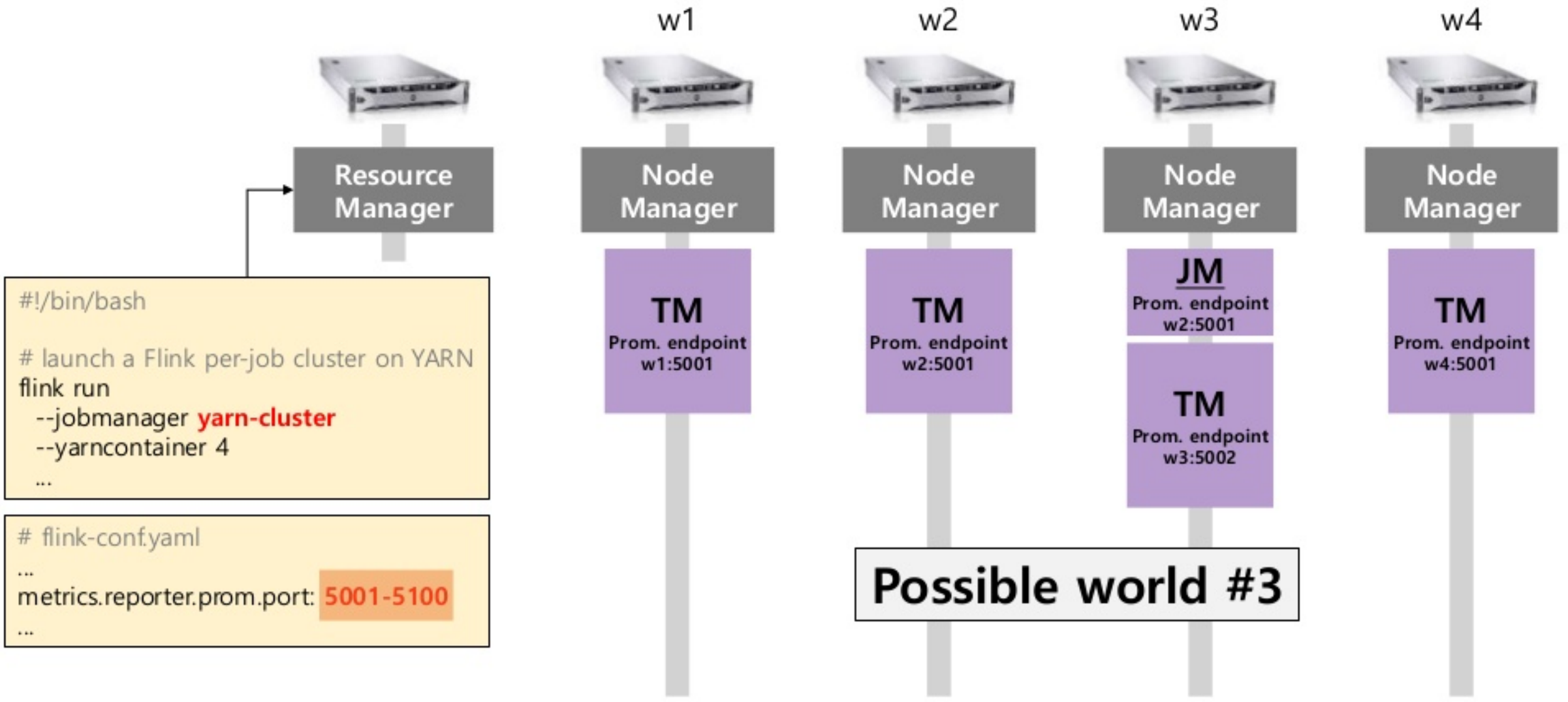
Endpoint addresses cannot be determined in advance



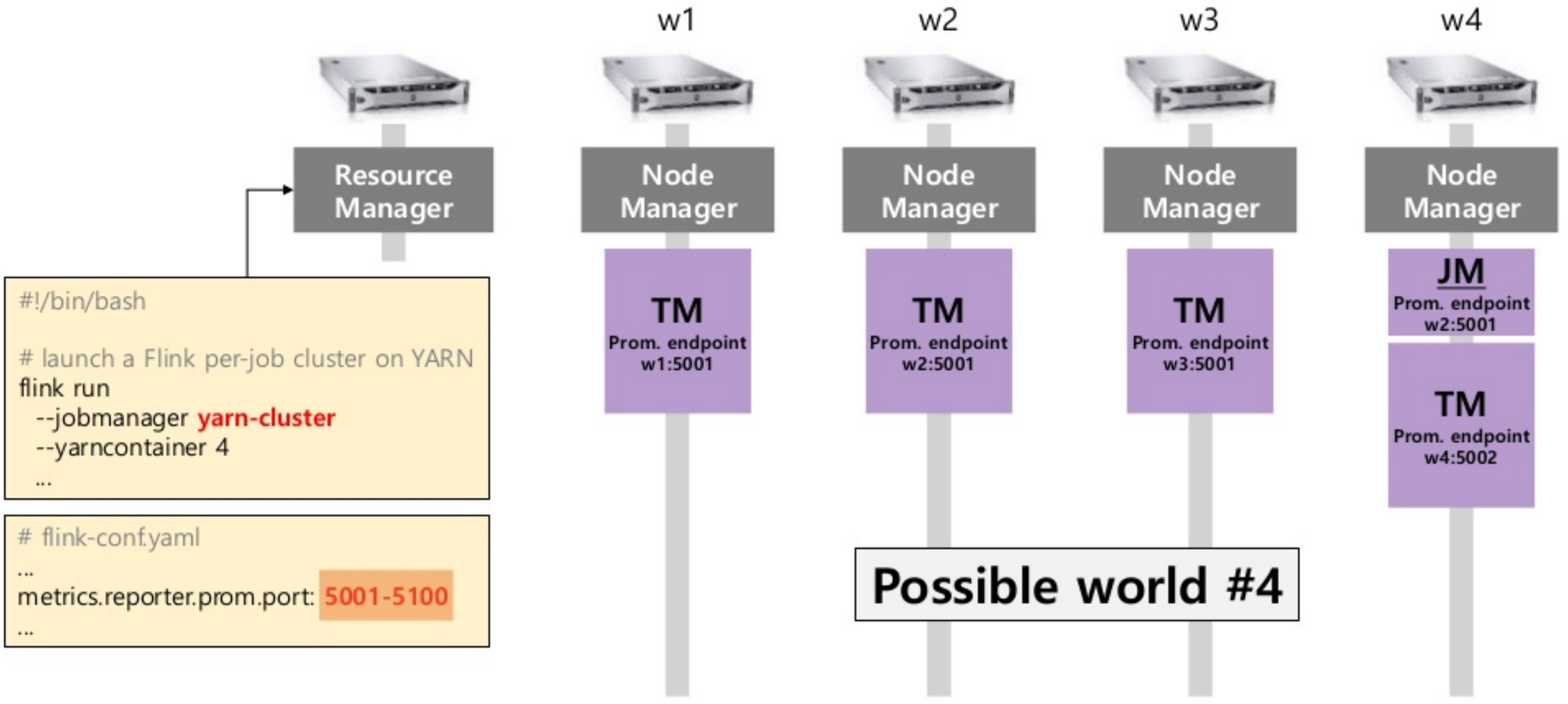
Endpoint addresses cannot be determined in advance



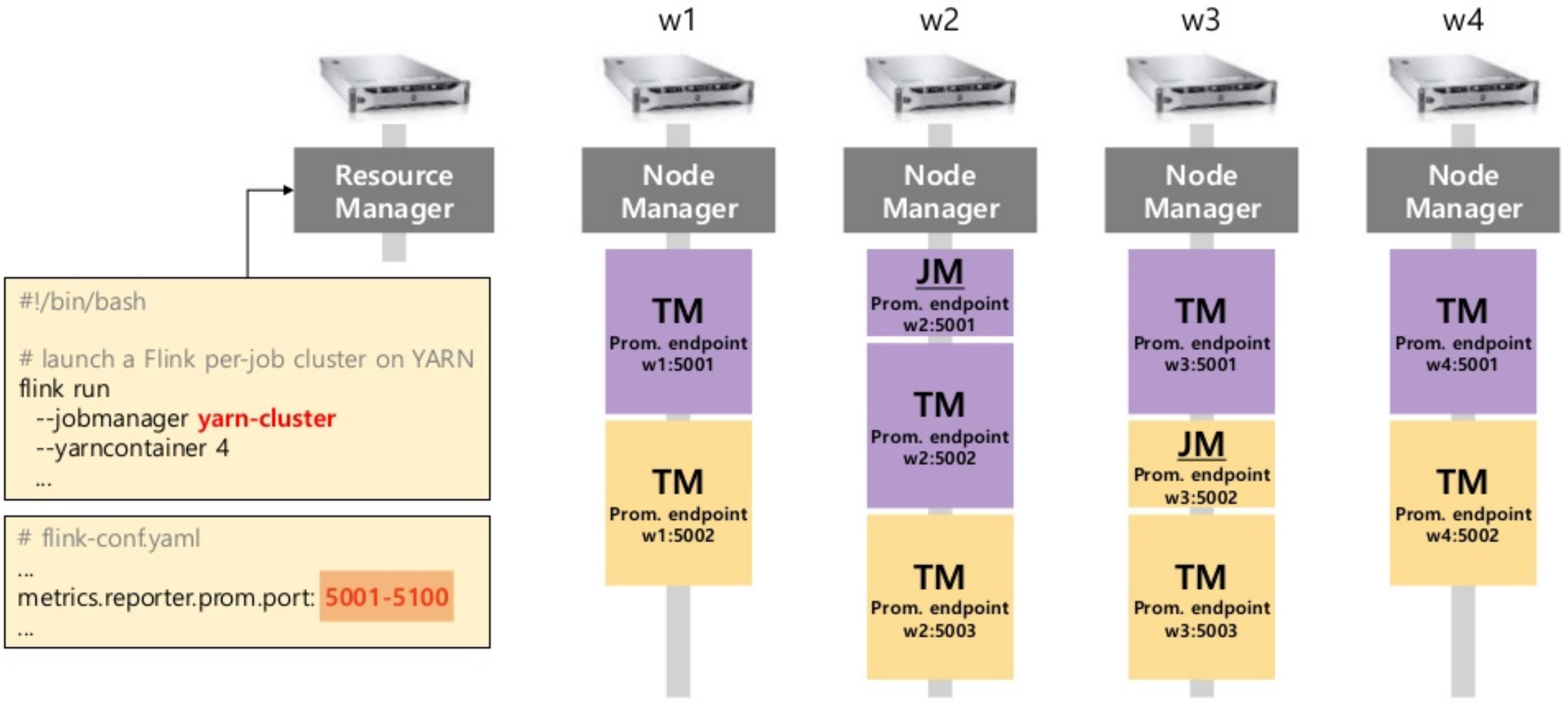
Endpoint addresses cannot be determined in advance



Endpoint addresses cannot be determined in advance

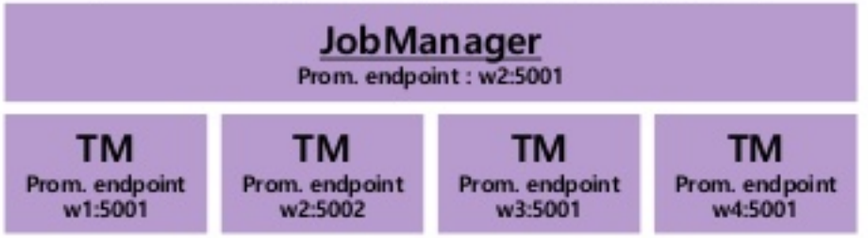


Endpoint addresses cannot be determined in advance

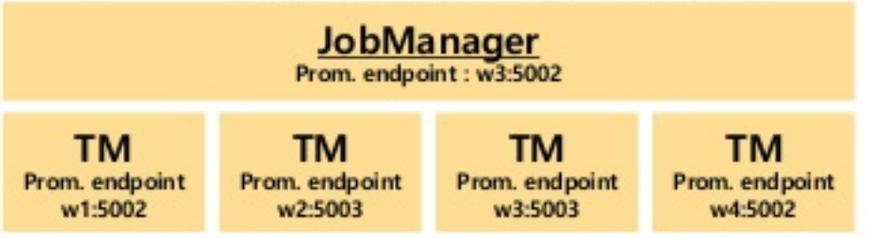


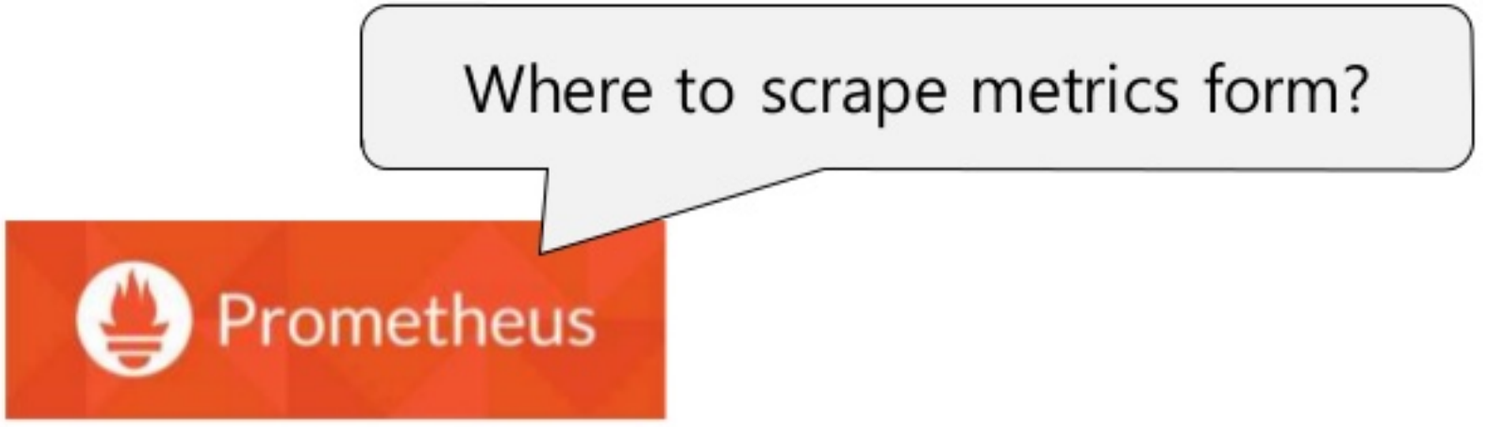
Endpoint addresses are available after a cluster is up

A per-job cluster
(YARN ID : application_1500000000000_0001)

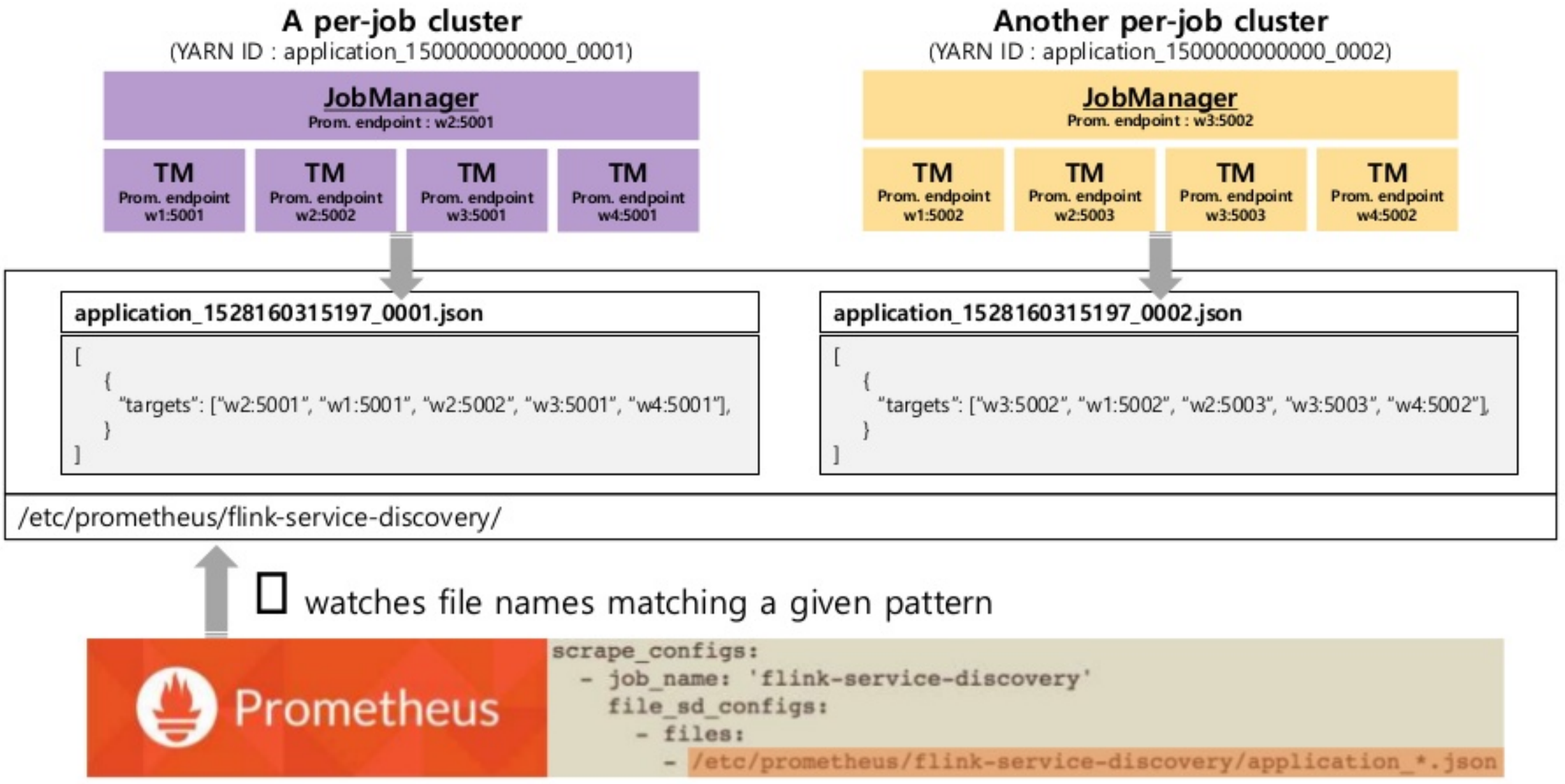


Another per-job cluster
(YARN ID : application_1500000000000_0002)

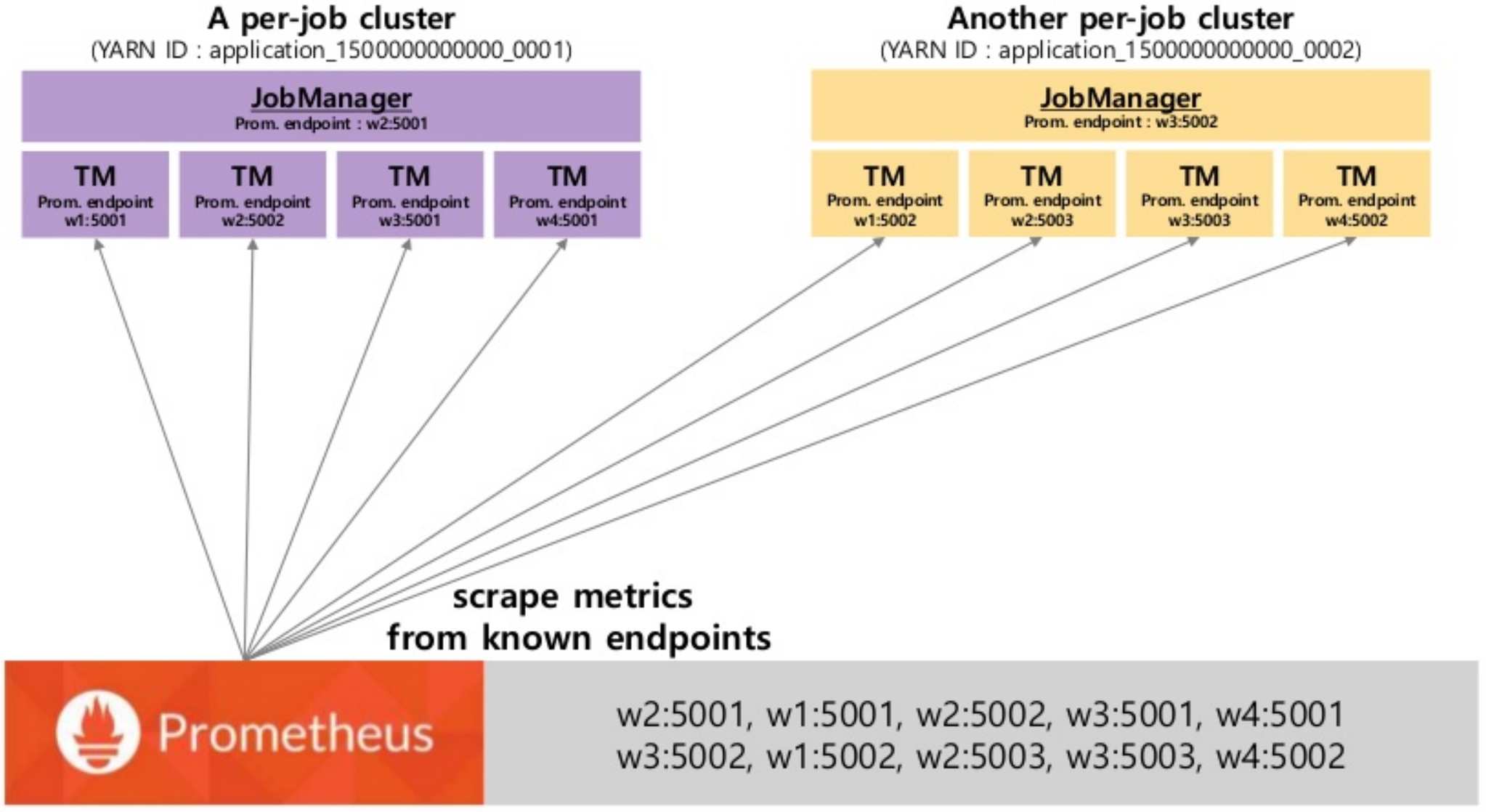




File-based service discovery mechanism

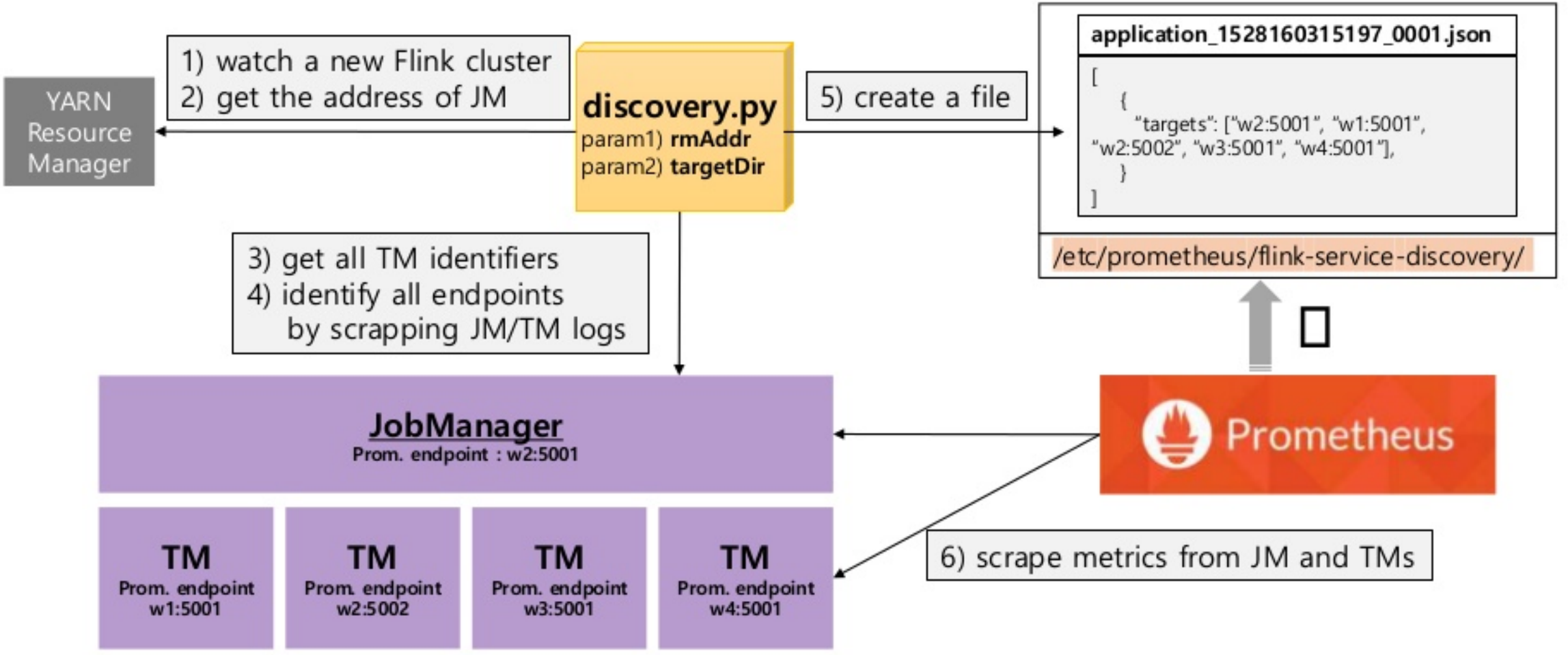


File-based service discovery

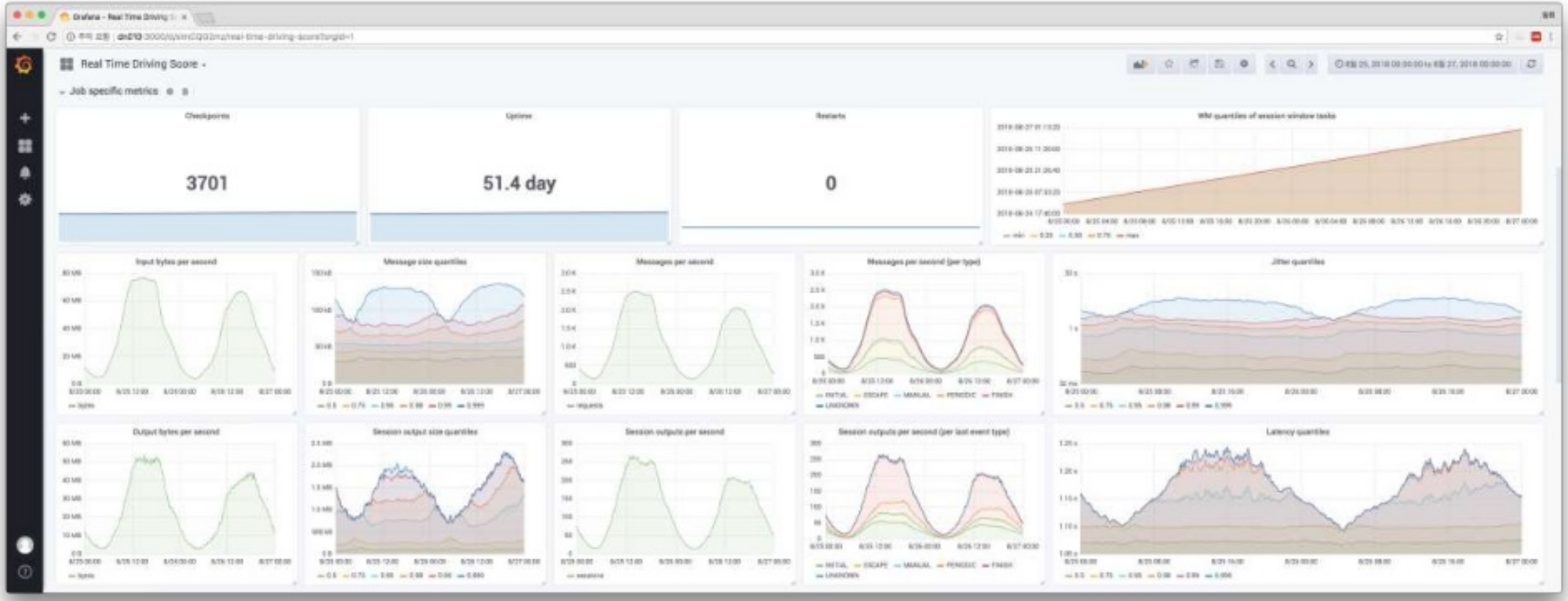


flink-service-discovery

https://github.com/eastcirclek/flink-service-discovery

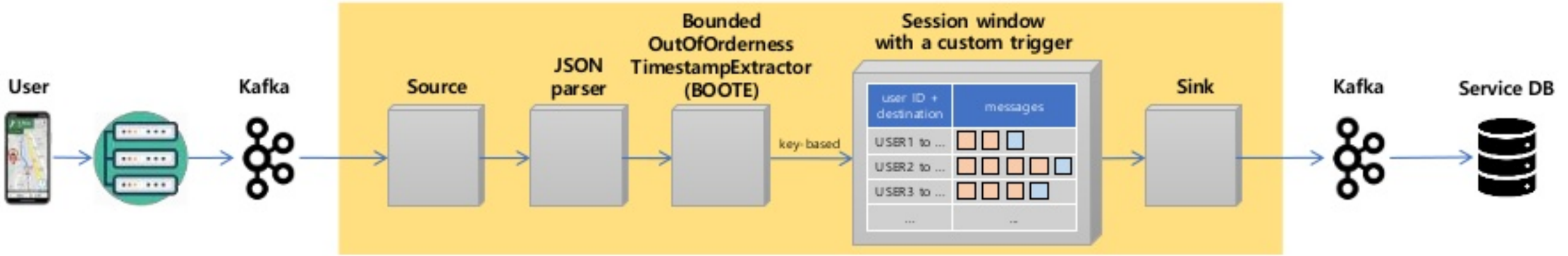


Grafana dashboard



Overview & summary

- Dataflow design and trigger customization



- Instrumentation with Prometheus

THE END