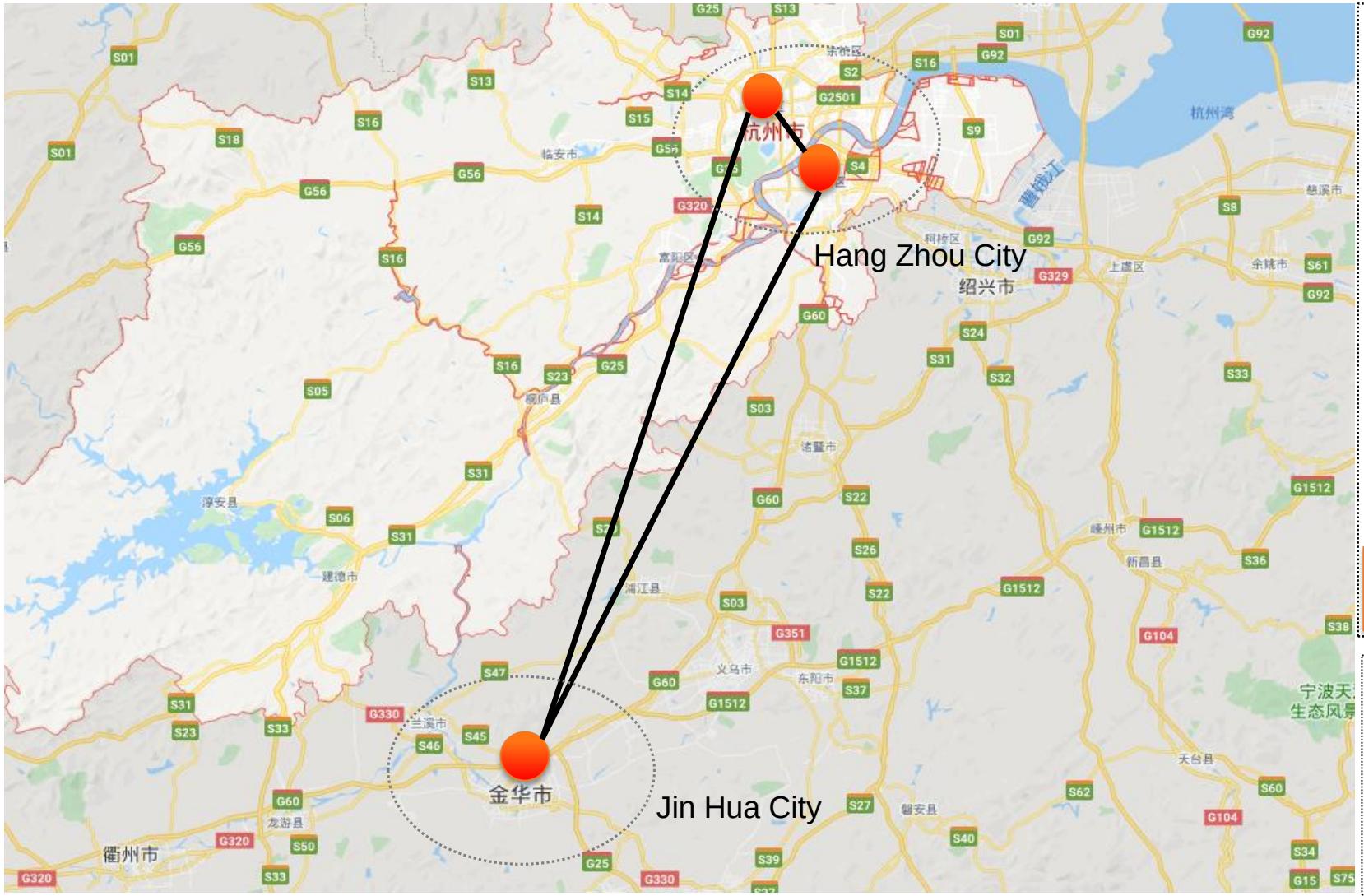
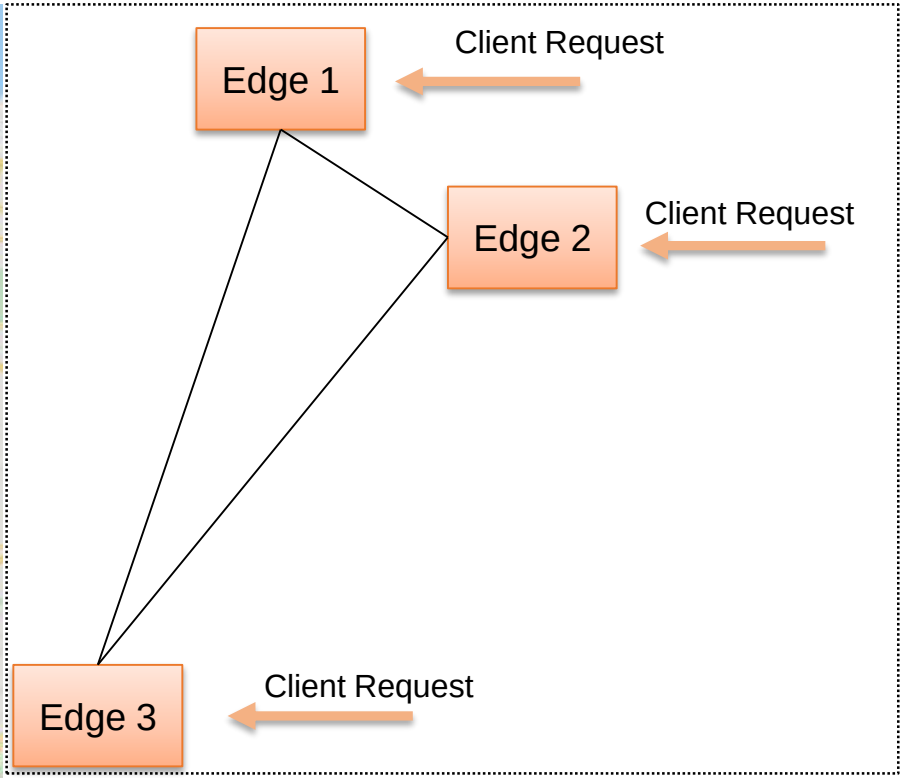
CFN (**C**ompute **F**irst **N**etwork) Field Trial POC highlights

Nov 5th 2019

Field Testbed





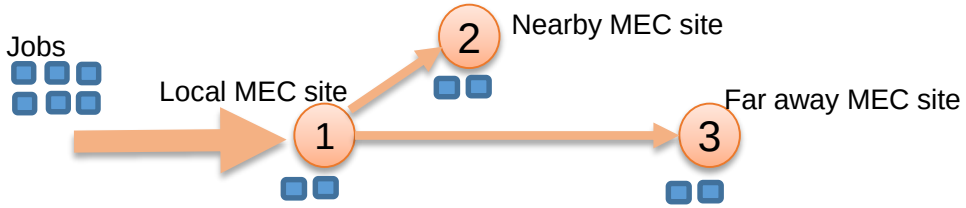
Setup
- Service randomly accessed from 3 Edges
- 10~30 request per client

Objective
- Use JCT (**J**ob **C**ompletion **T**ime) as key KPI for comparison

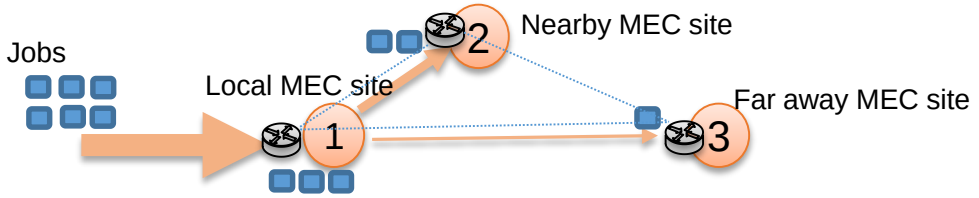
Case 1: Schedule equivalent services considering of **Network Metric**

Pre-assumption : Network metric (e.g. delay, throughput) is different and changing, this will impacts on **J**ob **C**ompletion **T**ime (JCT)

Expectation: Schedule traffic among equivalent services with considering of Network Metric will get gains



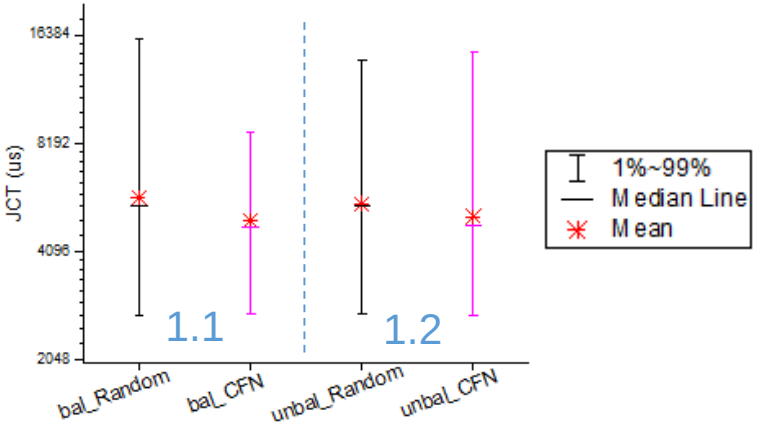
Load Balancer scheduling using static scheduling algorithm



CFN: scheduling with the considering of Network Metric

Gain from **Network metric awareness**

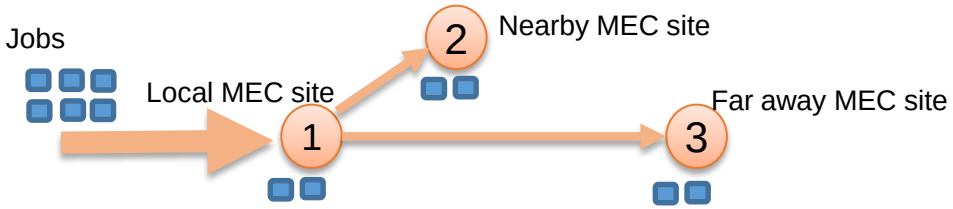
JCT (Job Completion Time)	Load Balancer	CFN	JCT Gain
CASE1.1 Balanced requests	5760	4971	15.8%
CASE1.2 Unbalanced requests	5601	5114	9.5%

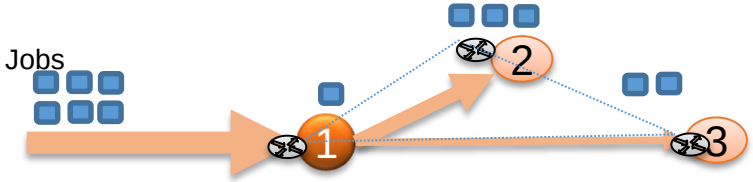


Case 2: Schedule equivalent services considering of **Computing Metric**

Pre-assumption : Computing metric (e.g. service load, CPU load, calculation time) is dynamically changing and have impacts on JCT

Expectation: Schedule traffic among equivalent services with considering of Computing Metric will get gains



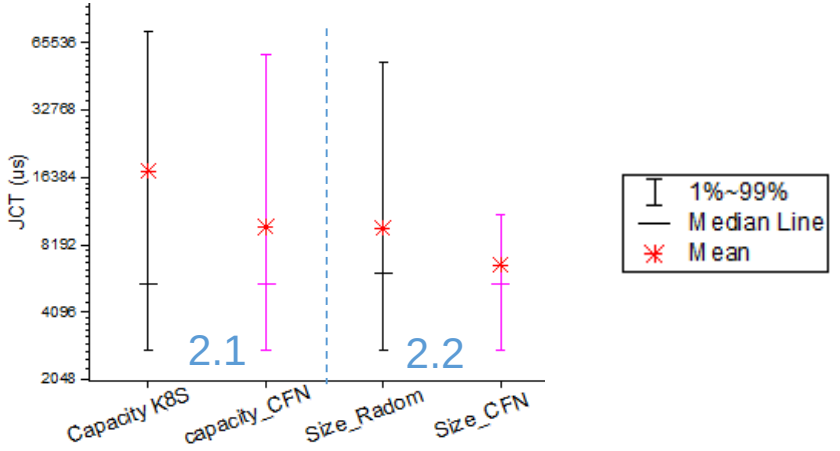


Via Load Balancer using static scheduling algorithm

JCT gains from Computing Metric awareness

JCT (Job Completion Time)	Load Balancer	CFN	JCT Gain
2.1 Impacted server capacity	17487	9805	78.3 %
2.2 Big Job impact on small job	9567	7226	32.4%

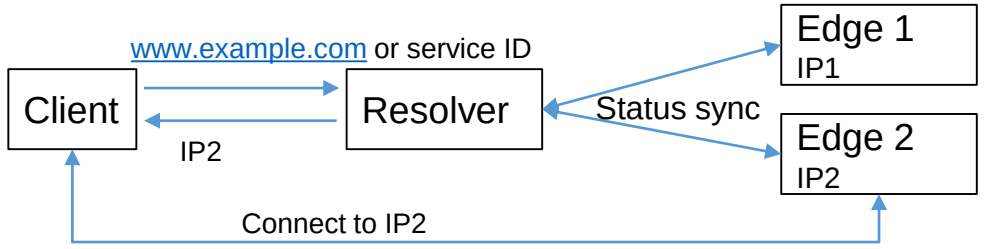
CFN: considering of Computing Metric



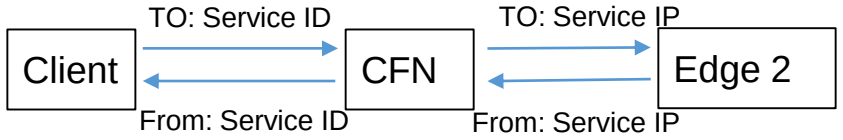
Case 3 – Centric VS Distributed CFN

Pre-assumption: Centralized solution to resolve a service IP before service request is widely used, and a dedicate ‘resolving’ procedure has negative impact on JCT

Expectation: In CFN, steering service in runtime will get JCT gains.



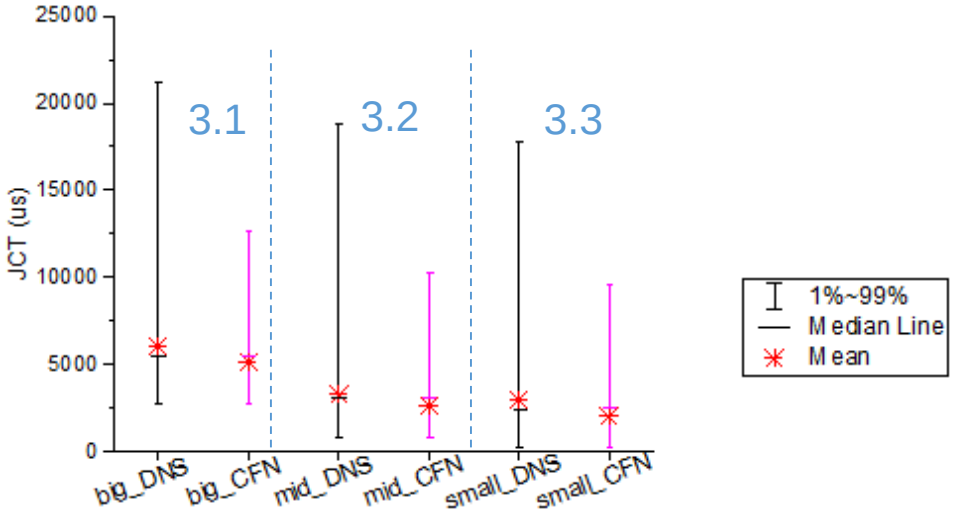
Connect to Edge 2 by additional ‘resolving’ procedure



CFN : Connect to Edge 2 in runtime

Smaller job higher overhead reduced, higher JCT gain

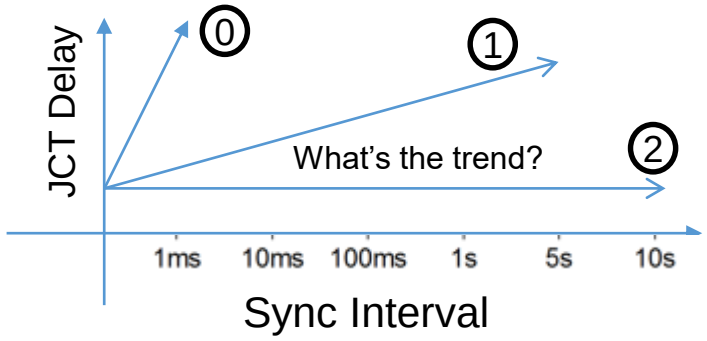
JCT (Job Completion Time)	Load Balancer	CFN	JCT Gain
3.1 Big Job	6008	5145	16.8%
3.2 Medium Job	3276	2635	23.9%
3.3 Small Job	2971	2047	45.4%



Case 4 – Sync interval impact

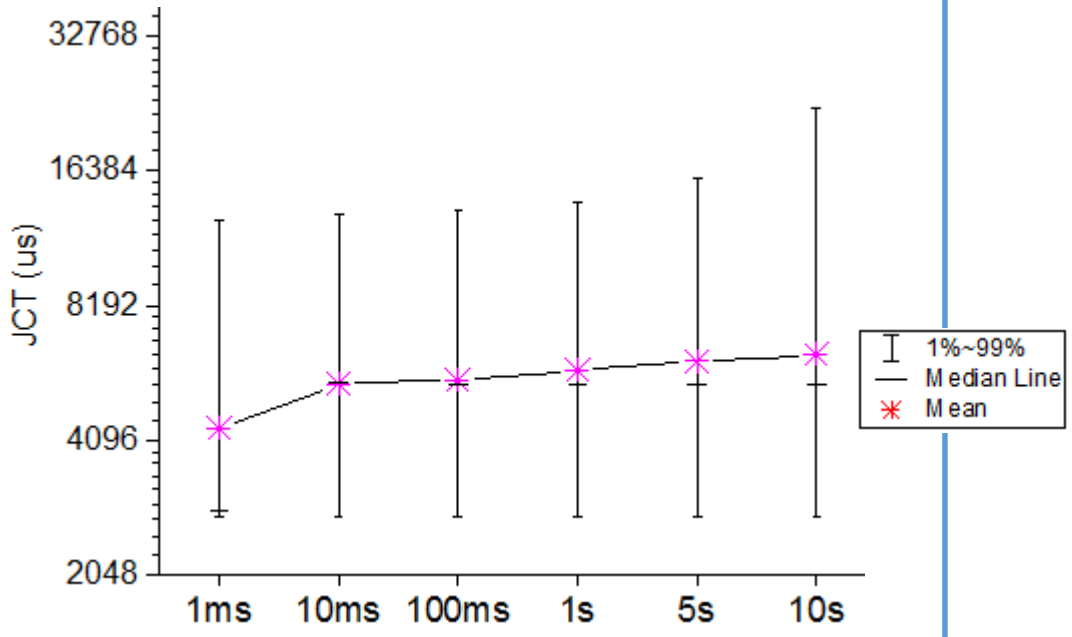
Pre-assumption: Higher sync interval will cause the synchronized service information out of date, thus service scheduling will be impacted.

Expectation: In CFN, higher sync interval has negative impact on JCT



Higher sync interval, worse JCT

JCT (Job Completion Time)	1ms	10ms	100ms	1s	5s	10s
30% Load	4058	4883	5312	5741	6255	6445
JCT Loss	**0**	**20.3%**	**30.9%**	**41.5%**	**54.1%**	**58.8%**



Conclusion

- Results as expected, CFN got JCT gain in all tested cases.
- Large gain during dynamic changing status (EG: changing server capacity, dynamic service traffic etc.)
- More benefit from distributed CFN Architecture
- To achieve good performance, the key is : fast sync of status change

Questions

Metric Calculation

```
SIDWeight = (
{
        sid = "200.200.200.201";        #Service ID
        scheduleMethod = 2 ;            # 0: best metric, 1: for Random Pick (same as K8S), 2: margin for random selection
        delayWeight = 0.3 ;             # network delay
        delaySLA = 10.0 ;               # ms
        calcSLA = 10.0;                 # ms
        calcWeight = 0.1 ;              # Computing delay on server side
        loadWeight = 0.6 ;              # load = connections / capacity
        loadSLA = 0.9  ;                #
        qpsWeight = 0.0;                #for QPS
        qpsSLA = 150.0;                 #for QPS
        punishFactorDelay = 1.0 ;       #when SLA not meet for delay (exceed)
        punishFactorLoad = 1.5 ;        #when SLA not meet for load (exceed)
        punishFactorQPS = 1.5 ;         #when SLA not meet for QPS  (exceed)
        margin = 0.01;                  #method 1:local prefer margin.  method 2: margin for random selection.
        redir = false;                  #whether redirection from network is allowed (when service unavailable for worse enough).
},
```

Network Metric

Computing Metric

Formula: (if metric not exceed SLA, otherwise a punish factor will be multiplied)

> MetricTotal = $\sum_{k=0}^{n} weight_k * \frac{Metric_k}{SLA_k}$