Distributed Monitoring and Analysis for Telecom Requirements

Yuki Kasuya <yu-kasuya@kddi-research.jp>
KDDI Research

Ryota Mibu <r-mibu@cq.jp.nec.com>
NEC

Tomofumi Hayashi <tohayash@redhat.com>
RedHat

| Research Inc. Proprietary and Confidential

Members

Agenda

1	Goal
2	Requirements / Challenges
3	Solution / Architecture
4	Feasible Study
5	Conclusion

1. Goal

KDDI Research Inc. Proprietary and Confidential

Telecom Operation Goal

- High Quality Network
  - High Speed
  - Low Latency
  - Not Congested
  - Wide Area
  - etc...
- NO Network Outage

How to Operate NFV

- Complex
  - Massive Component
  - Virtualisation Layer

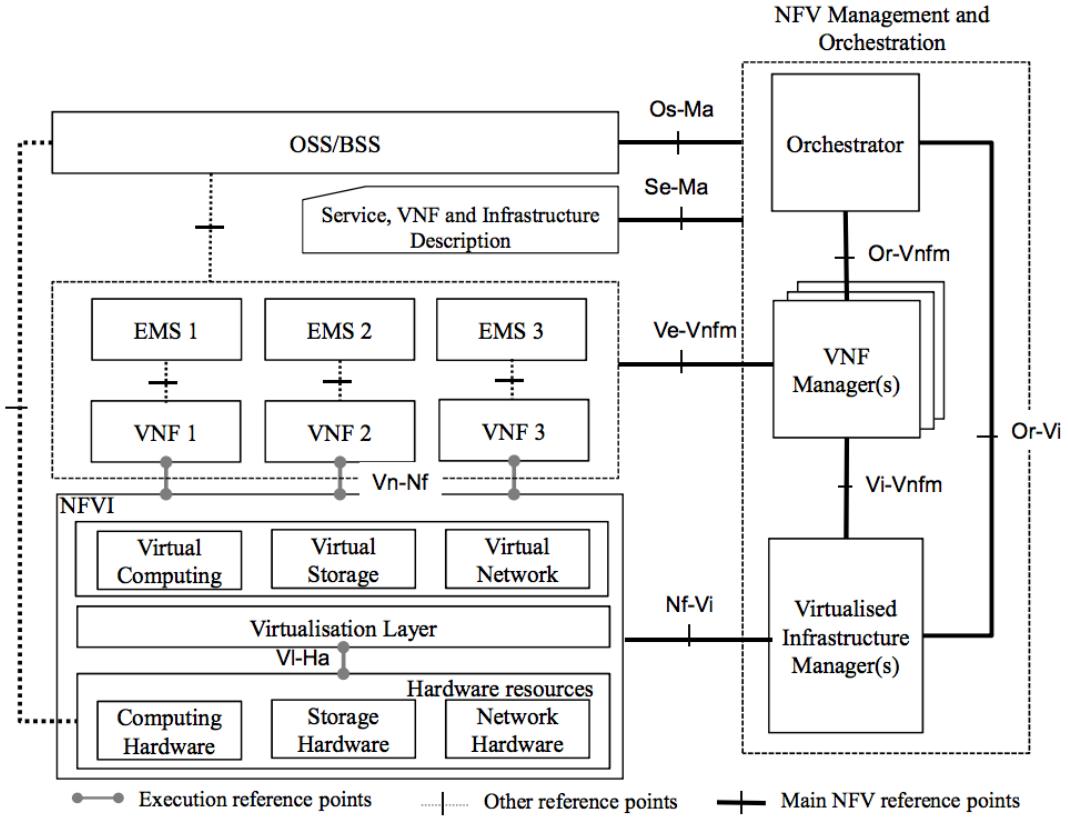
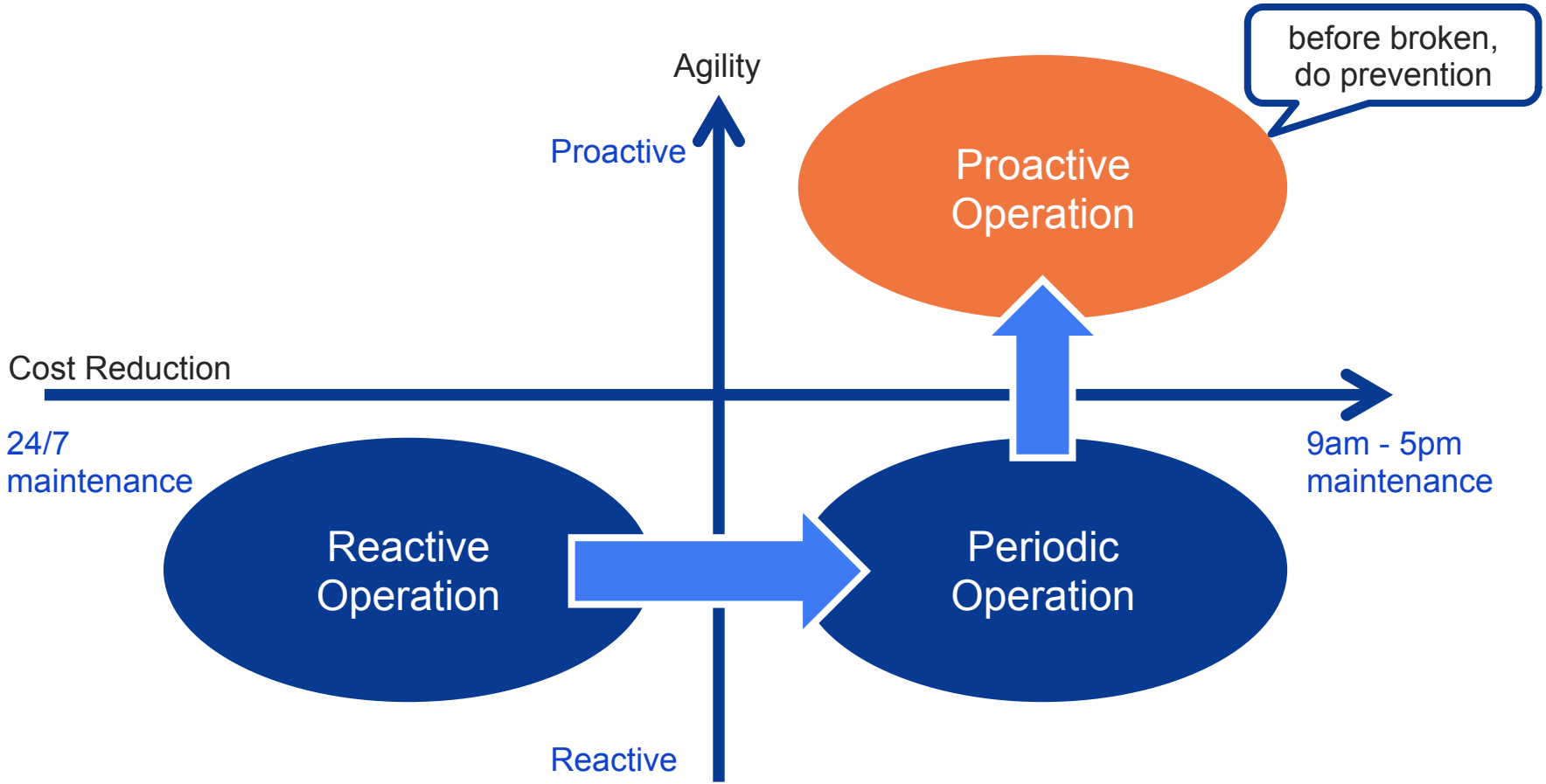


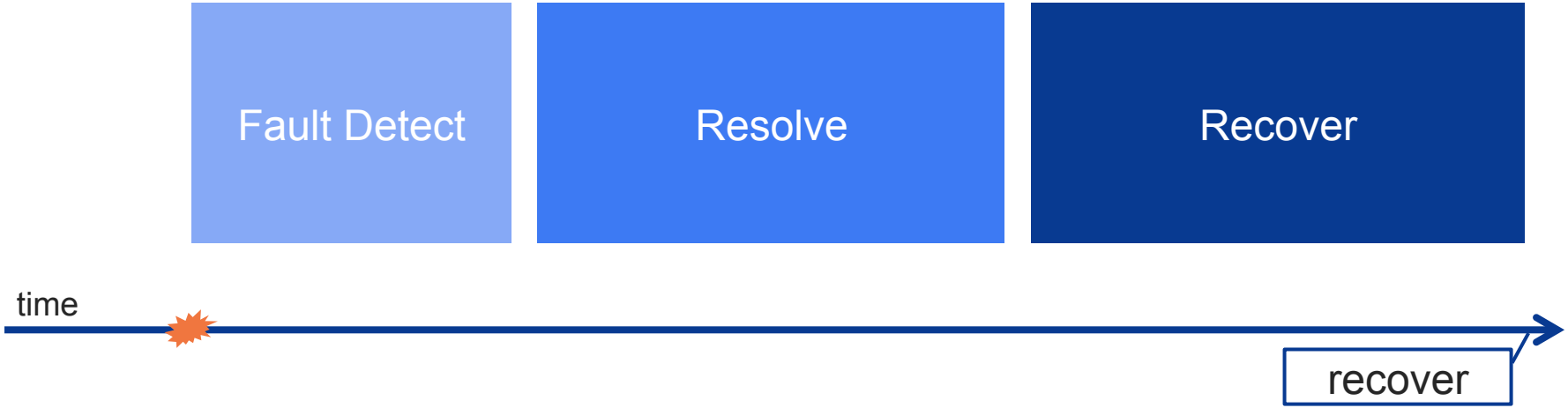
Figure 4: NFV reference architectural framework

Changing Operation Style



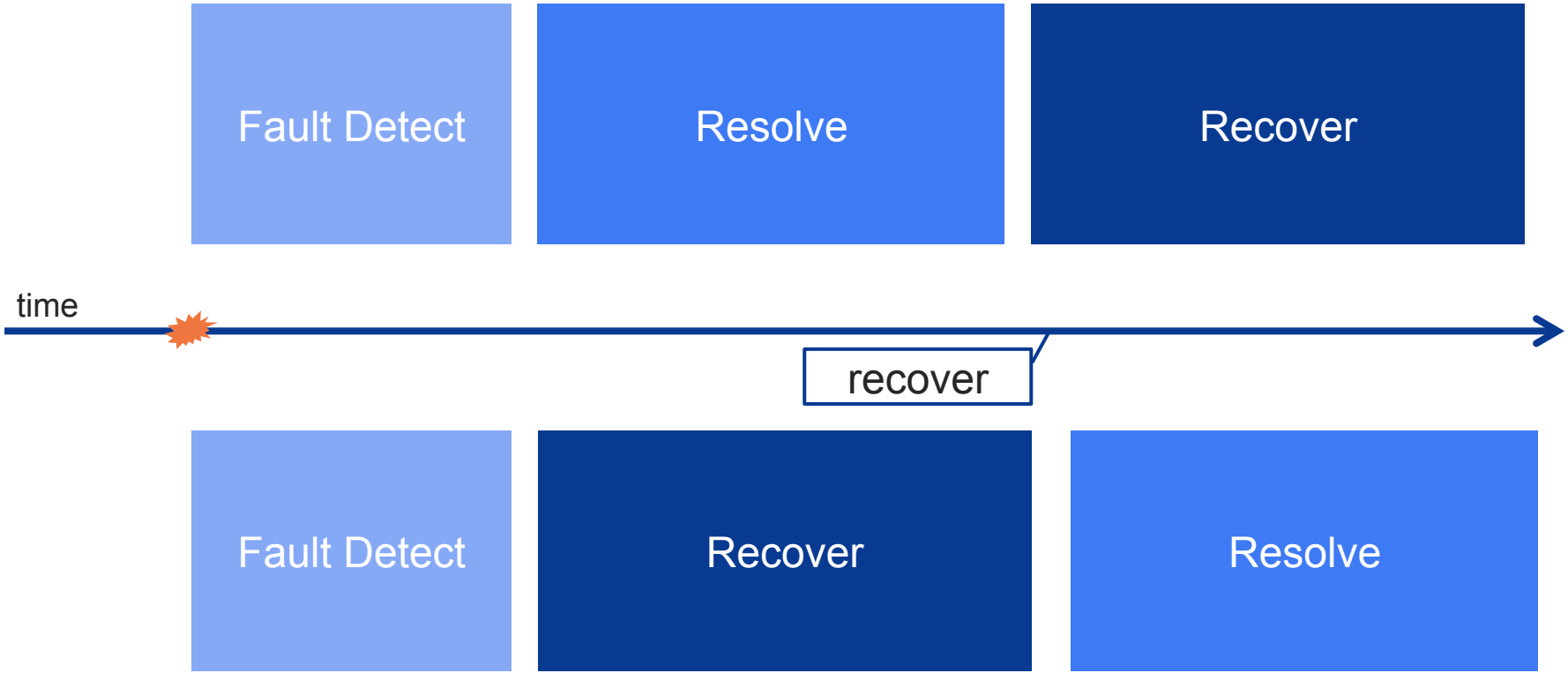
3 Steps for Proactive Operation

Reactive



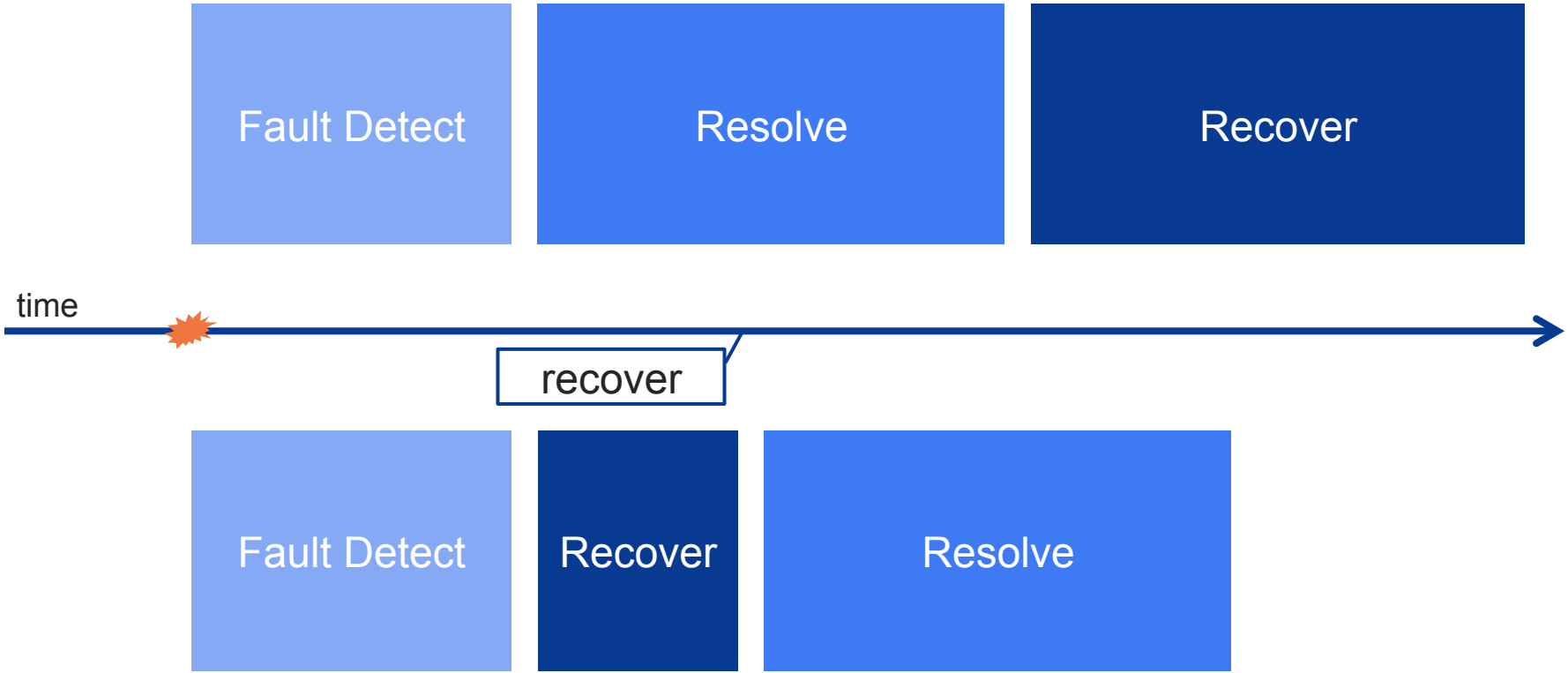
1st Step

Reactive

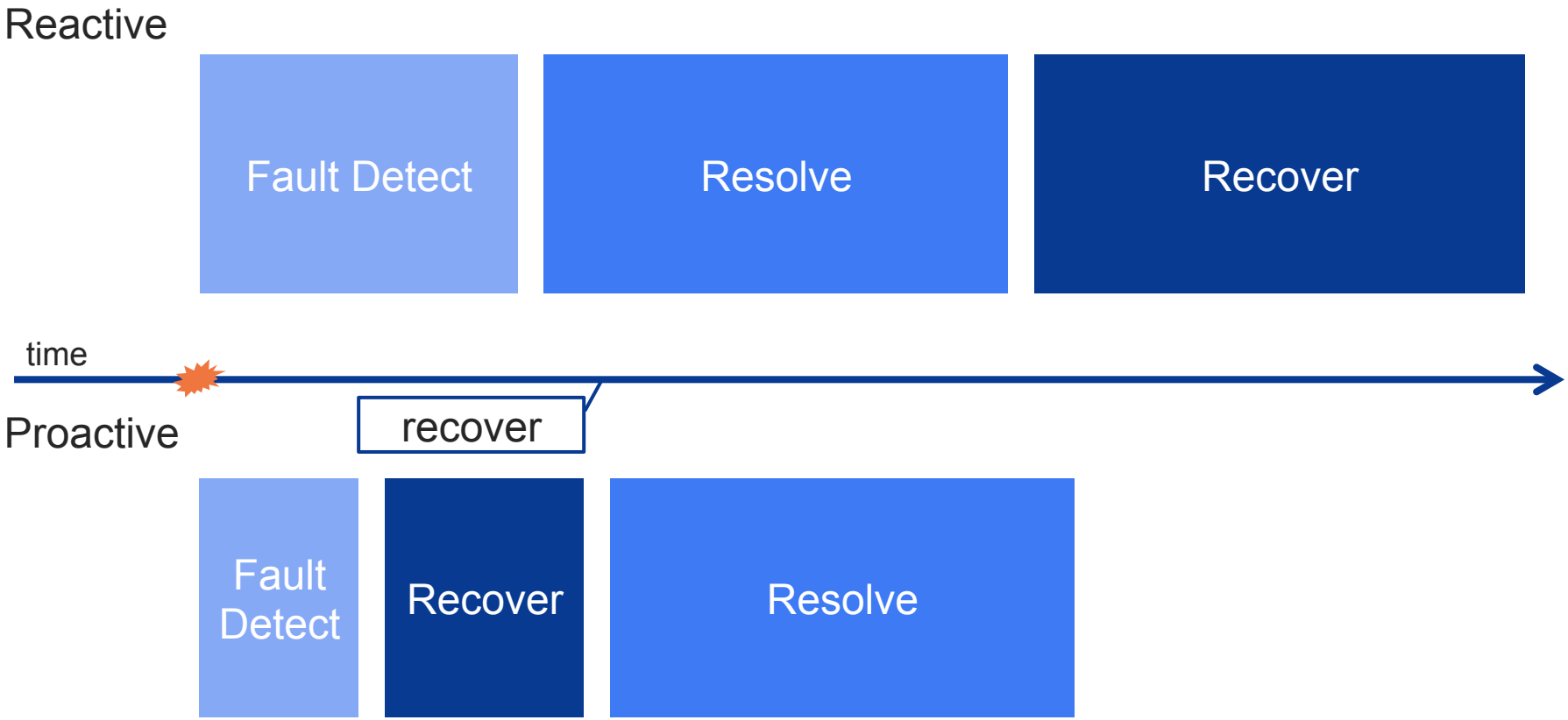


2nd Step

Reactive



3rd Step


Reactive
Fault Detect
Resolve
Recover
time
Proactive
recover
Fault Detect
Recover
Resolve

Proactive Operation

Reactive

time

Proactive

2. Requirements / Challenges

KDDI Research Inc. Proprietary and Confidential

NFV Applications

- Remote Access
  - L2TP/IPSec, SSL-VPN
- Security
  - Firewall, URL Filtering, Antivirus, WAF, DPI, IPS/IDS
- Storage
  - NAS
- Traffic Control
  - DNS, Wi-Fi Control, LB
- And More...
  - IP-PBX, Traffic Optimizer, Content Cache, Content Compression

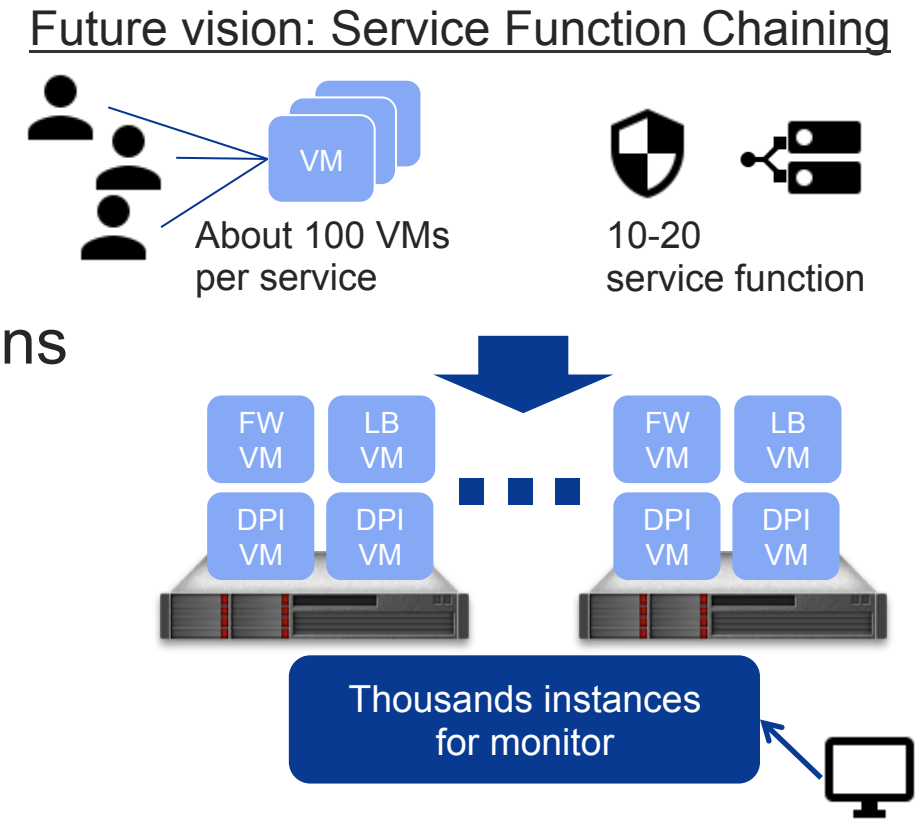
Requirements for Telecom's Proactive Operation

We need to adopt more advanced monitoring

- Increase Monitoring Targets
- Increase Monitoring Items
- Shorten Monitoring Interval

Increase Monitoring Targets

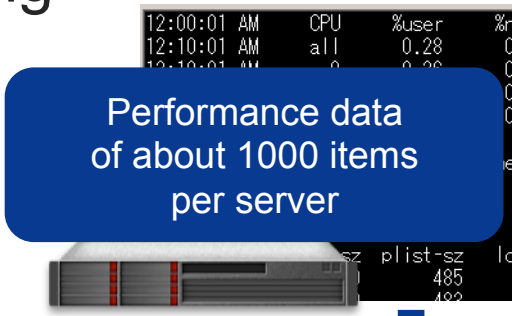
- Network functions (Firewall, DPI, LB, …) are provided as VMs
- Service function VMs are divided for various reasons
  - Performance
  - Security
  - APL Configuration
- Telecom needs to monitor a large number of targets
  - 1000 – 10000 VMs



Increase Monitoring Items

- **General cloud monitoring**
  - detect server stop
- **Telecom's monitoring**
  - detect silent failure
    ⇒ Novel approaches like AI
  - We want to use
    a large quantity data
    - About 100 - 1000 data /1 server
    - Performance
      - CPU, Memory, Load, Swap, Network, Disk IO, ...
    - Fault
      - Alarm, Error message, ...

Example:
NEC's MasterScope Invariant Analyzer

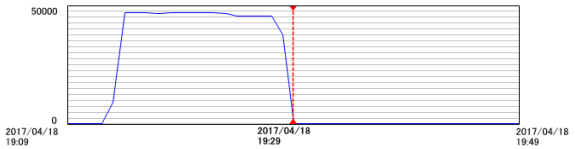


Performance data of about 1000 items per server

Learning

CPU is used for network processing

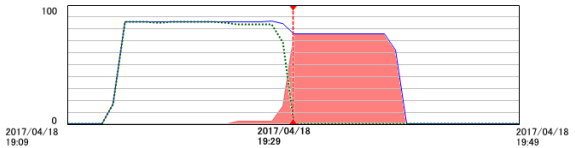
NW bandwidth



Analysis

CPU util is still high without network

CPU util

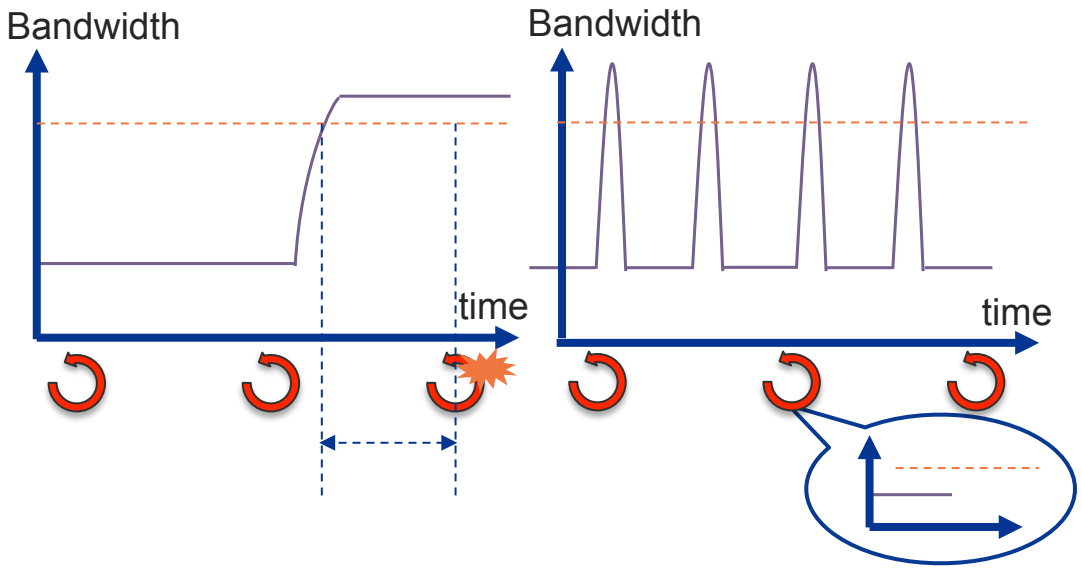


The more data, the more unknown failure AI can discover

Shorten Monitoring Interval

- **Existing data for metering** (Ceilometer etc.)
  - Collect data every few minutes
- **Telecom high availability**
  - Service stop should be less than 1 second
  - We aim for fault detection in 0.1 second
- **Metering data does not suit the purpose**

If we use metering data for monitoring…



Recovery start is delayed

Fault detection is missed

Challenges

- We need a architecture capable of collecting a large amount of data

1000-10000 targets | 100-1000 items | 0.1 second interval

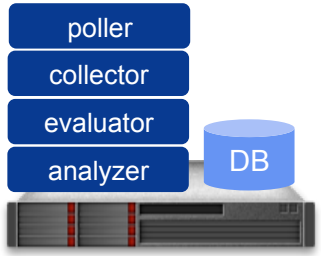
- How do you collect these data?

3. Solution / Architecture

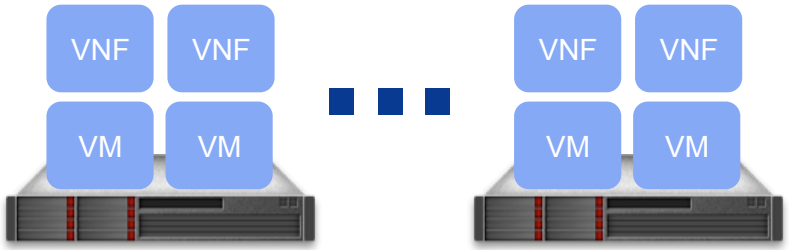
KDDI Research Inc. Proprietary and Confidential

Limitation Monitoring Function

- 1 Monitoring Function
  - 1000 Targets - 10000 Targets
  - * Items
  - * Interval



- Reduce Targets
  - 1 Monitoring Function manage



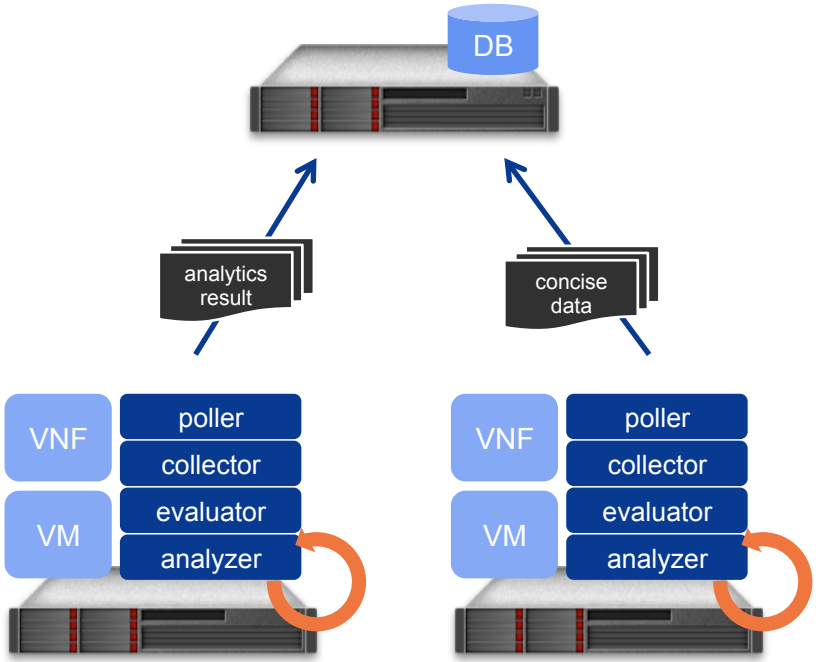
DMA (Distributed Monitoring and Analysis)

Each Computing has Its Monitoring Function

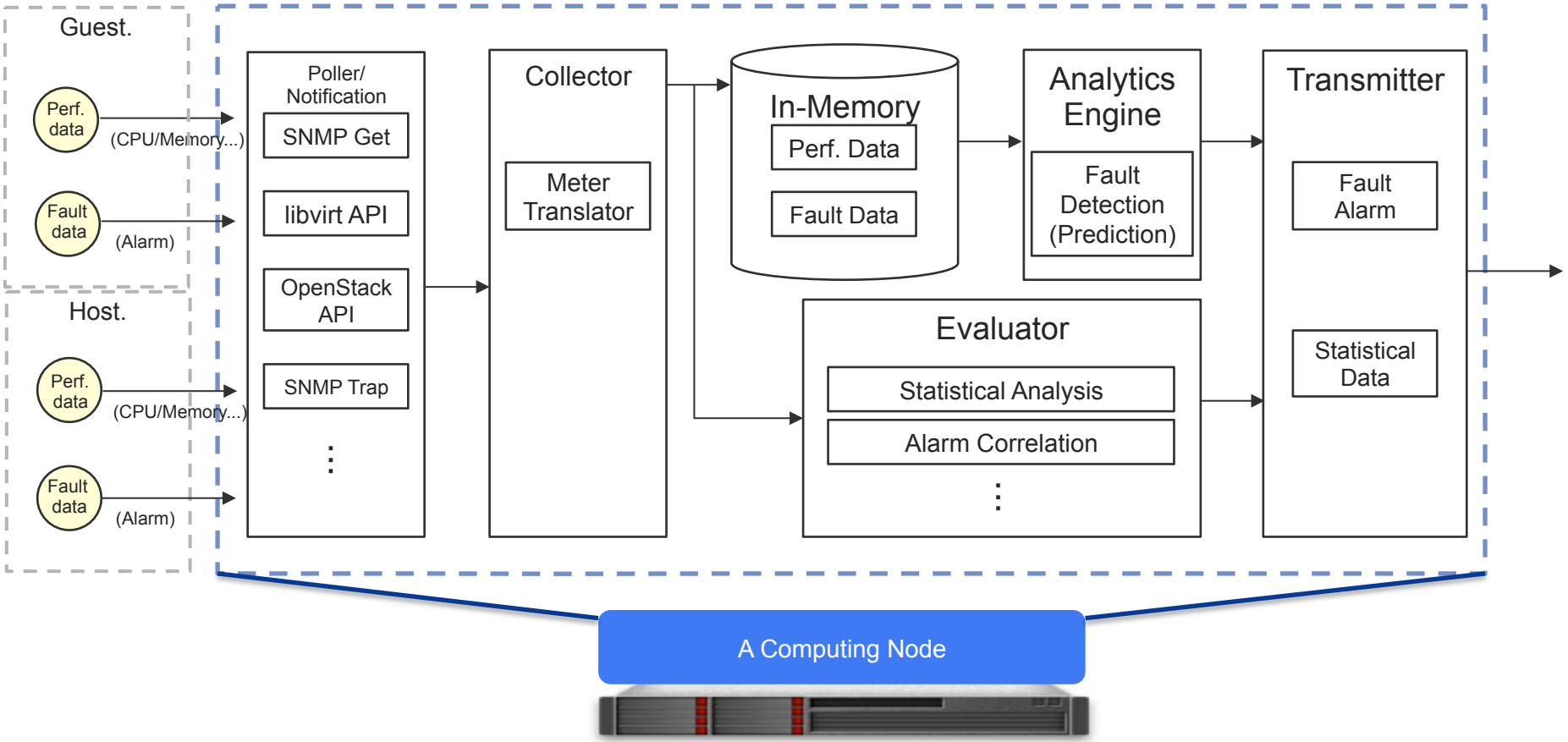
- Monitoring process is complete in each computing node

What is Good?

- Real-time(Fine-grained) monitoring
- Less Load
- Scalable

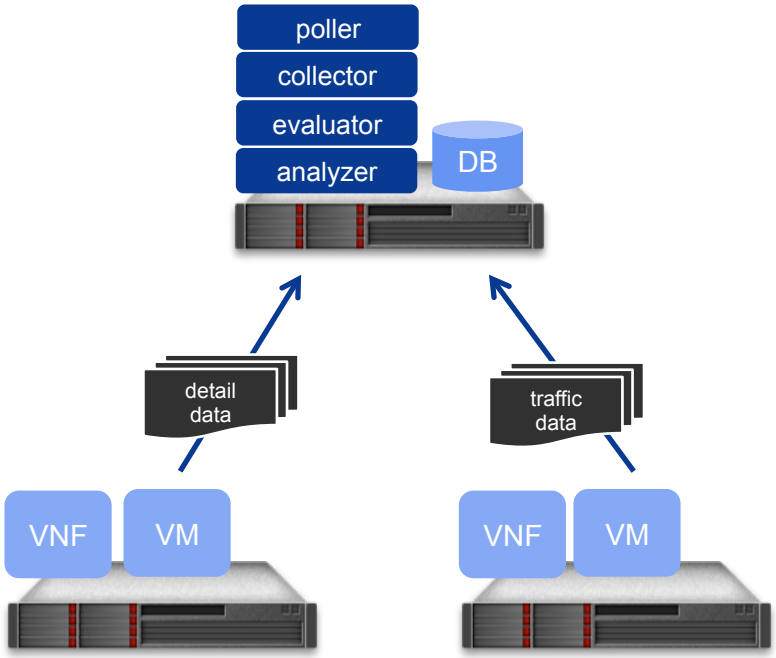


DMA Architecture Detail



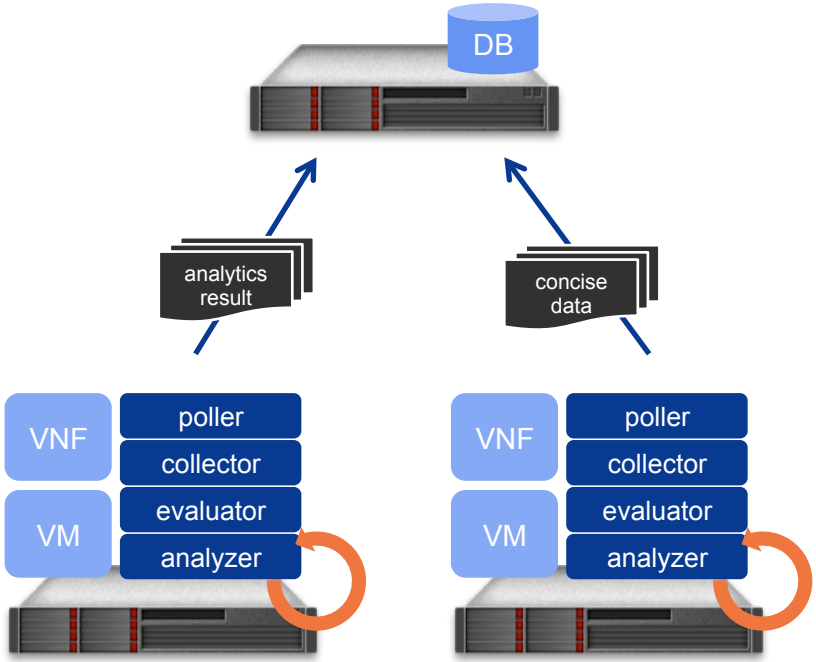
Total Architecture

Data Analysis(ceilometer)



&

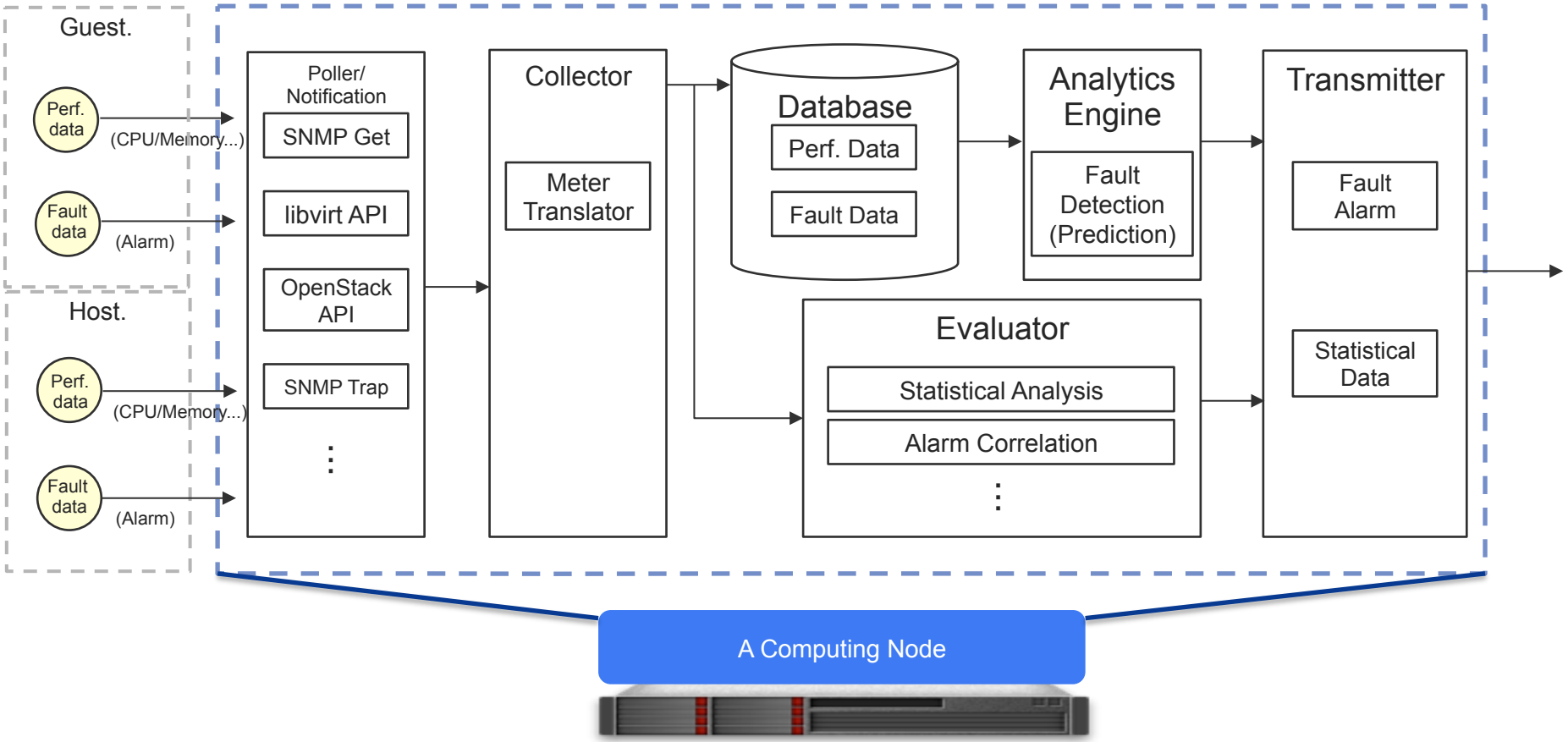
Fault Management(DMA)



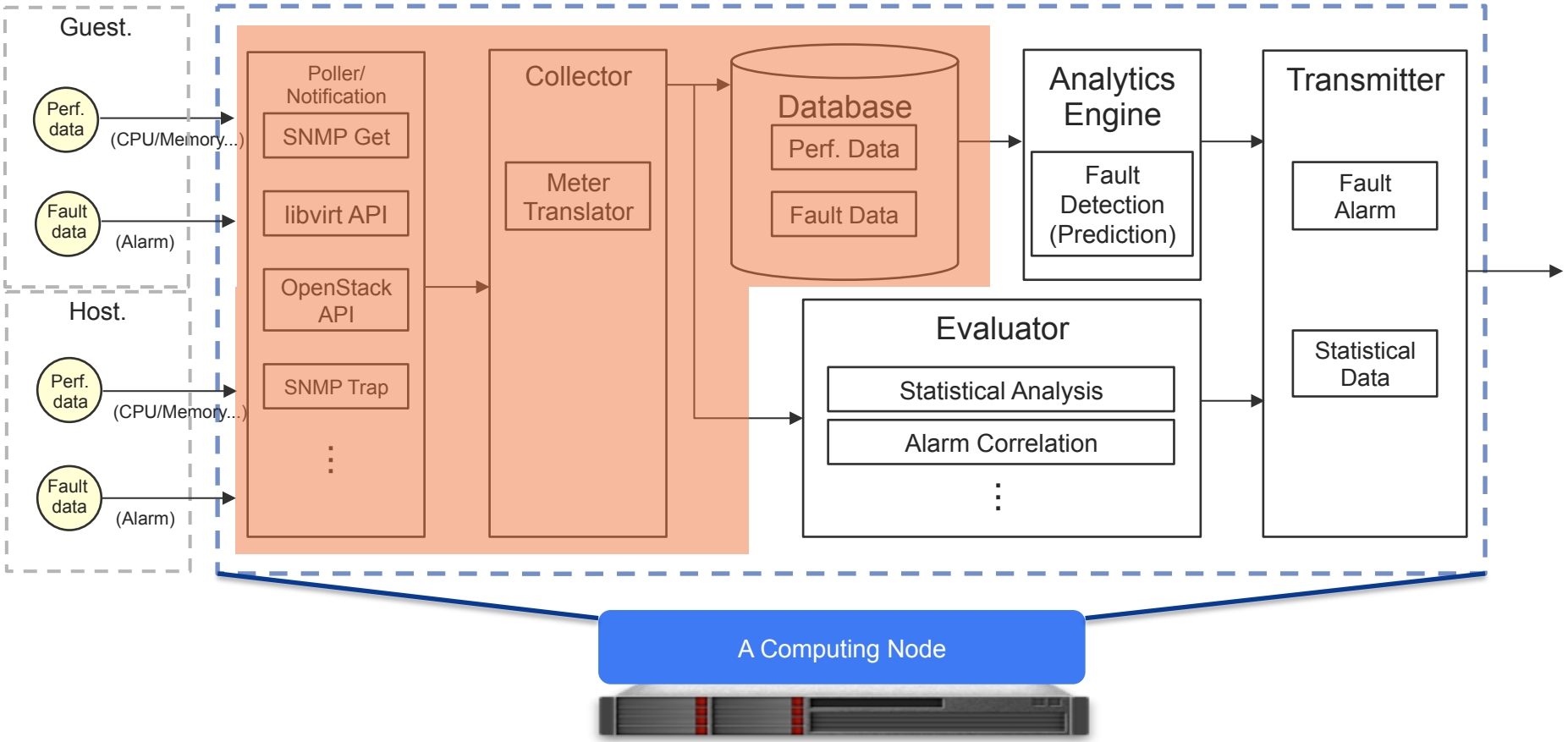
4. Feasible Study

KDDI Research Inc. Proprietary and Confidential

DMA Architecture Detail

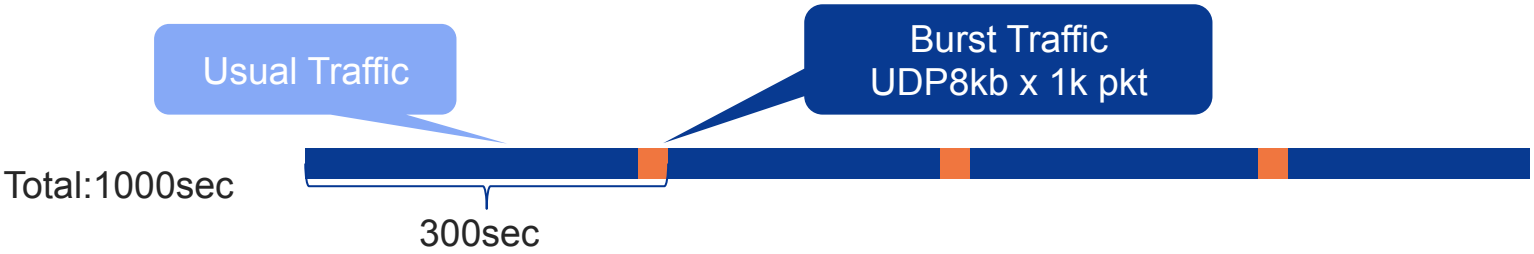


DMA Architecture Detail



What is the Goal?

- The Goal: Identify which framework can be suitable/feasible to our requirement on statistics collection
  - Scenario: Detect micro burst traffic?



  - Key Metric: Less performance impact on scaling situation (VM and node)
    - CPU Usage on node
- Target Framework
  - Ceilometer
  - Collectd
  - Monasca

Equipments

OpenStack

- Newton (RDO)
  - CentOS 7.3
  - MariaDB 10.1.20-1.el7
  - RabbitMQ
- Controller x 1
  - CPU: Xeon E3-1230v5 3.40GHz 8core
  - Memory: 32G
  - NIC x 2 (IPMI,Mgmt / API,Tenant)
- Compute x 4
  - CPU: Xeon E5-2420/E31240/X3450
  - Memory: 24G/32G
  - NIC x 2 (IPMI,Mgmt / API,Tenant)

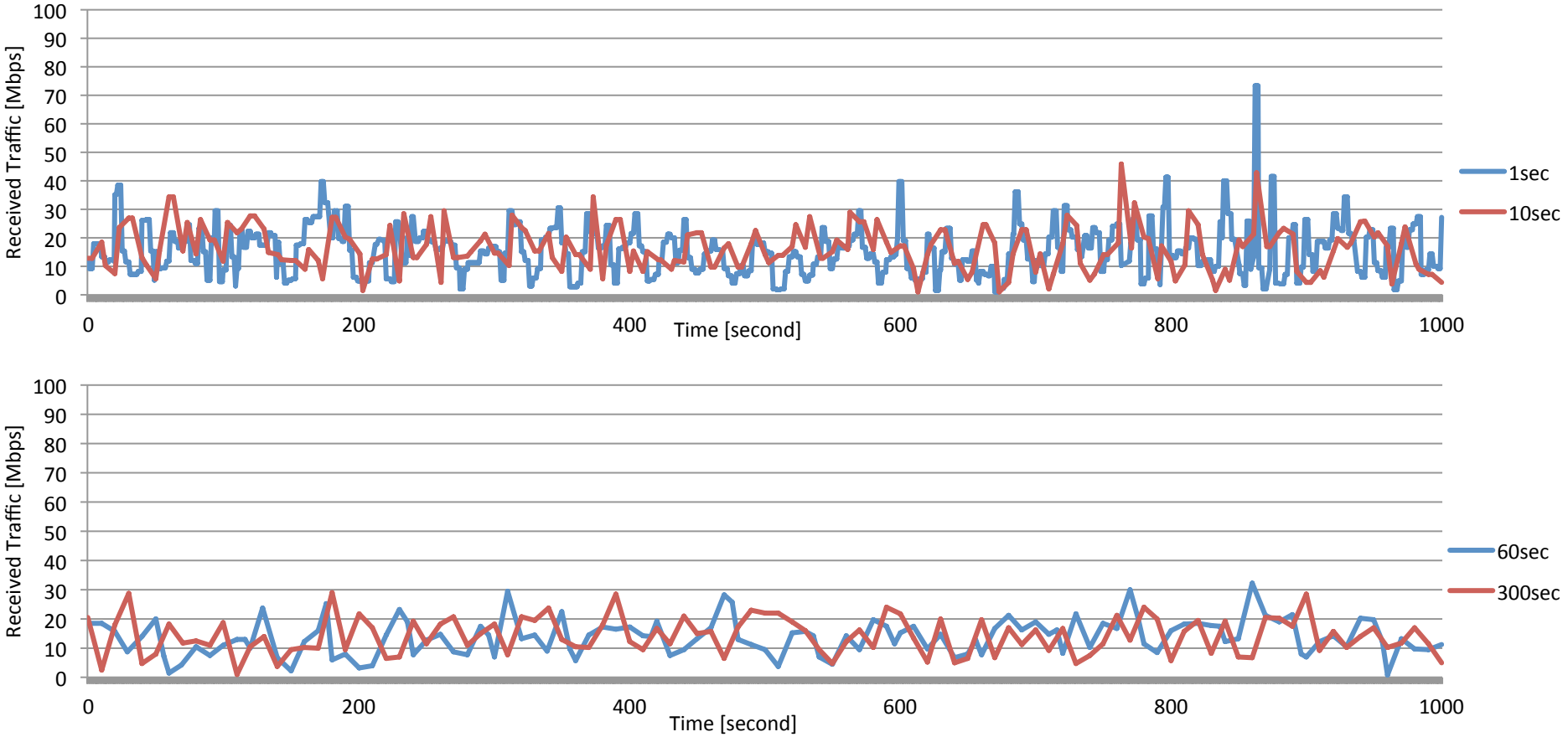
Target Framework

- Ceilometer 7.0.2
  - Output: Gnocchi 3.0.5
- Collectd
  - Output: Redis 3.2.8-1.el7
- Monasca 1.3.1
  - Agents are in compute node
  - Metrics DB: Influxdb 0.9.5-1

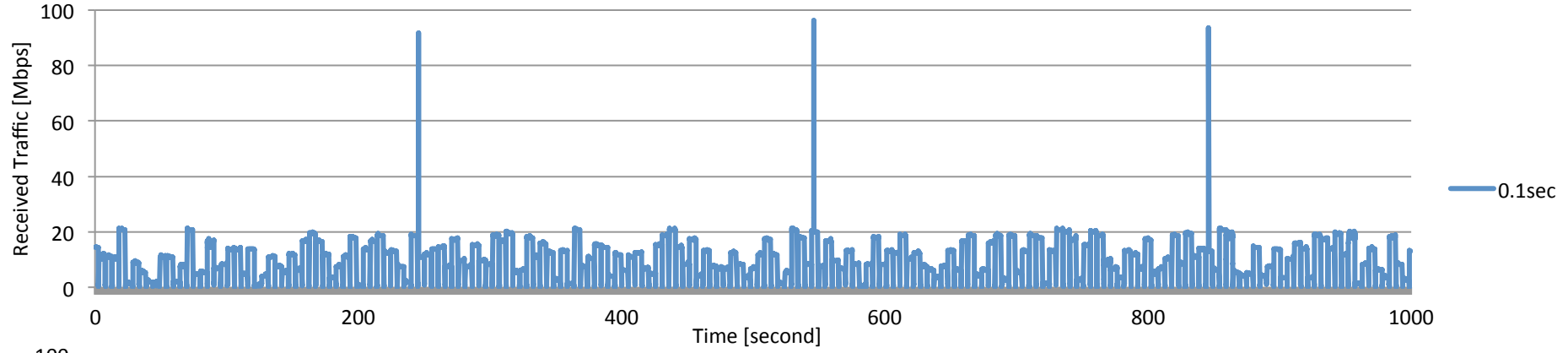
The Study Details

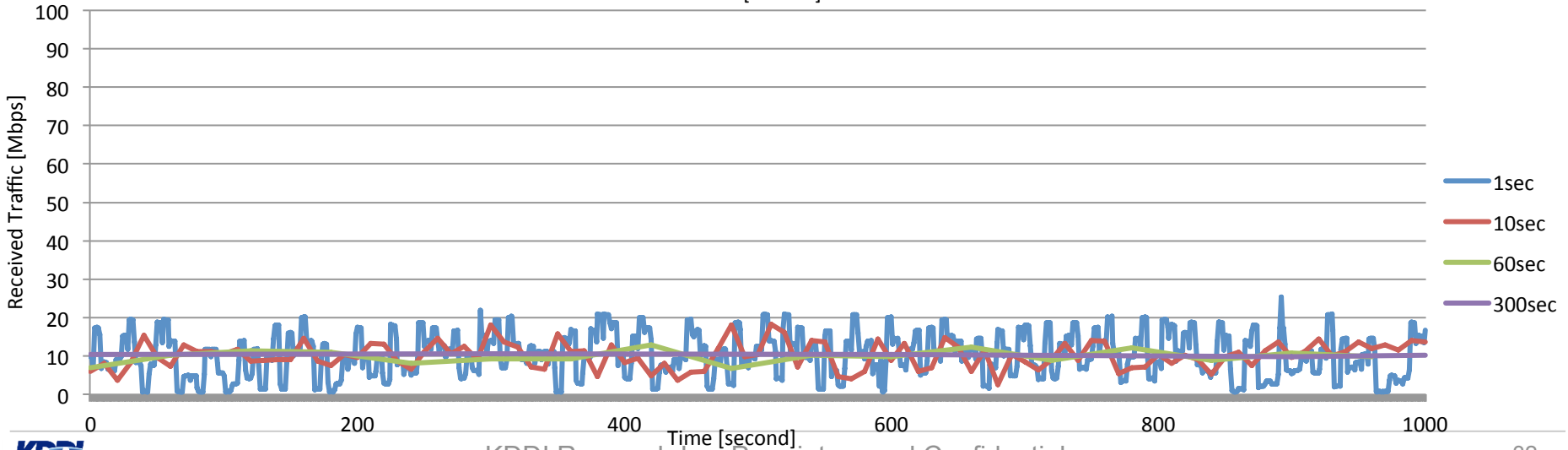
- Environment
  - Micro burst traffic case (one VM in node does huge traffic in 1sec every 1min)
- Examination
  - Can Framework detect microburst?
  - How much CPU does the framework use?
- Target Process
  - Ceilometer
    - api, collector, notification, polling, polling-central, gnocchi-api, gnocchi-metricd, gnocchi-statsd
  - Collectd
    - collectd, redis
  - Monasca
    - api, collector, forwarder, kafka, persister, storm-supervisor, supervisord, zookeeper
- Parameters
  - Interval to get metrics: 300 sec, 60 sec, 10 sec, 1 sec, 0.1 sec (collectd supports 0.1sec interval)

Ceilometer (Rx Traffic)

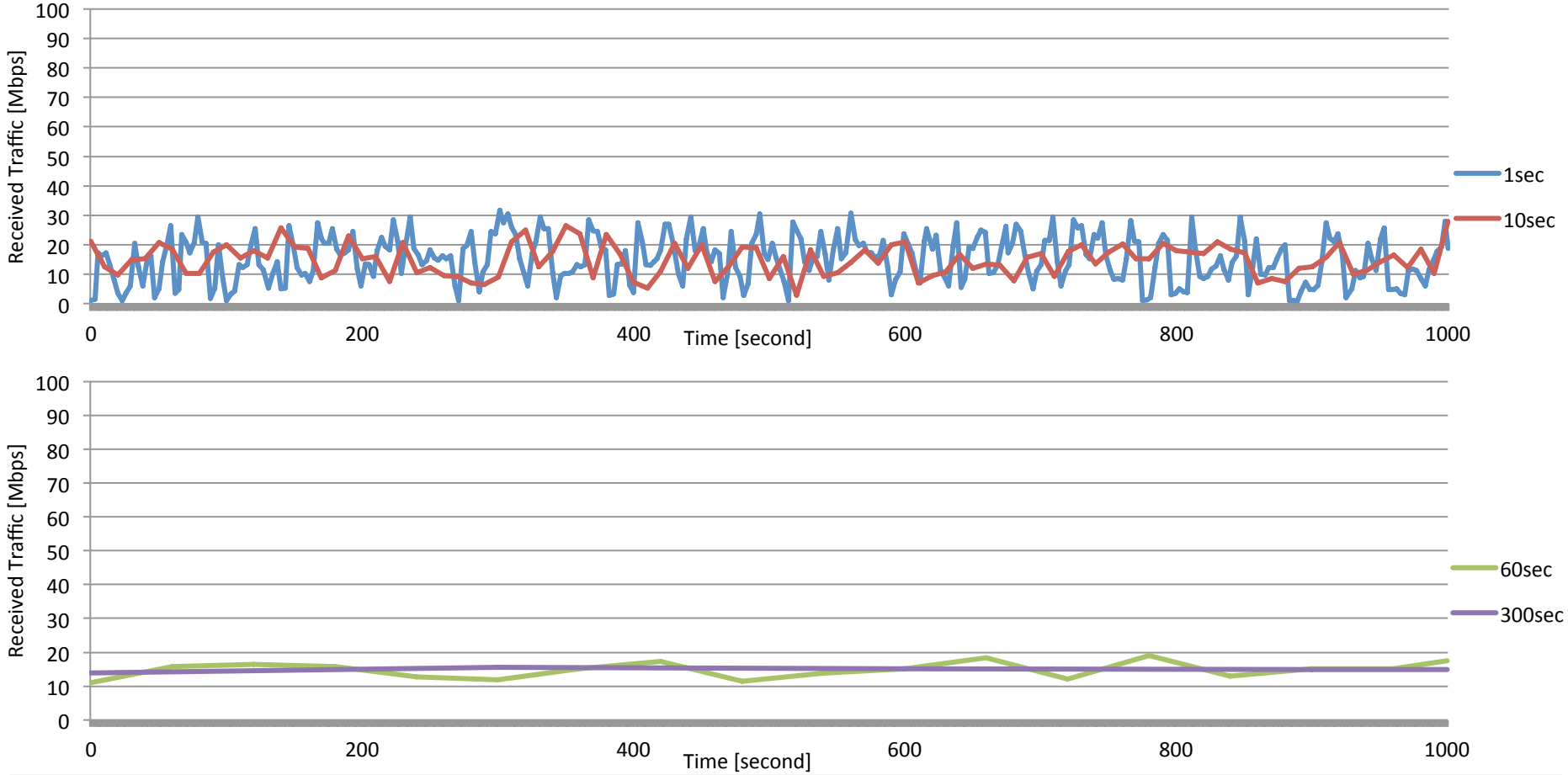


Collectd (Rx Traffic)





Monasca (Rx Traffic)



CPU Usage (Sum of each processes' CPU%)

		Sum of CPU%			
		min	max	mean	median
Ceilometer	1sec	16	493	234.4429371	226
	10sec	11	286	65.4810989	24
	60sec	10	409.32	48.07490509	18
	300sec	9	479	44.21182817	17
collectd	0.1sec	3.97	121	30.30854146	7
	1sec	0	104	8.273686314	1
	10sec	0	104	5.763236763	0
	60sec	0	105	5.66033966	0
	300sec	0	101	6.524475524	0
monasca(controller)	1sec	0	30	3.424285714	3
	10sec	0	29	3.29953047	3
	60sec	0	64	3.554995005	3
	300sec	0	19	3.743066933	3
monasca(compute)	1sec	0	205	133.8796004	148
	10sec	0	6	0.635334665	0
	60sec	0	6	0.375624376	0
	300sec	0	6	0.356593407	0

So what is learned?

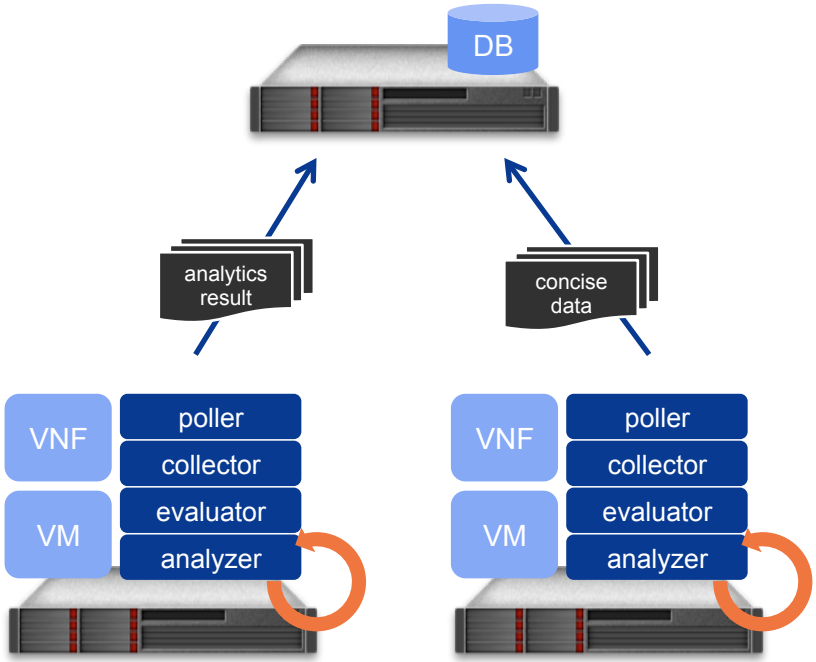
- Ceilometer (designed for telemetry)
  - Gnocchi is good performance
  - Gnocchi CLI shows the latest data after 1min
  - Shorter intervals make rabbitmq busy state
- Collectd
  - Good performance in short interval
  - Redis is feasible to our requirements in data storing point of view
    - No need to store in long time (last 1 day is enough)
    - Need quick access (in-memory DB)
- Monasca (designed for centralized monitoring)
  - Current monasca-agent designed for 15sec/30sec interval, usually
    - Less than 3sec interval, monasca-agent does not work (not supported?)
    - But it is reasonable interval for usual monitoring
  - Asynchronous processing (to put DB) makes 1-3 sec delay to visible
    - Monasca-api receives the data but it is not appeared immediately
    - Kafka seems bulk/batch data

5. Conclusion

KDDI Research Inc. Proprietary and Confidential

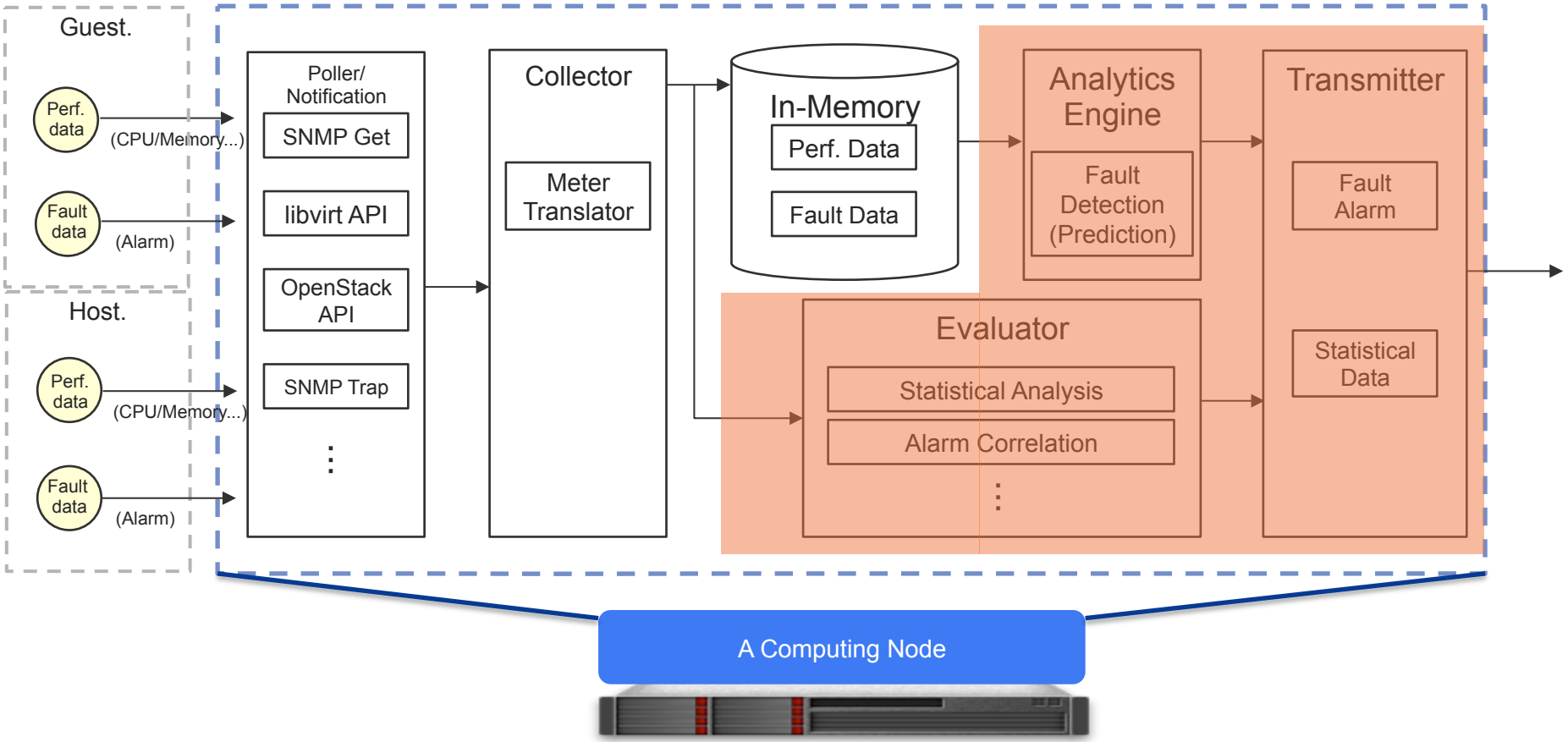
Conclusion

- High Quality Network Service
- Proactive Operation
- Monitoring
- DMA



KDDI Research Inc. Proprietary and Confidential

Future Work


Guest.
Perf. data
(CPU/Memory...)
Fault data
(Alarm)
Host.
Perf. data
(CPU/Memory...)
Fault data
(Alarm)
Poller/ Notification
SNMP Get
libvirt API
OpenStack API
SNMP Trap
Collector
Meter Translator
In-Memory
Perf. Data
Fault Data
Analytics Engine
Fault Detection (Prediction)
Evaluator
Statistical Analysis
Alarm Correlation
Transmitter
Fault Alarm
Statistical Data
A Computing Node

Join us

- Appendix: http://goo.gl/oAeGC5
- Etherpad: https://etherpad.openstack.org/p/dma-discussion
- Contact: dma-discuss@redhat.com