TPS

1 ~ 60 1 .

, TPS TPS

> TPS
>
> .

. 1 10TPS 10 .

~ 1 .

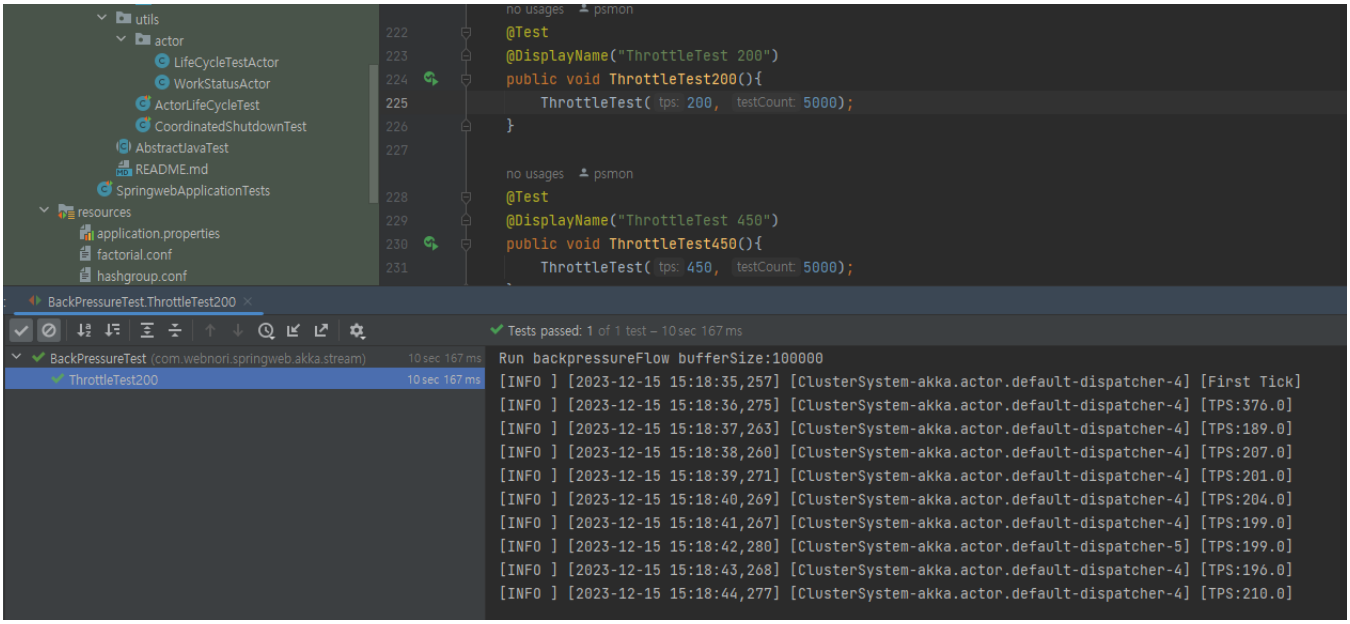
```
    private static final Executor executor = Executors.newFixedThreadPool(450);

    private static CompletionStage<String> callApiAsync(Integer param) {
        // CompletableFuture
        return CompletableFuture.supplyAsync(() -> {
            try {

                double dValue = Math.random();
                int iValue = (int)(dValue * 1000);
                Thread.sleep(iValue); // API
                tpsActor.tell("CompletedEvent", ActorRef.noSender()); // TPS
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
            return "Response for " + param;
        }, executor);
    }
```

TPS AkkaStream .

sink .

```
    public void ThrottleTest(int tps, int testCount ) {
        new TestKit(actorSystem) {
            {
                final ActorMaterializerSettings settings = ActorMaterializerSettings.create(actorSystem)
                        .withDispatcher("my-dispatcher-streamtest");

                final Materializer materializer = ActorMaterializer.create(settings, actorSystem);
                final TestKit probe = new TestKit(actorSystem);

                tpsActor = actorSystem.actorOf(TpsMeasurementActor.Props(), "TpsActor");
                tpsActor.tell(probe.getRef(), getRef());
                expectMsg(Duration.ofSeconds(1), "done");

                // Source
                Source<Integer, NotUsed> source = Source.range(1, testCount);

                //    Flow
                final int parallelism = 450;
                Flow<Integer, String, NotUsed> parallelFlow = Flow.<Integer>create()
                        .mapAsync(parallelism, BackPressureTest::callApiAsync);

                // Buffer   OverflowStrategy.backpressure
                int bufferSize = 100000;
                Flow<Integer, Integer, NotUsed> backpressureFlow = Flow.<Integer>create()
                        .buffer(bufferSize, OverflowStrategy.backpressure());

                AtomicInteger processedCount = new AtomicInteger();

                // Sink
                Sink<String, CompletionStage<Done>> sink = Sink.foreach(s -> {
                    //
                    processedCount.getAndIncrement();
                    if(processedCount.getAcquire() % 10 == 0) {
                        //System.out.println("Processed 10");
                    }
                });

                System.out.println("Run backpressureFlow bufferSize:"+bufferSize);

                // RunnableGraph
                source.via(backpressureFlow)
                        .throttle(tps, FiniteDuration.create(1, TimeUnit.SECONDS), tps, (ThrottleMode)
ThrottleMode.shaping())
                        .via(parallelFlow)
                        .to(sink)
                        .run(materializer);

                within(
                        Duration.ofSeconds(15),
                        () -> {
                            // Will wait for the rest of the 10 seconds
                            expectNoMessage(Duration.ofSeconds(10));
                            return null;
                        });
            }
        };
    }
```

```
my-dispatcher-streamtest {
  type = Dispatcher
  executor = "thread-pool-executor"
  thread-pool-executor {
    fixed-pool-size = 50
  }
  throughput = 1
}
```



TPS200 ~ .

- API TPS .

: https://github.com/psmon/java-labs/blob/master/springweb/src/test/java/com/webnori/springweb/akka/stream/BackPressureTest.java#L234

Next : - API