DNSの可視化検討

2022年1月27日

NTTコミュニケーションズ株式会社

小坂　良太

本日の流れ

・はじめに
  - 可視化の重要性
  - DNSを可視化して分かったこと

・DNSの構成について
  - OCNのフルサービスリゾルバ(キャッシュDNS)について
  - KDDI株式会社 松本様より「LBレス事例の共有」

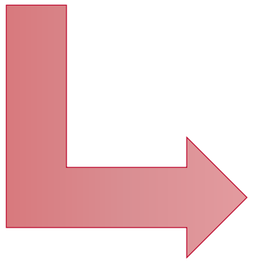
・可視化システムの検討
  - DNSトラヒックの「何を」「どうやって」可視化するか
  - 可視化システム(PoC)のご紹介

・【参考】その他の可視化方法について
  - 他ISPのヒアリング内容の共有

はじめに

可視化の重要性

何のために可視化するのか？



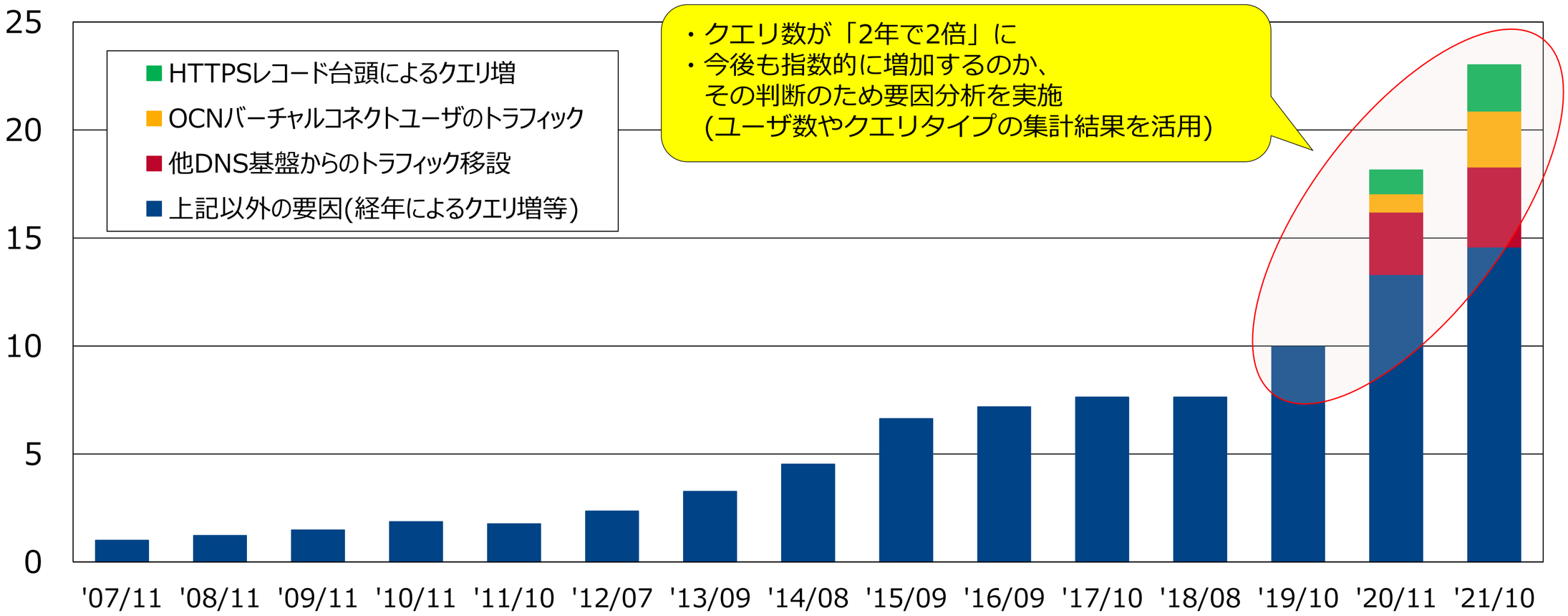
・需要予測を行い適切に増設対応を行うため

・障害/攻撃発生時の一次切り分けのため

・トレンドの把握のため

DNSを可視化したことで分かったこと：需要予測について

■お客様からのDNSクエリ数の推移と直近2年の「クエリ数増の要因」

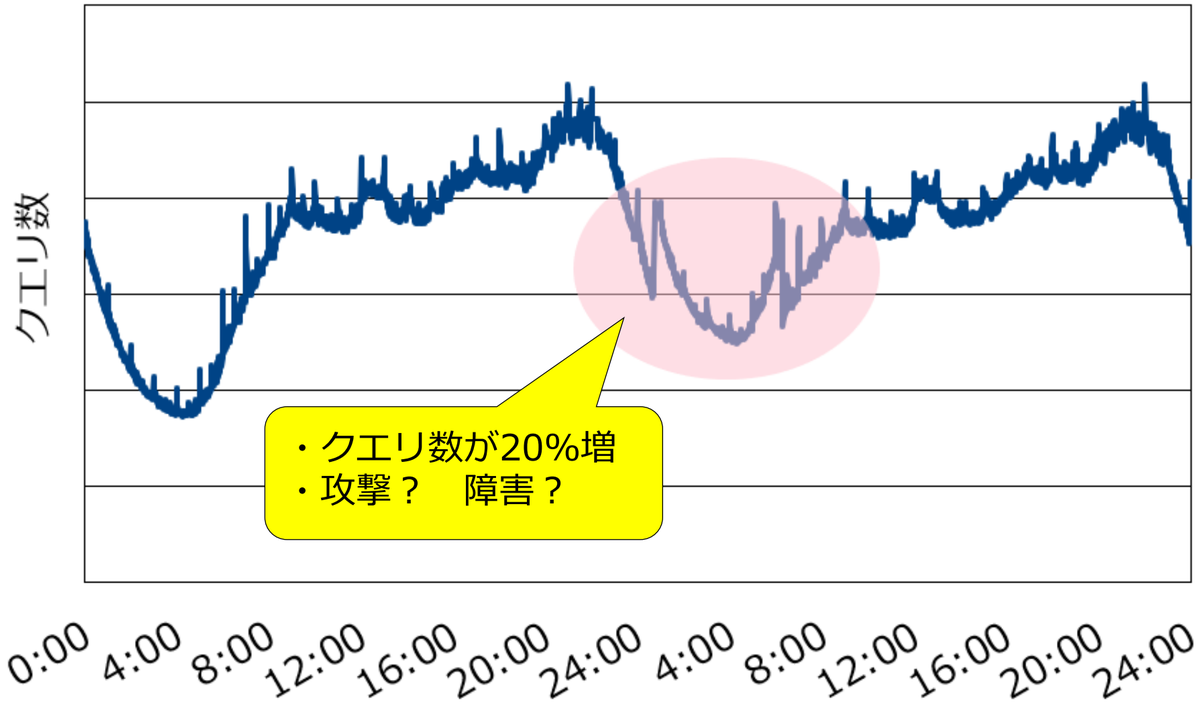


※縦軸は'07/11の値を１とする

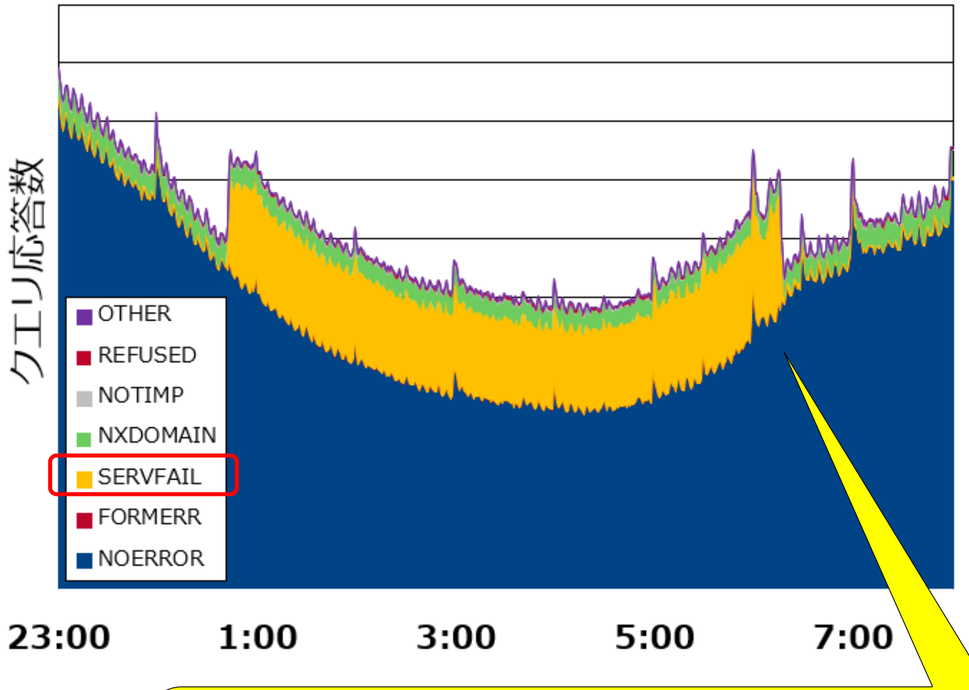
DNSを可視化したことで分かったこと：障害の一次切り分け

■とある障害時のDNSクエリ数の変化

■ユーザクエリ(48時間)



■応答クエリ(クエリ増の時間帯の抜粋)

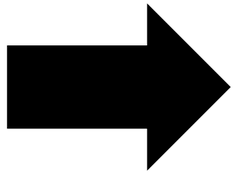


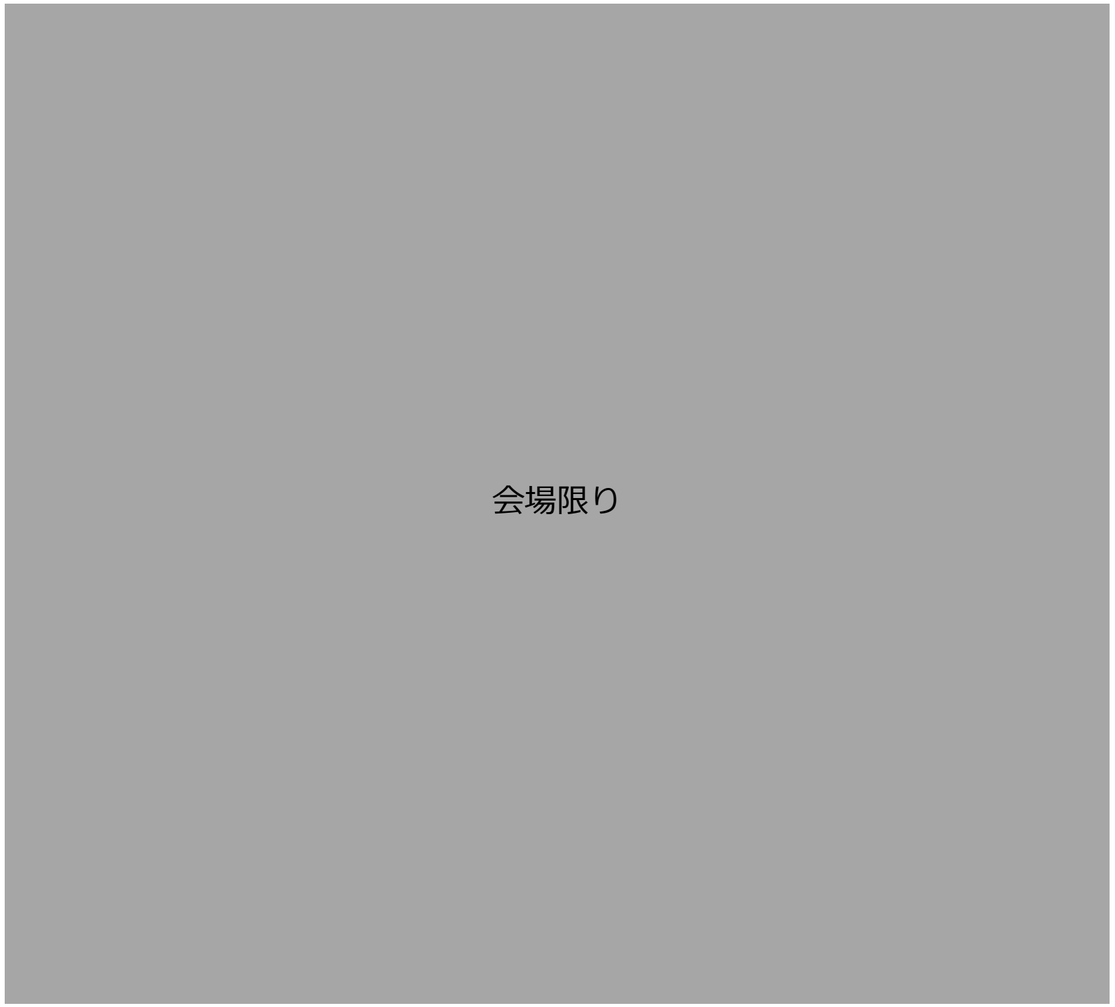
・NOERRORの数は問題なし＝全断ではない
・その上でSERVFAILが増加
　＝どこかの権威DNS(ドメイン)に障害あり？
　⇒SERVFAILのドメイン調査が最優先

DNSを可視化したことで分かったこと：トレンドの把握①

■DNSレコードタイプ毎のクエリ集計

■OCNの全DNSトラヒック





・固定回線・モバイル、IPv4・IPv6等を分けずに集計
・Aレコードが56%、AAAAレコードが30%、HTTPSレコードが11%

DNSを可視化したことで分かったこと：トレンドの把握②

■DNSレコードタイプ毎のクエリ集計

■OCNの全DNSトラヒック

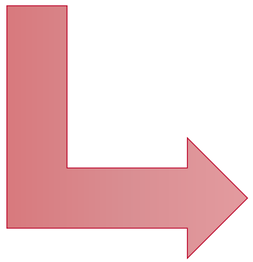
■全DNSトラヒックからSRVレコードのみ抽出

・SRVレコードはOffice365などやVoIPで必要となる
・法人のお客様がメインとなるので9:00にピークがある
　(grafanaを使ったことで、初めて気づいた)

・「DNSレコードタイプ毎のクエリ数の可視化」といっても見せ方は様々
・可視化ができていても見せ方(やUI)を考えていく必要あり

可視化を行うにあたり決めるべきこと

- 何を可視化するのか(データとして何を収集するのか)
- どう可視化するのか(どういったツールを使うのか)



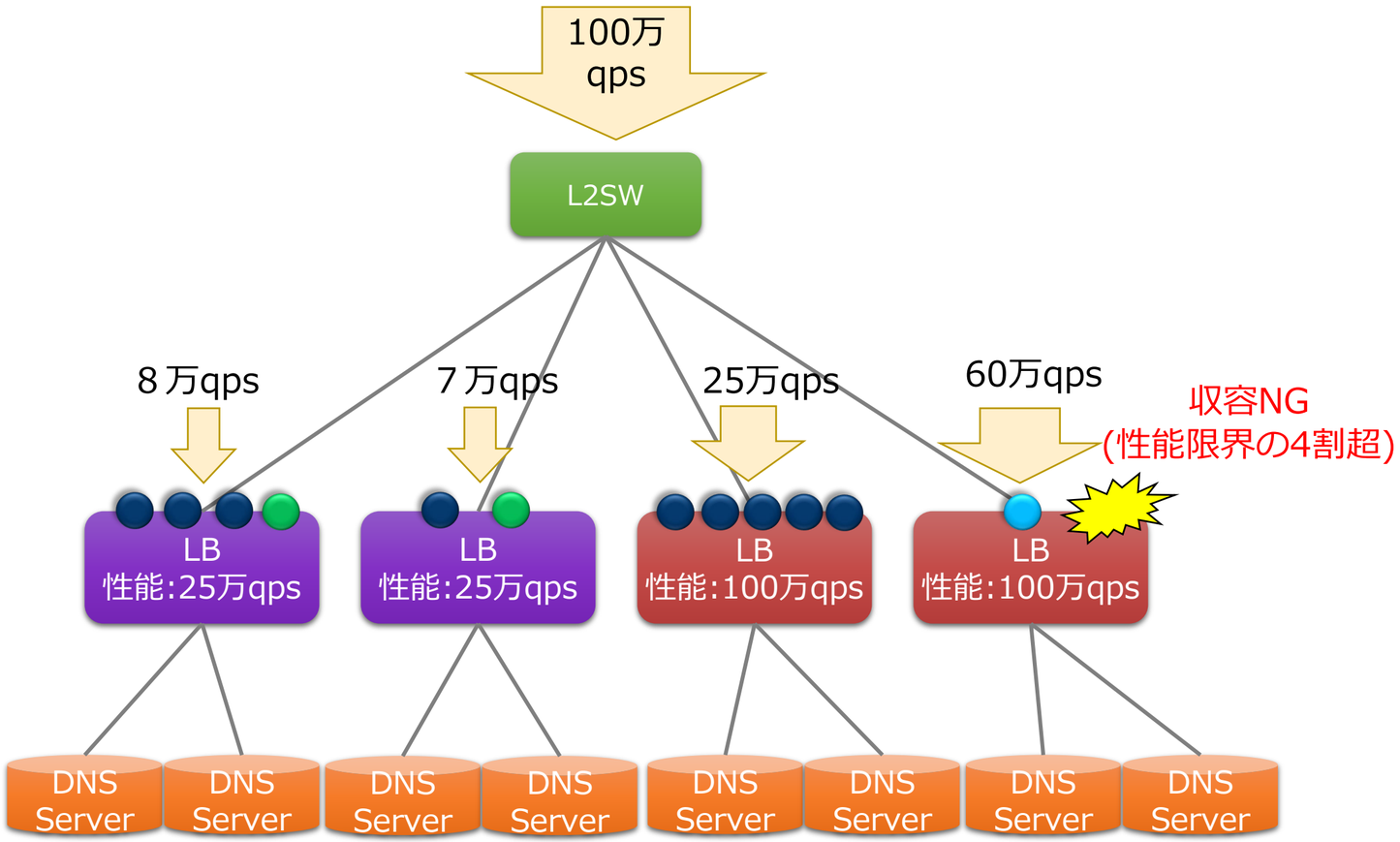
前提条件の整理のため
キャッシュDNSの構成をご紹介致します！

OCNのフルサービスリゾルバ(キャッシュDNS)について

OCN DNSの構成遷移

(※)DSR=Direct Server Return

ロードバランサー NAT構成を採用していた時の話

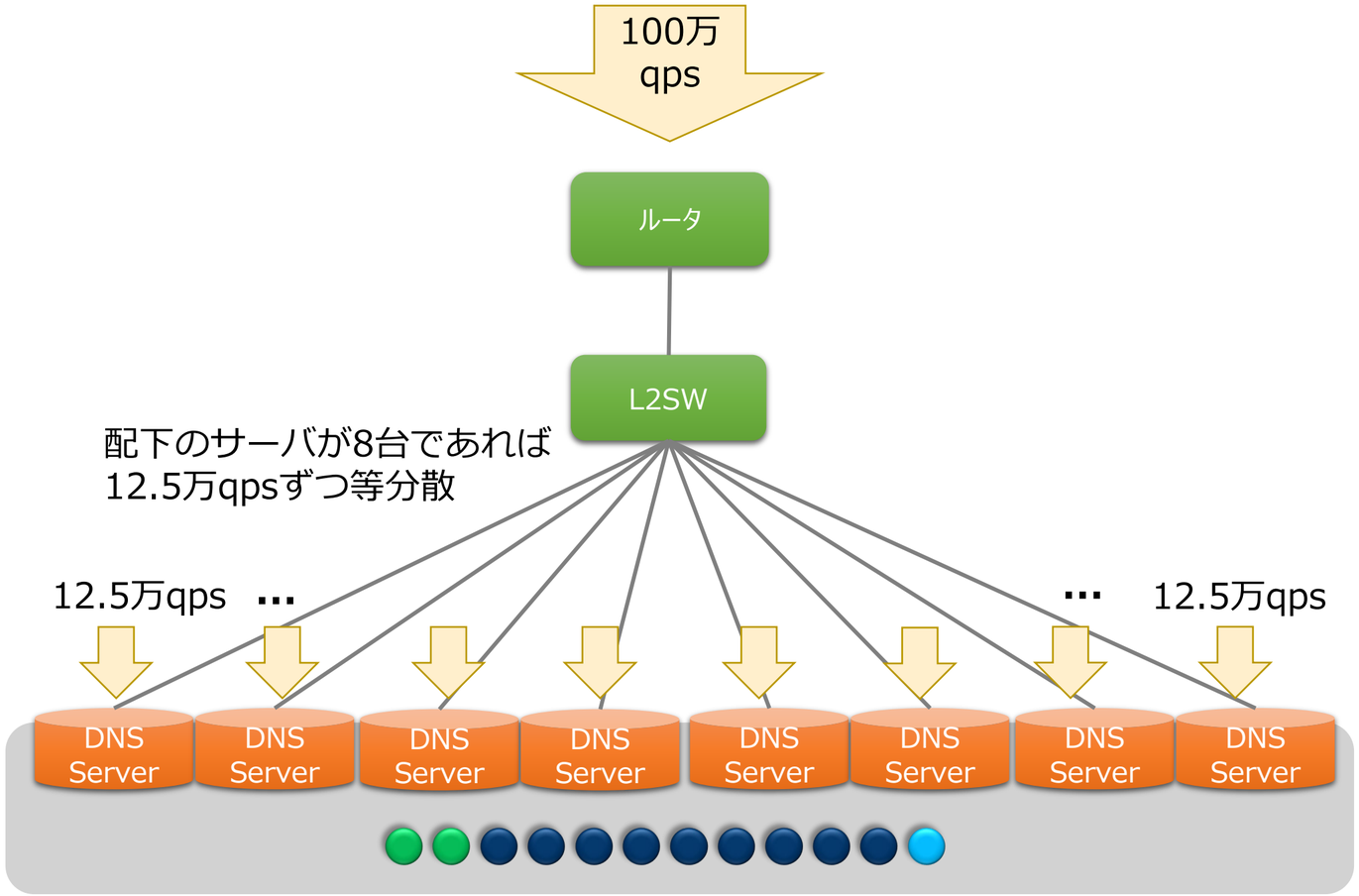


- サービス、地域毎に宛先IP(VIP)を分離
- 複数のロードバランサー(LB)を運用
- VIP毎にクエリ数をモニタリングし
  必要に応じて収容変更
- LBを増設する際にもVIPの収容替えを実施
- IPoE向けVIPのクエリ増に伴い
  アーキテクチャの見直しを実施

🟢：OCN ADSLのお客様用DNS
🔵：OCN光 (PPPoE)のお客様用DNS
🔵：OCN光 (IPoE)のお客様用DNS

(補足)本資料の構成および性能値は実際のものではありません(シンプル化しています)

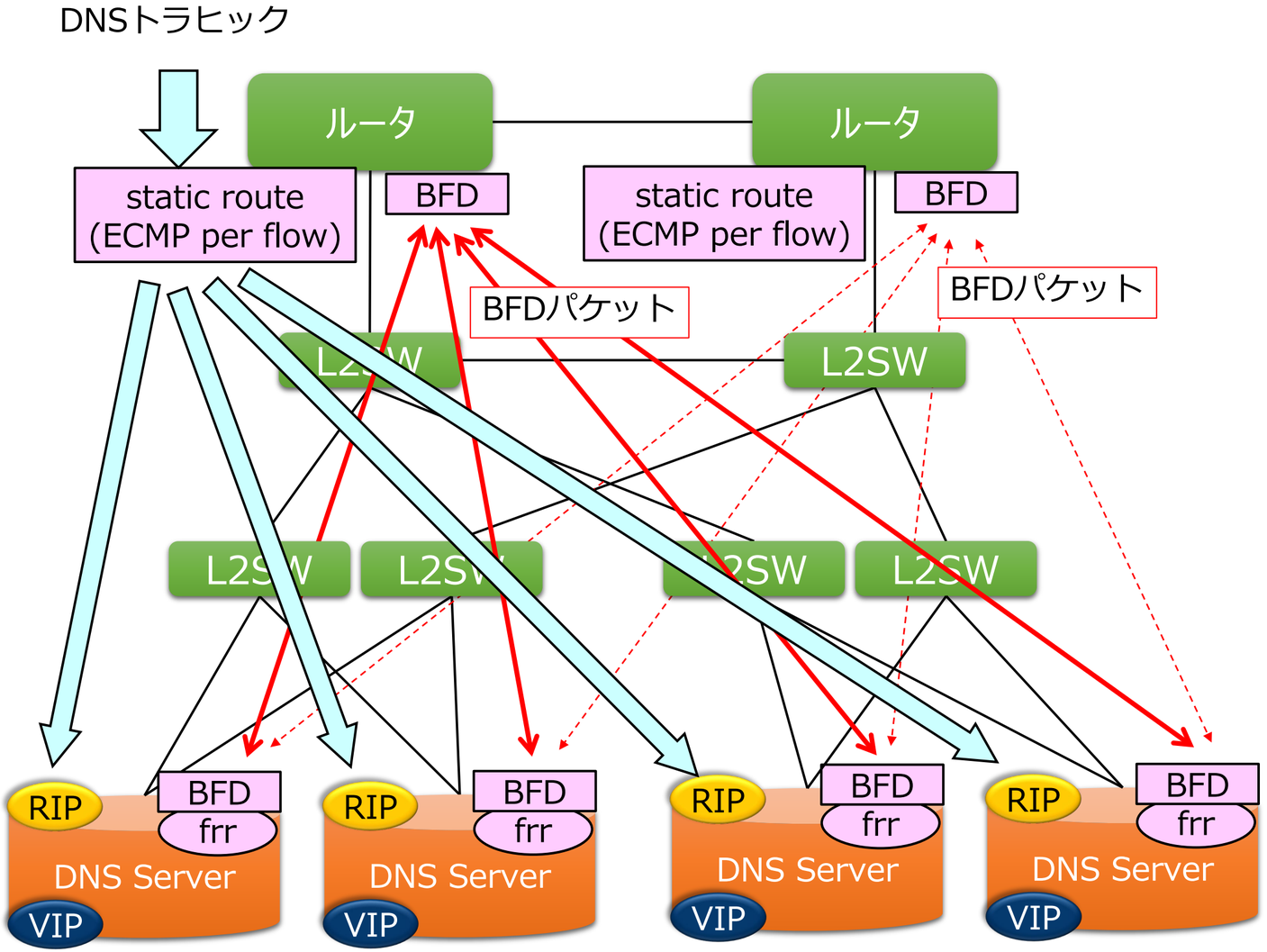
ロードバランサーなし構成の概要



- ボトルネックがLBだったため
  LBなし構成を検討＆リリース
- サーバを増やすだけでスケールアウトが可能
- VIPのクエリ数に偏りがあっても問題なし
  - but クエリ数は引き続きモニタリングが必要

: OCN ADSLのお客様用DNS
: OCN光 (PPPoE)のお客様用DNS
: OCN光 (IPoE)のお客様用DNS

ロードバランサーなし構成のアーキテクチャ



・トラヒックの分散のためECMPを有効化

・負荷分散対象の決定にBFDを利用
- 組み込み/組み外しを1秒未満にて実施可能

・DNSサーバでBFDを動かすためFRRを導入

■用語

ECMP:	Equal Cost Multi Path
BFD:	Bidirectional Forwarding Detection
FRR:	Free Range Routing

・・・

詳細はDNS Summer Day 2021の資料参照
https://dnsops.jp/event/20210625/13-kosaka.pdf

可視化システムの検討

何を可視化するか

まずは基本的なデータを可視化することを目標とする

- 宛先IP(VIP)毎のクエリ数、QTYPE、RCODE
  - QTYPE=A/AAAA/NS/SOAなど
  - RCODE=NOERROR/NXDOMAIN/SERVFAILなど
- DNS基盤全体やリージョン毎のクエリ数、QTYPE、RCODE(合計値)
  - 総クエリの伸びが一目で分かるように
- UDPとTCPのそれぞれのクエリ数
- 送信元IP数

可視化システムを作る上で前提とした条件について

OCN DNSシステムにおける前提条件	前提条件による影響や制限、備考など
サーバ台数が多い	・サーバ側でデータを取得する場合は、適切に合算処理が必要 ・サーバ側に何か仕込む場合は全台工事が必要
クエリ数が多い	・クエリログを全てDBに保持し続けることは困難 (1日10TB以上、10年で50PB以上⇒専用の分析基盤が必要なレベル)
宛先IPが多い	・DNSアプリが出力する統計ファイルでは不十分 - サーバ毎の統計であれば取得可能
宛先IP毎に統計を取りたい	
1つのDNSサーバに複数の宛先IPが相乗りしている	
商用のDNSサーバに負荷をかけたくない	・サーバ側に何かを仕込む場合は軽量である必要あり
リアルタイム性の優先度は低	・数百万qpsを処理する場合に10分程度の処理時間を許容 - 高速化・並列化が可能なら数分程度に改善も可能と想定
可視化システムには移行性をもたせたい	・例えばCactiの場合は移行にハードルあり - LB①からLB②に切替をした場合 - サーバを増設/除却した場合 にデータの引継ぎが困難

可視化の方式検討

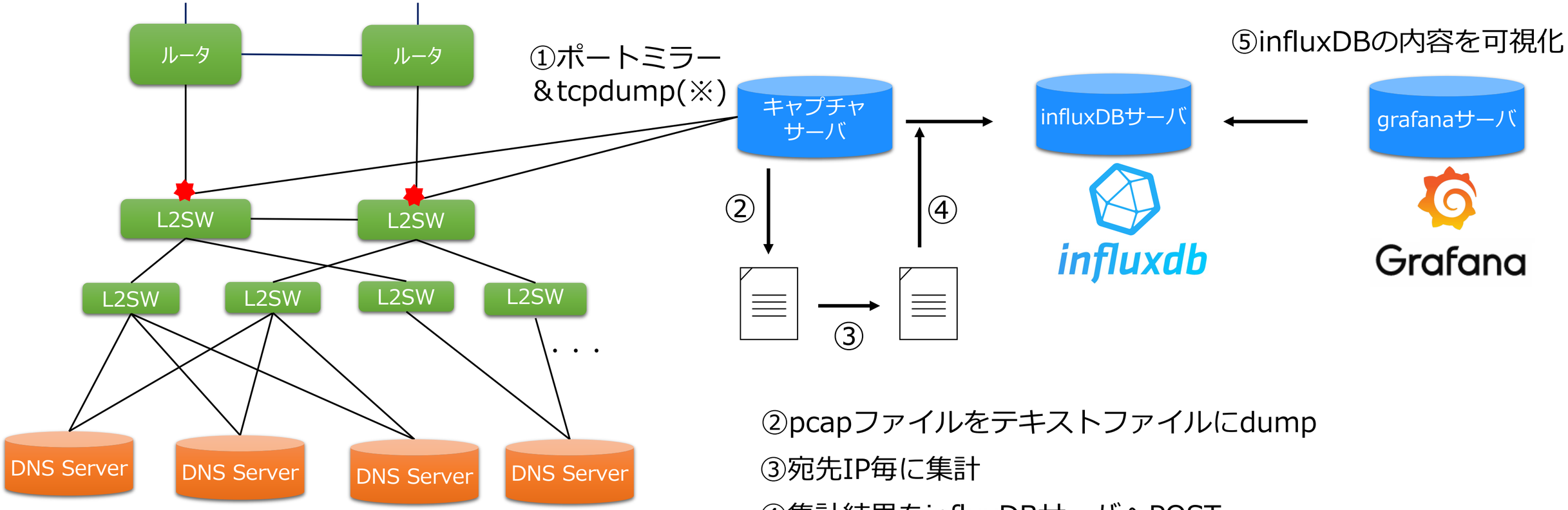
赤字：評価に影響した項目

	Netflow	ポートミラー (NW上)	パケットキャプチャ (サーバ上)	クエリログ (dnstap)	DNSソフトウェアの統計	SNMP (LB上)
可視化ソフトウェアの例	-	DSC(DSP)	DSC(DSP)	-	Prometheus	Cacti
宛先IP毎に統計が取れるか	○	○	○	○	×	○
DNSサーバ上で作業が必要か	なし	なし	あり	あり	あり	なし
L7(DNS)情報が取れるか	×	○	○	○	○	×
TCPクエリ数が取れるか	○	○	○	○	○	△
送信元IP数が取れるか	○	○	○	○	×	×
DNS over HTTPS対応(※)	×	×	×	○	○	×
キャッシュヒット率(※)	×	×	×	×	○	×
評価	×	○	△	△	×	-

(※)要件に入れていないため今回は評価項目から除外

- 今回の可視化はFirst Stepであり今後も再検討＆要修正の予定
- そのためDNSサーバ上での作業が不要なポートミラー(NW上)方式を採用
- PoCとして構築したシステムは前ページの他の前提条件もクリア

可視化システム(PoC)の全体像



②pcapファイルをテキストファイルにdump

③宛先IP毎に集計

④集計結果をinfluxDBサーバへPOST

(※)サーバ側にはキャプチャボードとしてNT100A01を採用

可視化システム(PoC)の説明①

①ポートミラー＆tcpdump

```
# /opt/napatech3/sbin/tcpdump -i napa0 -G 60 -w /ram/dns-%Y-%m-%d_%H.%M.%S.pcap -s 8192 -z /root/launch_dns-decoder.sh port 53
```

コマンド、オプション	説明
/opt/napatech3/sbin/tcpdump -i napa0	• キャプチャボード用にコンパイルしたtcpdumpを利用 • キャプチャボードのIF名であるnapa0を指定
-G 60	• pcapファイルをローテートする秒数 • RAMディスクが溢れないよう60秒毎にローテート
-w ファイル名	• pcapファイル出力先(RAMディスクを指定) • SSDは書き込み上限(TBW)があるため利用しないこと
-s 8192	• 取得する最大パケットサイズ • 高速化のため上限を設定
-z 実行ファイル名	• ローテート後、pcapファイルに対して実行するスクリプトを指定 • 詳細は次ページ
port 53	• DNSクエリのみを集計の対象とする(UDPとTCPの53番ポート)

可視化システム(PoC)の説明②

②pcapファイルをテキストファイルにdump

```
# cat /root/launch_dns-decoder.sh
#!/bin/sh

/root/dns-decoder $1
rm -rf $1

exit 0
```

・pcapのファイル名は引数で渡される
・ファイル名をテキストdump用プログラムに渡す
・pcapファイルはファイルサイズが大きいため処理後は消去

■dns-decoderについて

・libpcapのサンプルプログラムを参考にC言語で自作(ソフトウェアは未公開)
・IPv4とIPv6のパケットをそれぞれ次ページのフォーマットに従ってテキストファイルに出力
・出力後、集計プログラム(後述)を起動して終了
・tcpdump –rと比べ30倍で処理可能(100万qps、60秒のpcapファイルで5分程度)

可視化システム(PoC)の説明②のつづき

```
# 1パケット1行で出力。以下はとある1パケットのテキスト化。
1640155460,79,IPv4,192.0.2.1,198.51.100.1,UDP,1040,53,0,0100,28,0
```

L7の情報(DNS)

例	説明
1640155460	• UNIX時間(pcapパケットヘッダから取得)
79	• パケット長(pcapパケットヘッダから取得)
IPv4	• プロトコルバージョン(IPヘッダから取得)。IPv4とIPv6のみ出力
192.168.2.1	• 送信元IP
198.51.100.1	• 宛先IP
UDP	• L4プロトコル。UDPとTCPのみ出力
1040	• 送信元ポート番号
53	• 宛先ポート番号
0	• DNSのQRbit(問合せの時は0、応答の時は1。DNSヘッダから取得)
0100	• DNSヘッダの16bit目～32bit目を16進数で表示
28	• QTYPE (AやAAAAなどの10進数表示) • **固定位置ではないためDNSデータ部分をデコードし取得**
0	• RCODE(NOERRORやNXDOMAINなどの応答コードを10進数表示)

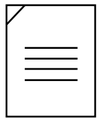
L7の情報(DNS)

・tcpdump –r で表示される情報と大差なし
・視認性向上とデータの処理の都合上、プロトコル部分やIP部分は数値・バイナリではなく文字列で出力

可視化システム(PoC)の説明③

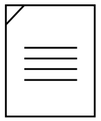
③宛先IP毎に集計

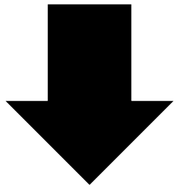
pcapを
テキスト化したもの



＋

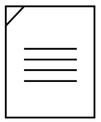
宛先IP一覧





pythonスクリプトで処理

集計結果



```
# cat (宛先IP一覧)

VIP    PPPoE  v4     East     198.51.100.1
VIP    PPPoE  v4     East     198.51.100.2
VIP    PPPoE  v4     East     198.51.100.3
VIP    Mobile  v4     East     198.51.100.4
VIP    PPPoE   v6     East     2001:db8::1
VIP    IPoE    v6     East     2001:db8::2

VIP    PPPoE  v4     West     192.51.100.5
VIP    PPPoE  v4     West     192.51.100.6
VIP    PPPoE  v4     West     192.51.100.7
VIP    Mobile  v4     West     192.51.100.8
VIP    PPPoE  v6     West     2001:db8::3
VIP    IPoE    v6     West     2001:db8::4
```

・集計を行いたいIPを記載

・合算処理をしやすくするためいくつかtagを併記
(何用のDNSか、DNSサーバの収容ロケーションなど)

可視化システム(PoC)の説明③のつづき

```
# cat (集計結果ファイル)
1640155460
tag,tag,tag,tag,tag,value,value,value,value,value,value,value,value,value,value,value,value,value,value,(略)
IPType, Usage, IPv, Location, IP, in qps, out qps, in bytes, out bytes, A,NS,CNAME,(略), NOERROR,FORMERR,(略),UDP,TCP,srcip128,srcip56
VIP,Mobile,v4,East,198.51.100.4,26,26,2256,4541,20,0,0,0,0,0,0,0,0,0,0,6,0,0,26,0,0,0,0,0,0,26,0,5,5
VIP,PPPoE,v6,East, 2001:db8::1,1722,1701,180090,290841,882,0,0,0,34,0,9,684,0,0,0,0,113,0,0,1188,1,470,40,0,0,2,1718,4,318,312
```

- influxDBへPOSTしやすい形、grafanaで可視化しやすい形に集計
- 1行目はUNIX時間(1分のpcapファイルの集計結果だが、ここでは一番最初のパケットの時間を記載)
- 2-3行は固定値(tag/valueはinfluxDB用語)
- 4行目以降に各VIPの統計サマリを記載
  - QTYPEは主要な14タイプをカウント(それ以外はその他としてカウント)
  - RCODEは主要な6種をカウント(それ以外はその他としてカウント)
  - 送信元IP数はユニークIP数をカウント(IPv6は/128と/56の集計をそれぞれ実施)
- qps(query per sec)として出力するためソースIP数以外はカウント後に1/60倍を実施(1分間のpcapの場合)

可視化システム(PoC)の説明④

④集計結果をinfluxDBサーバへPOST

```
$ curl -s -XPOST “http://(indluxDBのIP):8086/write?db=(DB名)&precision=s” --data-binary “(measurement名),¥
IPType=VIP,Usage=Mobile,IPv=v4,Location=East,IP=198.51.100.4 inqps=26,outqps=26,inbytes=2256,outbytes=4541,¥
A=20,NS=0,CNAME=0,SOA=0,PTR=0,MX=0,TXT=0,AAAA=6,SRV=0,DS=0,RRSIG=0,DNSKEY=0,HTTPS=0,ANY=0,OTHER=0,¥
NOERROR=26,FORMERR=0,SERVFAIL=0,NXDOMAIN=0,NOTIMP=0,REFUSED=0,OTHER2=0,¥
UDP=26,TCP=0,srcip128=5,srcip56=5 1640155460"
```

・集計結果をcurlでinfluxDBへPOST(1VIPに対して1POST)

・--data-binaryで送るデータは
measurement名,tagA=hoge,tagB=fuga valueA=1,valueB=2 UNIX時間　という形

可視化システム(PoC)の説明④のつづき

```
$ influx -precision rfc3339
Connected to http://localhost:8086 version 1.8.9
InfluxDB shell version: 1.8.9
> use DB名
> select time,Usage,IP,inbytes,outbytes,A from measurement名 where   IP='198.51.100.4’ limit 5
name: measurement名
time                    Usage    IP            inbytes outbytes A
----                    -----    --            ------- -------- -
2021-11-26T02:43:30Z Mobile 198.51.100.4 56      106      0
2021-11-26T02:44:30Z Mobile 198.51.100.4 93      209      0
2021-11-26T02:45:30Z Mobile 198.51.100.4 230     462      1
2021-11-26T02:46:30Z Mobile 198.51.100.4 114     209      0
2021-11-26T02:47:30Z Mobile 198.51.100.4 219     422      2
>quit
$
```

・influxDBにPOSTした時刻ではなくパケットキャプチャした時刻
・キッチリ1分間隔となる(1分毎にpcapを取得しているため)

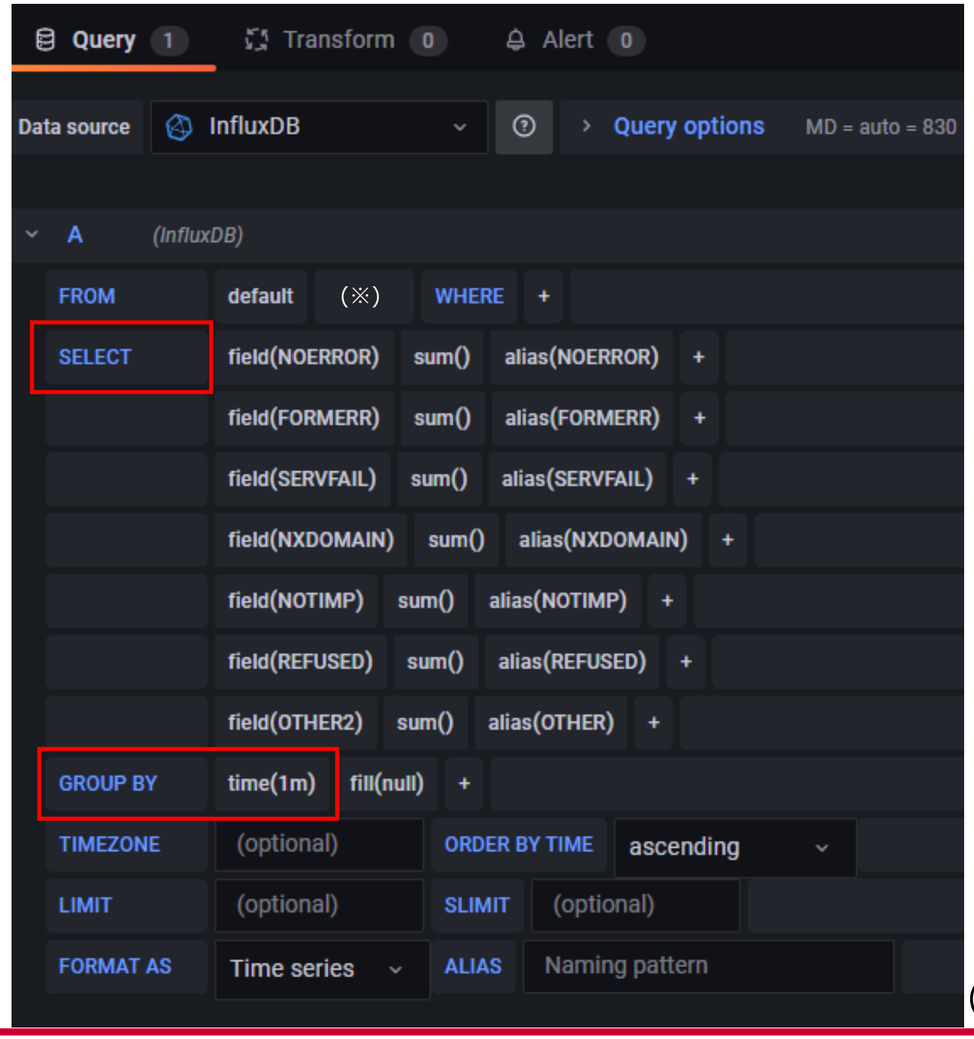
・influxDBでは一般的なDBと同じようなSQL文が利用可能

・必要なディスク容量は1VIP、1ヵ月分のデータで1MBほど
  - 各VIPに対して1分毎にデータを格納

可視化システム(PoC)の説明⑤

⑤influxDBの内容を可視化

■grafanaのパネル設定画面



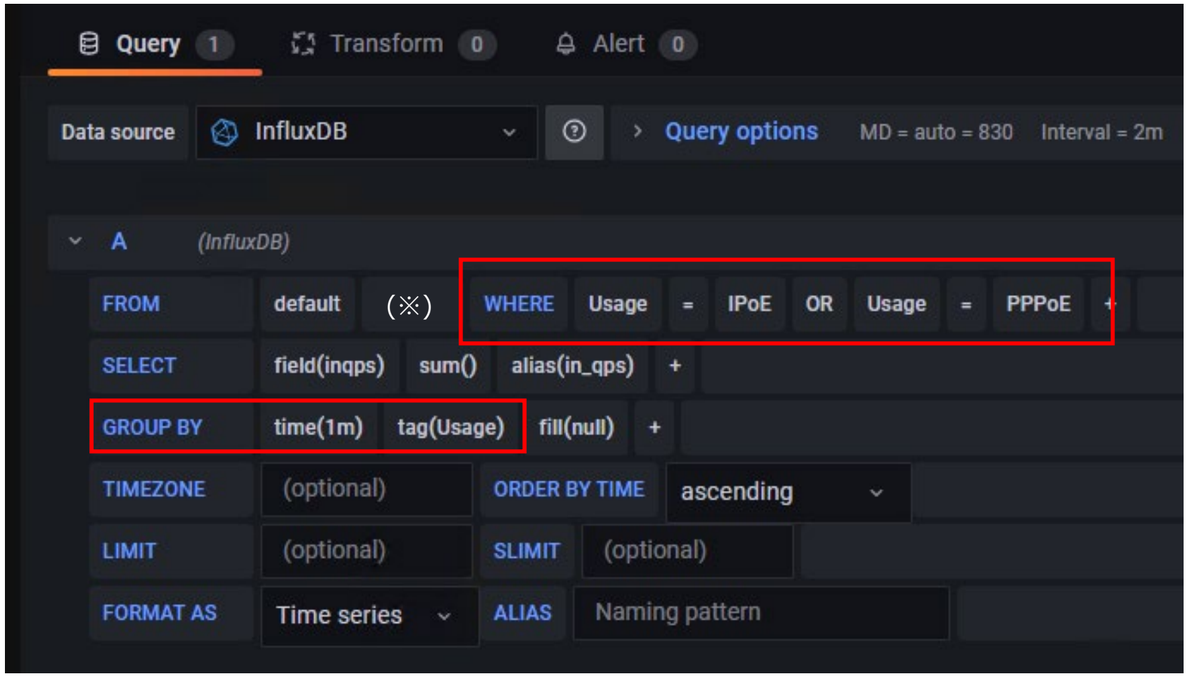
会場限り

・表示したい数値をSELECT文で指定
  - sum()を使うことで合算値を可視化

・折れ線グラフとなるようにGROUP BYを1mに変更

(※)measurement名を指定

可視化システム(PoC)の説明⑤のつづき

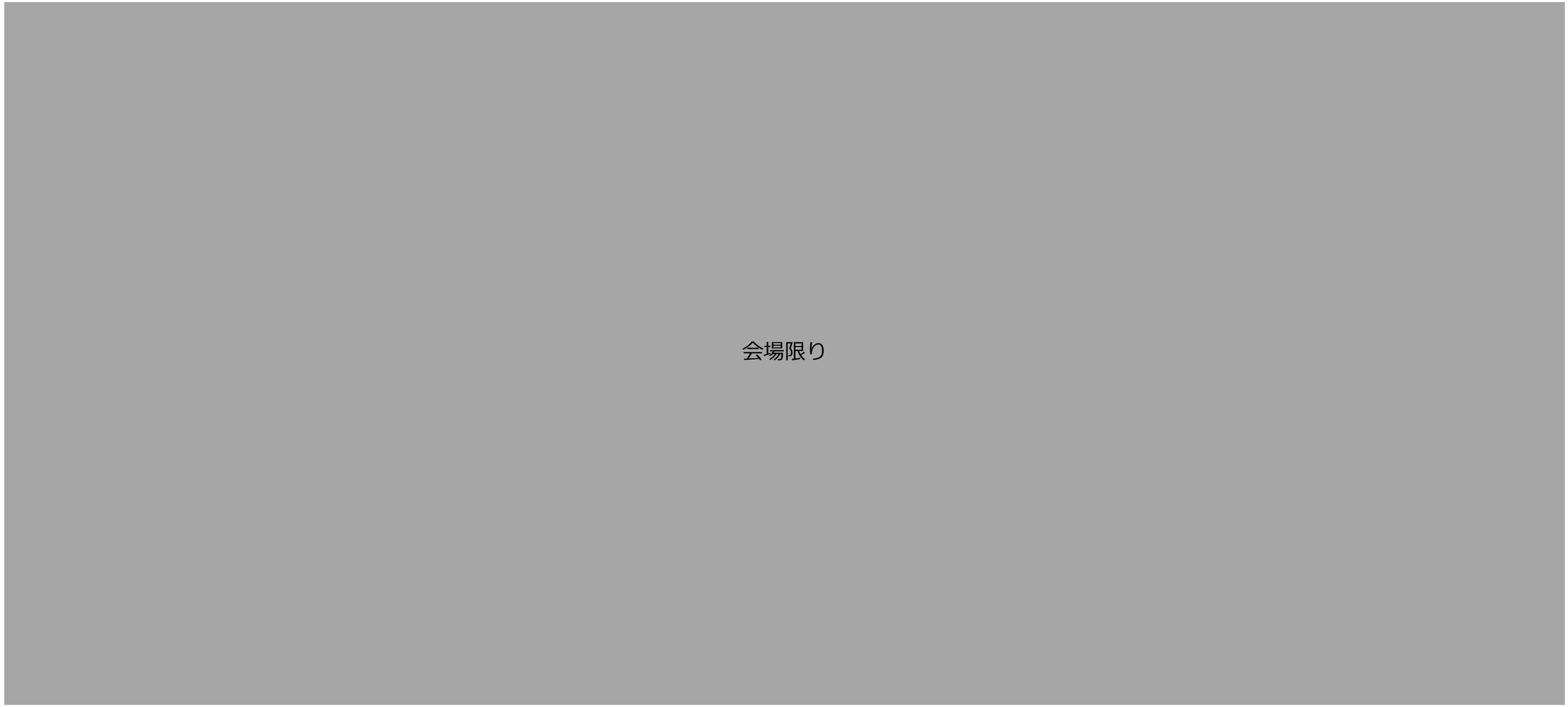


(※)measurement名を指定

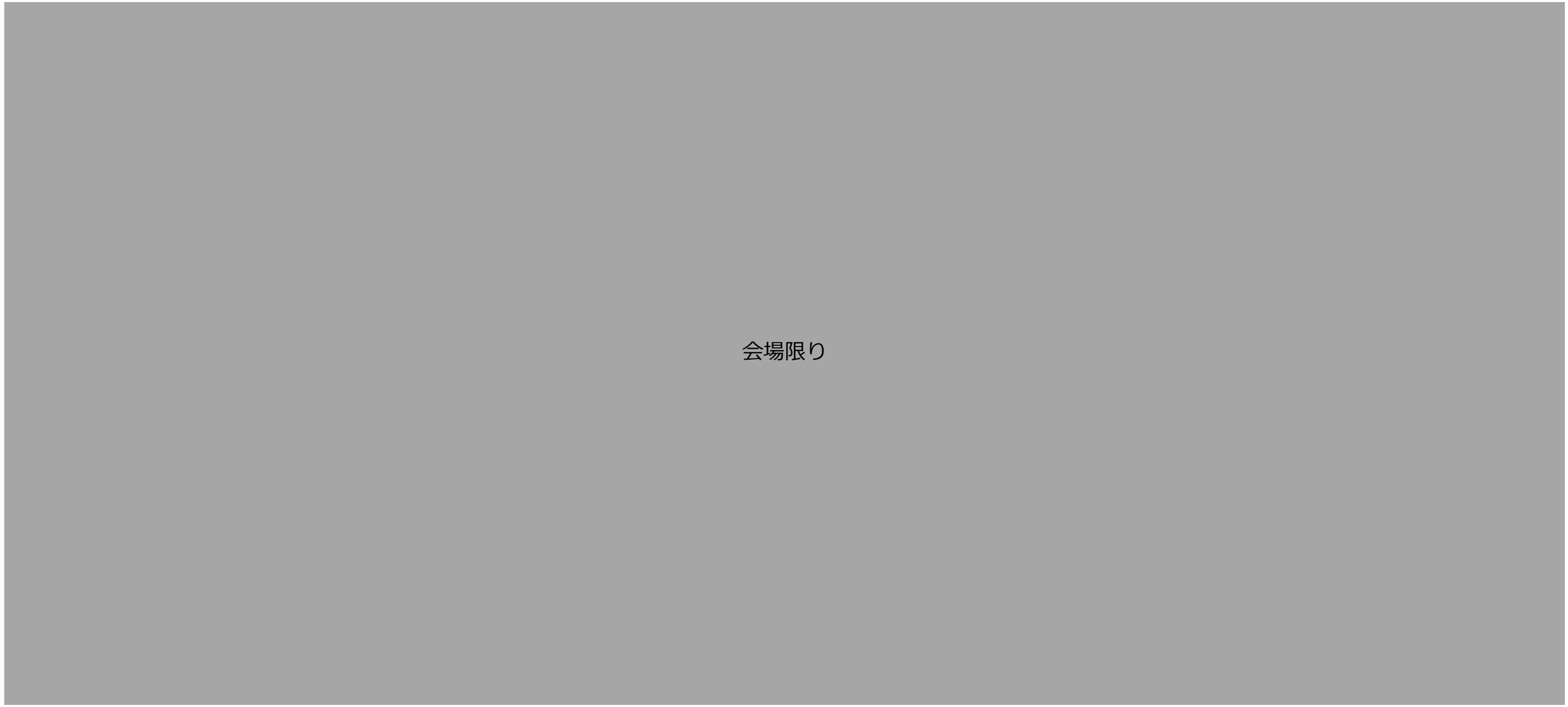
・GROUP BYでtagを指定でき、簡単にグルーピングが可能
  - (例)PPPoE/IPoE、IPv4/IPv6、サーバのロケーション

・WHERE句もtagを指定可能
  - 何をどう見たいかによってtag設計を行うと良い

最終的な可視化の様子 (1/2)



最終的な可視化の様子 (2/2)



今後の可視化について

通信の秘密における正当業務行為の範囲内で可視化項目を拡充予定
収集・可視化したデータを用いた高度なトラヒック分析も今後の目標

■追加で可視化したいもの

・FQDNの統計
- example.co.jpやexample.comのような単位(3LD、2LD)での合算
- 有名ドメインのTTLを延してもらうことでクエリ数を削減できないか？

・送信元IP毎のクエリ数
- 各々のユーザのクエリ数は増えているはず(中央値、平均値の分析が必要)
- また、ヘビーユーザの動向把握・攻撃クエリの検出にも必要

・ドメインに紐づくA/AAAAの情報
- 国内のCDNに正しくルーティングされているか？
- どれくらいの企業がCDNへ移行しているか？

・DNSのパケット長
- DNSの性能試験やAMP攻撃有無などの調査に活用

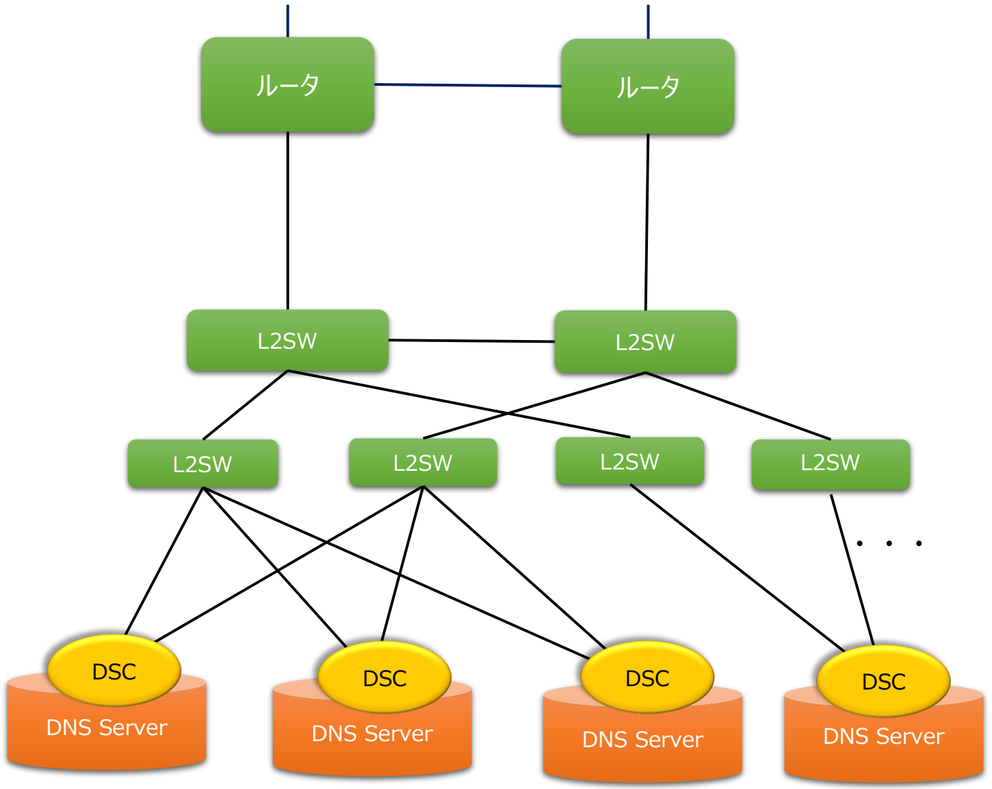
【参考】その他の可視化方法について
～ 他ISPのヒアリング内容の共有 ～

可視化の方式検討(再掲)

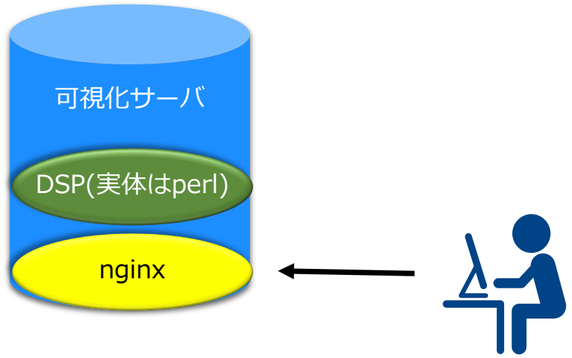
	Netflow方式	ポートミラー(NW上)	パケットキャプチャ(サーバ上)	クエリログ(dnstap)	DNSソフトウェアの統計	SNMP(LB上)
可視化ソフトウェアの例	-	DSC(DSP)	DSC(DSP)	-	Prometheus	Cacti
宛先IP毎に統計が取れるか	○	○	○	○	×	○
DNSサーバ上で作業が必要か	なし	なし	あり	あり	あり	なし
L7(DNS)情報が取れるか	×	○	○	○	○	×
TCPクエリ数が取れるか	○	○	○	○	○	△
送信元IP数が取れるか	○	○	○	○	×	×
DNS over HTTPS対応	×	×	×	○	○	×
キャッシュヒット率	×	×	×	×	○	×
評価	×	○	△	△	×	-

- 他ISPへヒアリングした結果、有用なツールをご紹介頂きました
- 弊社ラボ環境でも動作させてみましたので、その構成をご紹介致します

DSC(DSP)を用いた可視化システムの構成例



DNSサーバ(Node)毎にフォルダを用意し
サーバ毎にXMLファイルを格納



XMLファイル
または
JSONファイル

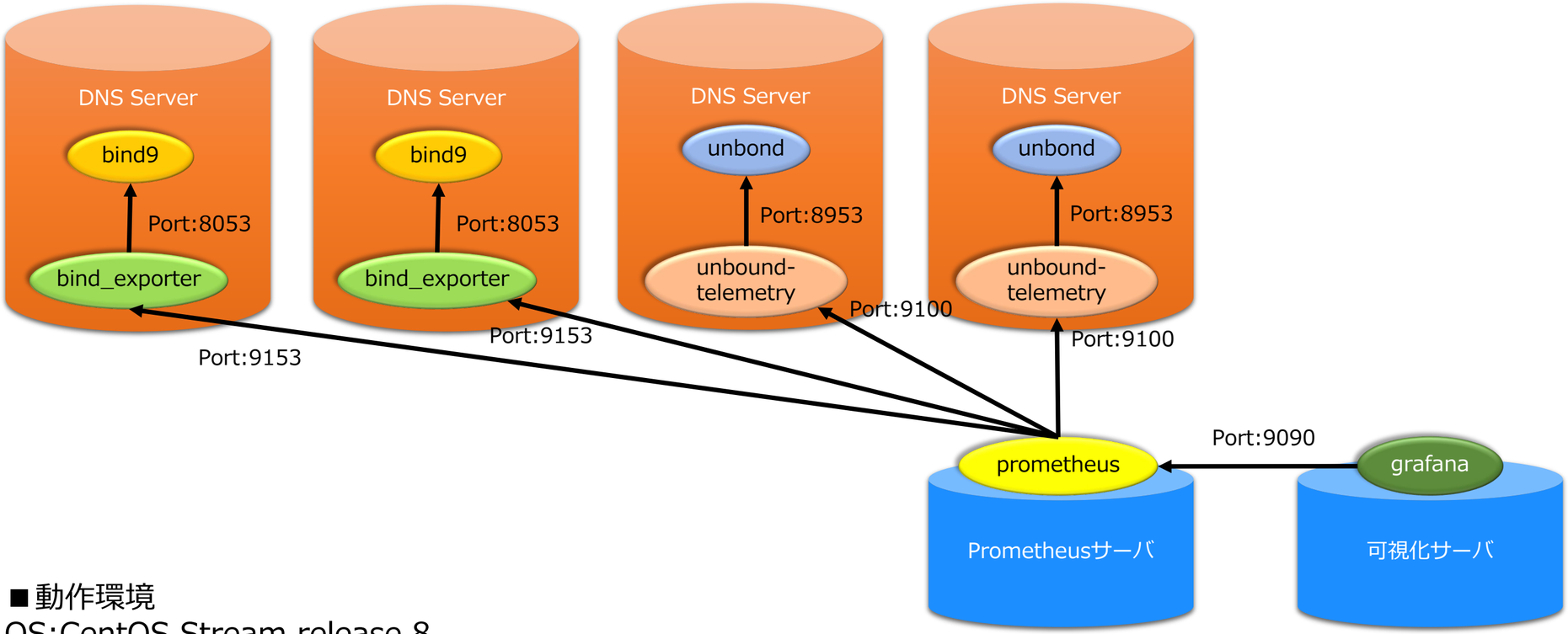
■動作環境
OS:Ubuntu 20.04.3 LTS
アプリ：dsc-2.11.2-1
　　dsp-2.0.1-1

・DSCは以下に対応
- IF指定のパケットキャプチャ
- pcapファイル(ファイル名指定)
- dnstap

・読み取ったパケット(クエリログ)を一定周期(デフォルト1分毎)で集計しファイルに出力

・集計結果はDSPというソフトで可視化(集計結果をパースして自身で可視化も可能)

DNSソフトウェアの統計ファイルをprometheusにて可視化



■動作環境
OS:CentOS Stream release 8
アプリ：　bind-9.11.26
unbound-1.7.3-17
bind_exporter-0.5.0
unbound-telemetry-0.1.0

・DNSソフトウェアであるbind9/unboundは統計結果をファイル出力可能(フォーマットはそれぞれ独自)

・bind9、unbound用のダッシュボードはサードパーティとして一般公開されているため、そちらを利用

・本方式ではサーバの負荷状況やキャッシュヒット率も同じポータルで手軽に可視化可能

参考URL

・DSC/DSPのインストール方法
https://github.com/DNS-OARC/dsp/wiki/All-in-One-Setup-Guide

・grafanaのダッシュボード(サードパーティ製)
bind9 exporter: https://grafana.com/grafana/dashboards/12309
unbound telemetry: https://grafana.com/grafana/dashboards/11705

■免責事項
上記は私が参考にしたサイトになりますが動作保証などは行っておりません。
不具合など発生した場合において弊社(私含め)は一切の責任を負いかねますのであらかじめご了承ください。

おわりに

議論したいこと/聞いてみたいこと

- NWやDNSトラヒックのうち何を可視化されていますか？
  - また、それを可視化したおかげで救われた事例はありますか？

- 大規模DNSトラヒックの可視化/解析はどのようにされていますか？
  - 本可視化システムではキャプチャカードを使い、また統計処理に特化させることにしました
  - クエリログを一旦全てDB等に格納して、その上で分析してる方はいらっしゃいますか？

- 可視化でお困りごとはありますか？
  - 私の場合、ツールの存在は知っていても実業務(可視化)に落とし込む際に苦労しました
  - 100万qps規模だとオンメモリでの処理やファイルの読み込み/移動だけでも考慮が必要となりました

【参考】通信の秘密に対しての考え方(ガイドラインより)

引用元：インターネットの安定的な運用に関する協議会
「電気通信事業者におけるサイバー攻撃等への対処と通信の秘密に関するガイドライン（第6版）」
https://www.jaipa.or.jp/other/mtcs/guideline_v6.pdf

(6) 網内トラヒックの現状把握

（ス） 現在、電話網におけるネットワークオペレーションセンターによる流量把握をIP網でも実施している。電話網であれば、（通信内容を含まない）共通信号線内の信号から、流量や流束の方向が把握可能であるが、IP通信の場合、パケットヘッダ部の統計データを取得する以外に手段がない。設備増強の必要性の判断や通信設備の障害発生時に障害の原因究明の円滑化などを目的として、網内トラヒックの現状把握をすべく統計データを取得してよいか。

【考え方】

「自社契約者から特定の ISP 別へのトラヒック及びその通信種別情報」及び「特定の ISP 別から自社契約者へのトラヒック及びその通信種別情報」を収集することは、個別の通信に係る送信元及び送信先IPアドレスを検知して利用しているため、通信の秘密の侵害（窃用等）に当たりうる。

ただし、設備増強の必要性の判断その他の自社業務を適正に遂行する等の業務目的のために、必要な範囲でそれらの情報を収集することは、正当業務行為として違法性が阻却される。

現状把握のための統計データ取得は
正当業務行為として違法性が阻却される
(ただし、必要な範囲内で行うこと)

ご清聴ありがとうございました