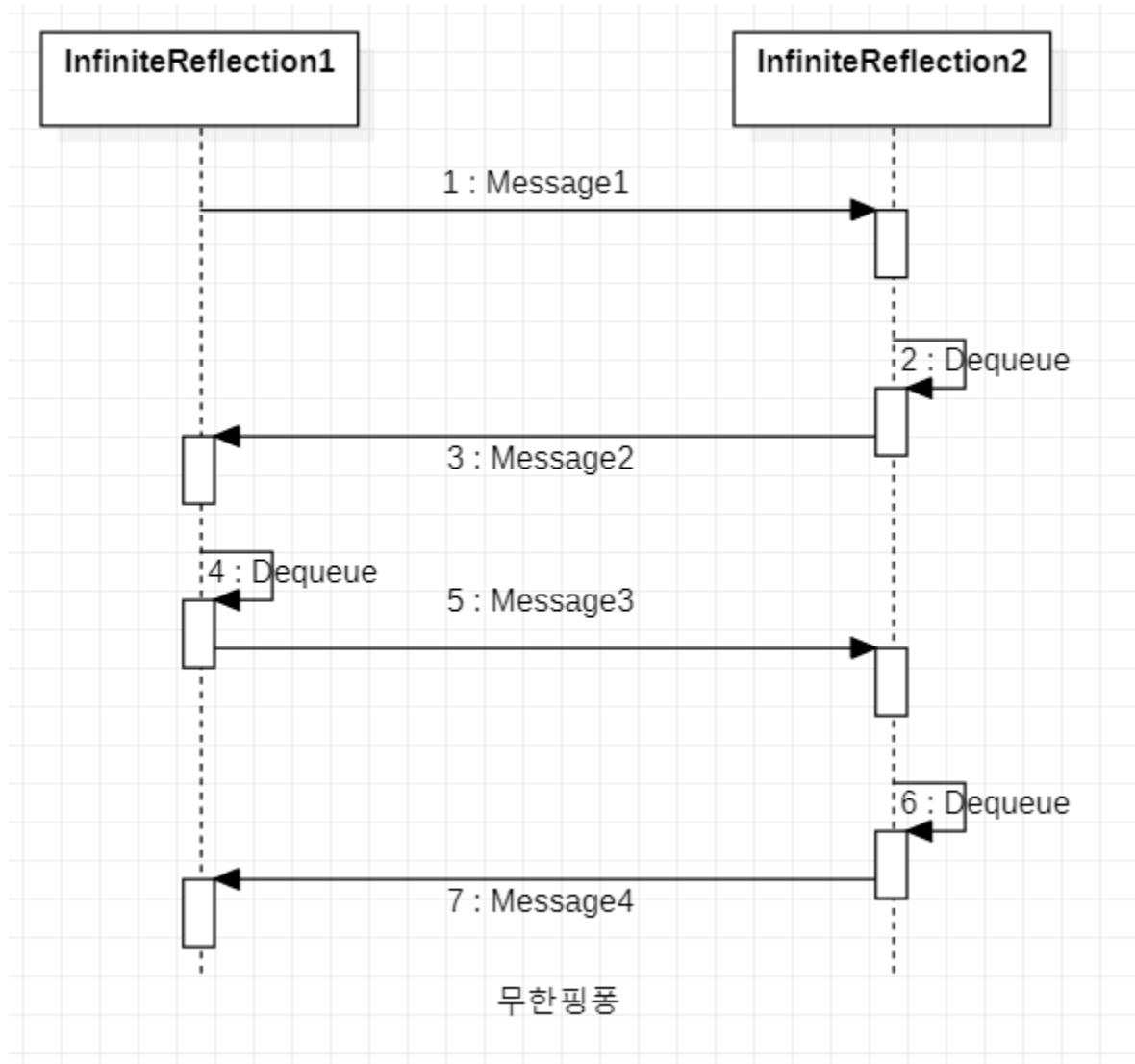




AKKA.net



- , .
- , .
- n, .

= (1) \* ()

blocked URL

() ,

TPS

```

using Akka.Actor;
using Akka.Event;
using Akka.Monitoring;

namespace AkkaDotBootApi.Actor
{
    public class InfiniteMessage
    {
        public string Message { get; set; }

        public uint Count { get; set; }
    }

    public class InfiniteReflectionActor : ReceiveActor
    {
        private IActorRef ReplyActor;

        private readonly ILoggingAdapter logger = Context.GetLogger();

        public InfiniteReflectionActor()
        {
            ReceiveAsync<IActorRef>(async actorRef =>
            {
                ReplyActor = actorRef;
            });

            ReceiveAsync<InfiniteMessage>(async infiniteMessage =>
            {
                Context.IncrementCounter("akka.infinite.metric"); // <-- , 1.
                var reply = new InfiniteMessage
                {
                    Message = infiniteMessage.Message,
                    Count = ++infiniteMessage.Count
                };

                if(reply.Count % 50000 == 0) // <-- 5 . TPS .
                {
                    logger.Info($"Count:{reply.Count}");
                }

                ReplyActor.Tell(reply); // 1 .( . - )
            });
        }
    }
}

```

```

// 
//custom-dispatcher , custom-task-dispatcher , default-fork-join-dispatcher
string disPatcher = "default-fork-join-dispatcher"; //
int pipongGroupCount = 1; // , . ( 21)
int ballCount = 6; //

// ...
for (int i=0; i < pipongGroupCount; i++)
{
    string actorFirstName = "infiniteReflectionActorA" + i;
    string actorSecondName = "infiniteReflectionActorB" + i;

    // Test Actor
    var infiniteReflectionActorA = AkkaLoad.RegisterActor(actorFirstName,
        actorSystem.ActorOf(Props.Create(() => new InfiniteReflectionActor()).WithDispatcher
(disPatcher),
        actorFirstName));

    var infiniteReflectionActorB = AkkaLoad.RegisterActor(actorSecondName,
        actorSystem.ActorOf(Props.Create(() => new InfiniteReflectionActor()).WithDispatcher
(disPatcher),
        actorSecondName));

    // ,
    infiniteReflectionActorA.Tell(infiniteReflectionActorB);
    infiniteReflectionActorB.Tell(infiniteReflectionActorA);

    //
    for(int ballIdx=0; ballIdx< ballCount; ballIdx++)
    {
        infiniteReflectionActorA.Tell(new InfiniteMessage()
        {
            Message = "A",
            Count = 0
        });
    }
}

```

```

default-fork-join-dispatcher {
    type = ForkJoinDispatcher
    throughput = 100
    dedicated-thread-pool {
        thread-count = 8
        deadlock-timeout = 3s
        threaddtype = background
    }
}

custom-dispatcher {
    type = Dispatcher
    throughput = 100
}

custom-task-dispatcher {
    type = TaskDispatcher
    throughput = 100
}

```

? TPL ? forkJoin .

: <https://getakka.net/articles/actors/dispatchers.html>

## AKKA.net Datadog

```
using Akka.Monitoring.Datadog;
using StatsdClient;
using Akka.Monitoring;

.....
    // 
    var statsdConfig = new StatsdConfig
    {
        StatsdServerName = "127.0.0.1"
    };
    ActorMonitoringExtension.RegisterMonitor(actorSystem, new ActorDatadogMonitor(statsdConfig));
```

Datadog Agent , Actor

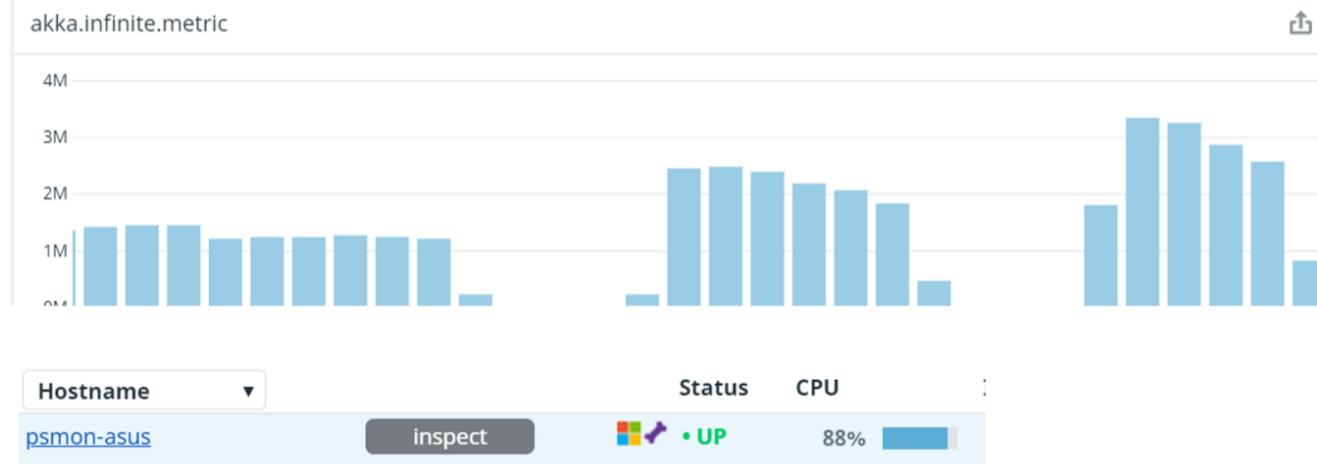
IncrementCounter .  
: Real time performance counters

The screenshot shows the Datadog Metrics Explorer interface. On the left is a sidebar with icons for Watchdog, Events, Dashboards, Infrastructure, Monitors, Metrics (which is selected and highlighted in blue), Integrations, APM, and Notebooks. The main area is titled 'Metrics Explorer' and contains a search bar with the text 'akka.infinite.metric'. Below the search bar is a list of metric names, with 'akka.actor.created' being the top item and highlighted in blue, indicating it is currently selected or the focus of the search.

Metric Name
akka.actor.created
akka.global.actor.created
akka.global.infinite.metric
akka.global.logging.info
akka.global.messages.deadletters
datadog.agent.python.version
datadog.agent.running
datadog.agent.started
datadog.dogstatsd.client.bytes_dropped
datadog.dogstatsd.client.bytes_dropped_queue

TPS( 30.)

M =

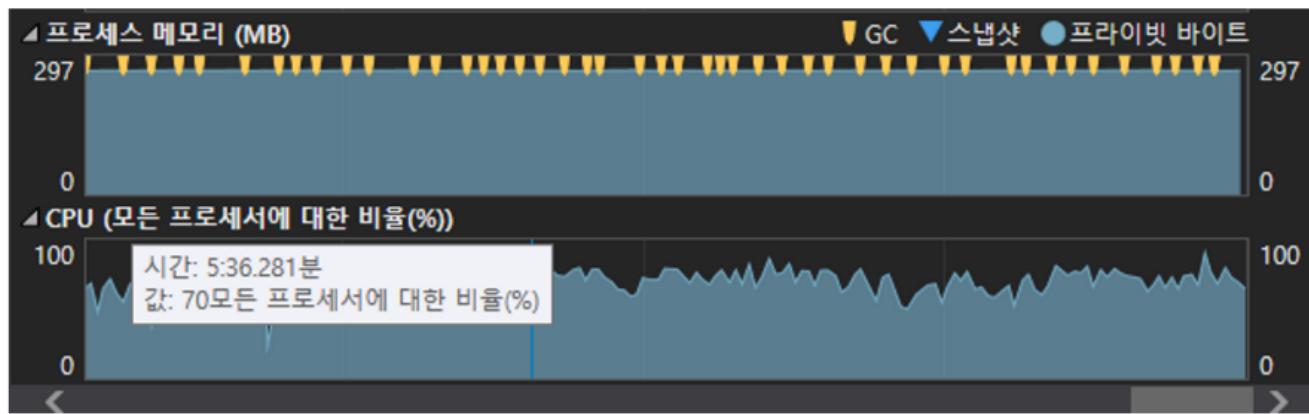


: 3 / 30 , 10/sec

## CPU

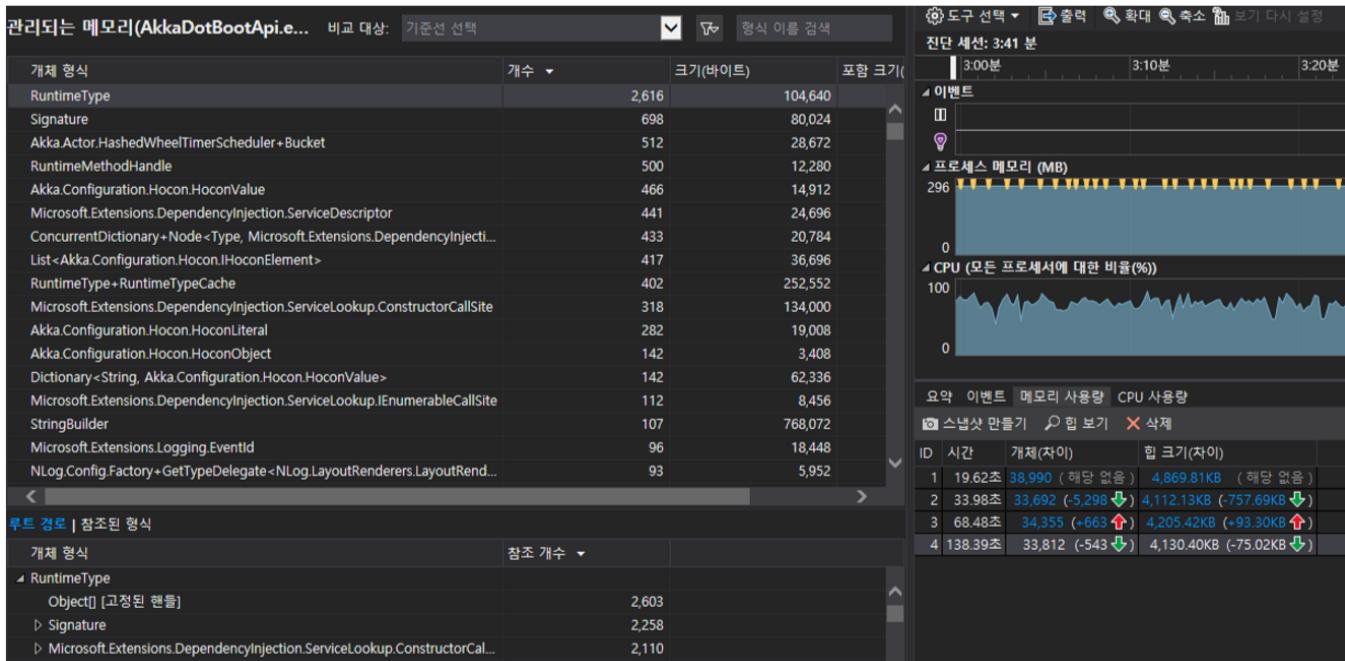


CPU



CPU , GC

GC .



- ?
- ?
- ?

- 10
- 4~6
- GC ,

## High Performance

Up to 50 million msg/sec on a single machine. Small memory footprint; ~2.5 million actors per GB of heap.



5 .

.( , 10 .)

git : <https://github.com/psmon/AkkaDotModule/blob/master/AkkaDotBootApi/Actor/InfiniteReflectionActor.cs>

: 30

: M ()

1 : : 2.17

akka.infinite.metric



Hostname ▾

psmon-asus

inspect

Status



CPU

88%



2 : : 2.18

akka.global.infinite.metric



6M

4M

2M

0M

01:07 01:08 01:09 01:10 01:11 01:12 01:13 01:14 01:15 01:16 01:17 01:18 01:19 01:20 01:21