JANOG

宣言的(Declarative)ネットワーキング

OVN のコントロールプレーンの例

Jan. 22, 2020

CTO, North Asia (Japan, Korea and Greater China)

Motonori Shindo (@motonori_shindo)

vmware®

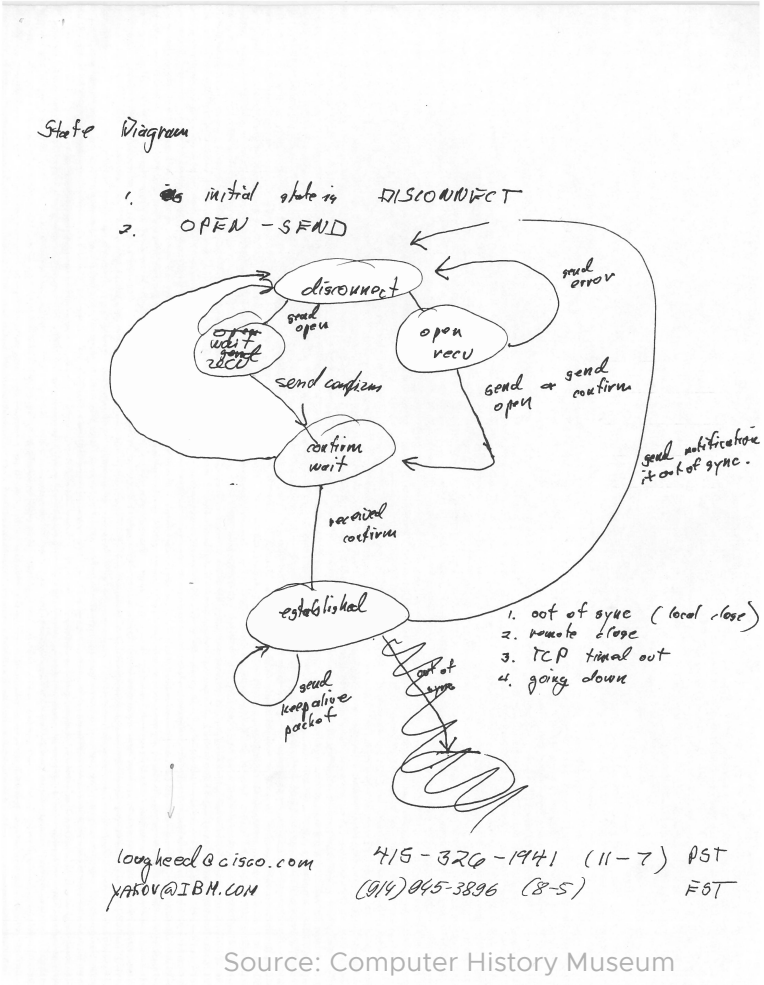
Nicira のコントローラ (NVP) はどう動いていたか？

Napkin Talk with Martin Casado

ネットワークは非常に複雑な分散型システム

- 非同期に起こる多重障害の可能性
- 一部の操作は冪等ではない

さまざまな実装を試したが、最もうまく動いたのは Dalalog をベースにした Nlog というエンジンを使った宣言的なコントローラであった。



Source: Computer History Museum

OVN (Open Virtual Network) とは

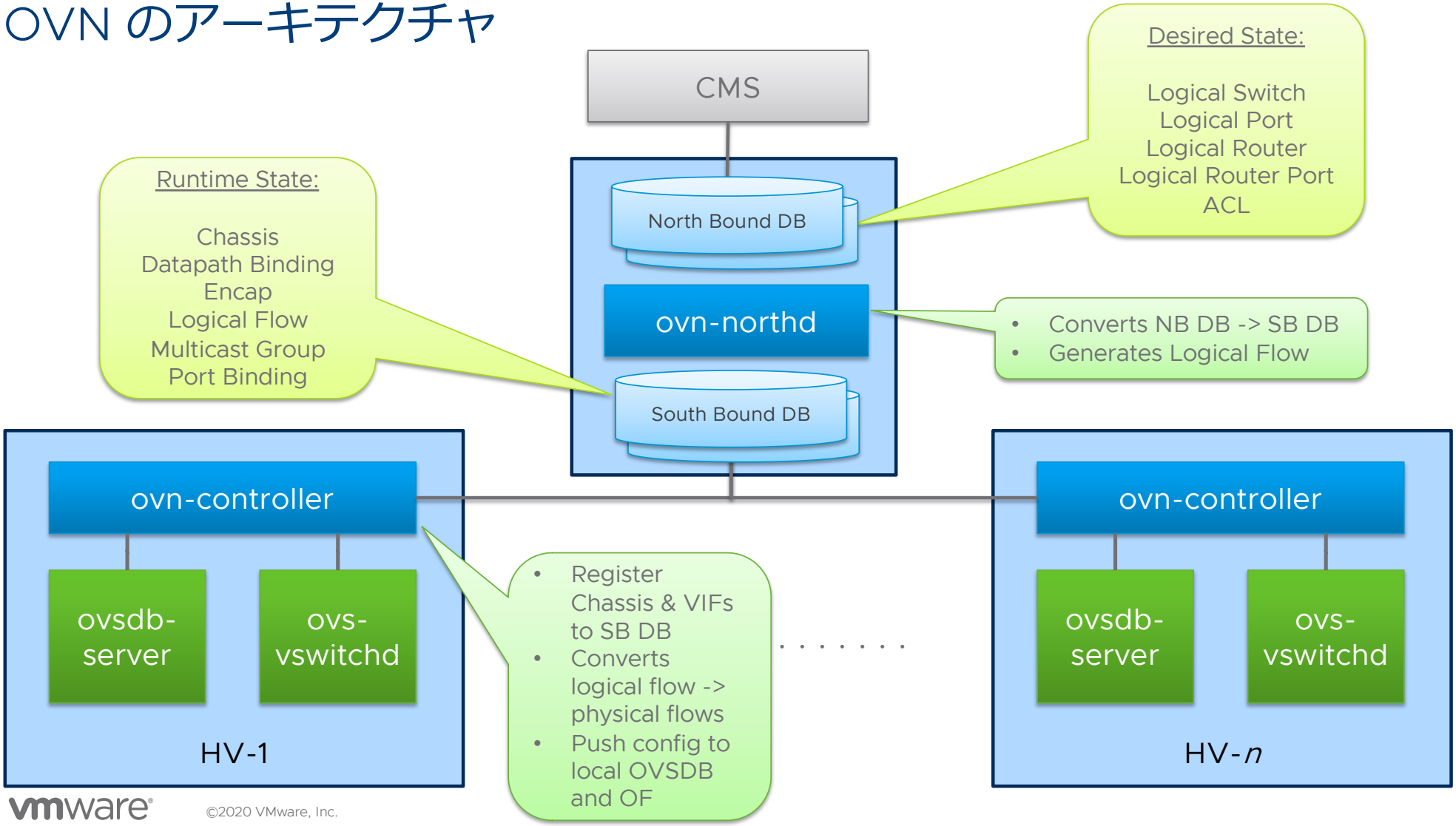
オープンソースな仮想ネットワークの実装

Open vSwitch (OVS) と共に開発されている（主に C で記述）

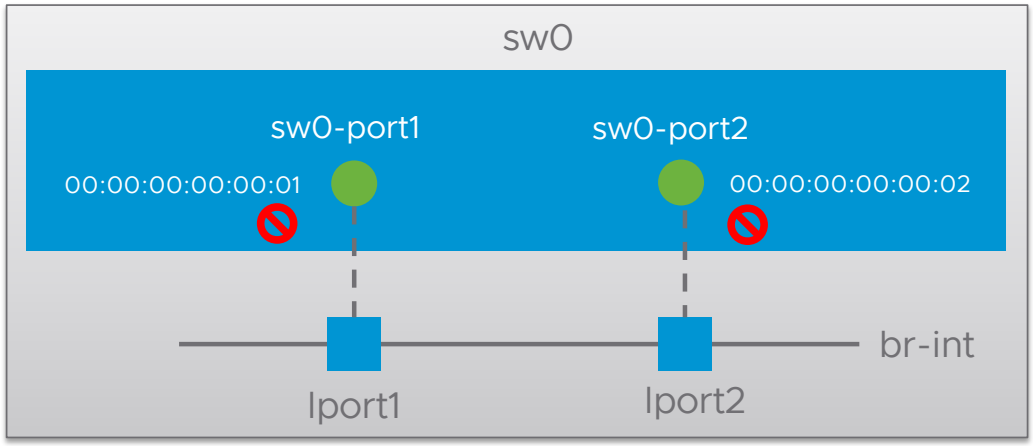
主な機能：

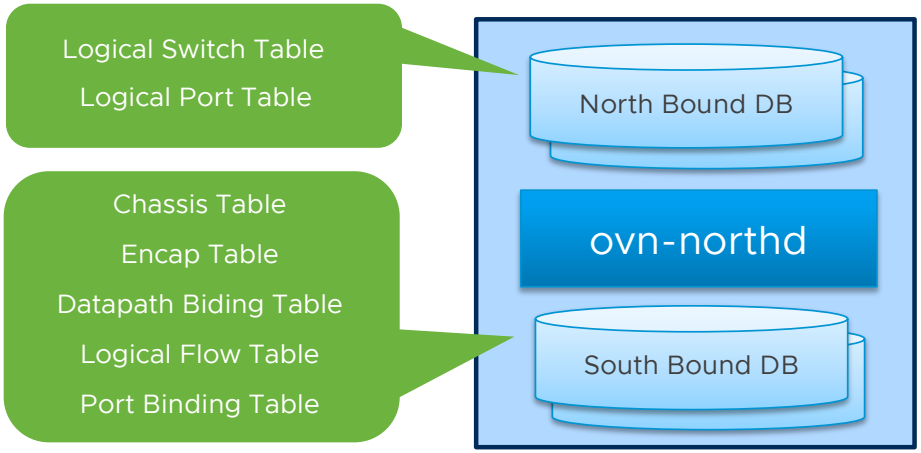
- 論理スイッチ
- 論理ルータ（IPv4、IPv6）
- オーバーレイネットワーク
- ACL
- Native NAT、Load Balancer、DHCP
- L2、L3 ゲートウェイ
- DPDK 対応
- Linux / Hyper-V で利用可能
- 複数のオーケストレータ（OpenStack、Kubernetes、Docker、Mesos、oVirt など）に対応

OVN のアーキテクチャ



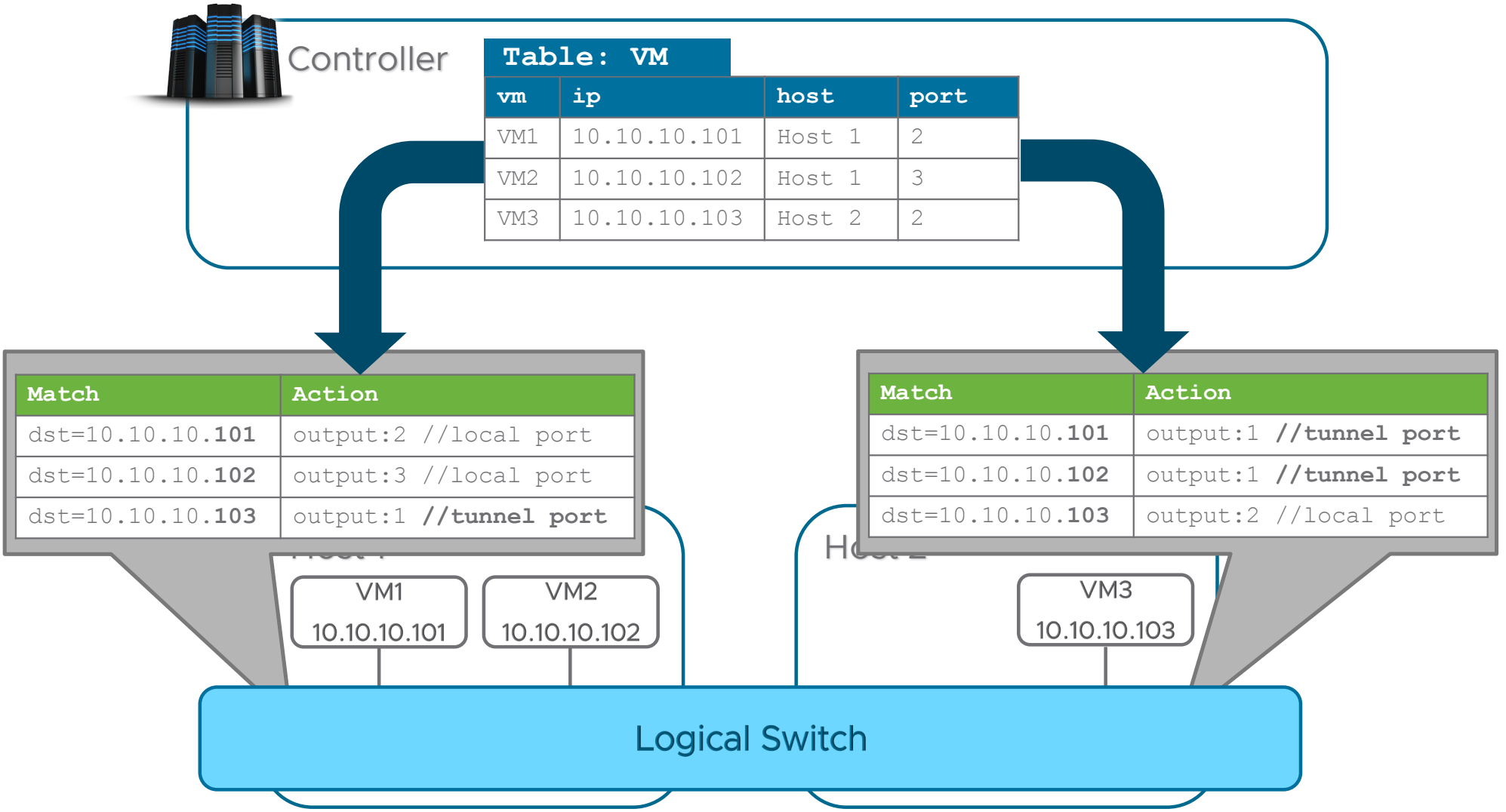
OVN 内部の動き






```
$ ovn-nbctl lswitch-add sw0                                          (論理スイッチ追加)
$ ovn-nbctl lport-add sw0 sw0-port1                                  (論理ポートの追加)
$ ovn-nbctl lport-add sw0 sw0-port2
$ ovn-nbctl lport-set-addresses sw0-port1 00:00:00:00:00:01          (MAC アドレスの設定)
$ ovn-nbctl lport-set-addresses sw0-port2 00:00:00:00:00:02
$ ovn-nbctl lport-set-port-security sw0-port1 00:00:00:00:00:01      (Port Security の設定)
$ ovn-nbctl lport-set-port-security sw0-port2 00:00:00:00:00:02      (Port Security の設定)
$ ovs-vsctl add-port br-int lport1 -- set Interface lport1 \         (br-int にポートを追加し、
    external_ids:iface-id=sw0-port1                                        インターフェースを接続)
$ ovs-vsctl add-port br-int lport2 -- set Interface lport2 \
    external_ids:iface-id=sw0-port2
```


VM / コンテナ環境は巨大かつ非常にダイナミック



Controller

Table: VM

vm	ip	host	port
VM1	10.10.10.101	Host 1	2
VM2	10.10.10.102	Host 1	3
VM3	10.10.10.103	Host 2	2

```
for h in Hosts:
  for vm in VM:
    if vm.host == h:
      h.add_flow(match="dst={vm.ip}",action="output:{vm.port}")
    else:
      h.add_flow(match="dst={vm.ip}",action="output:{tunport}")
```

Logical Switch

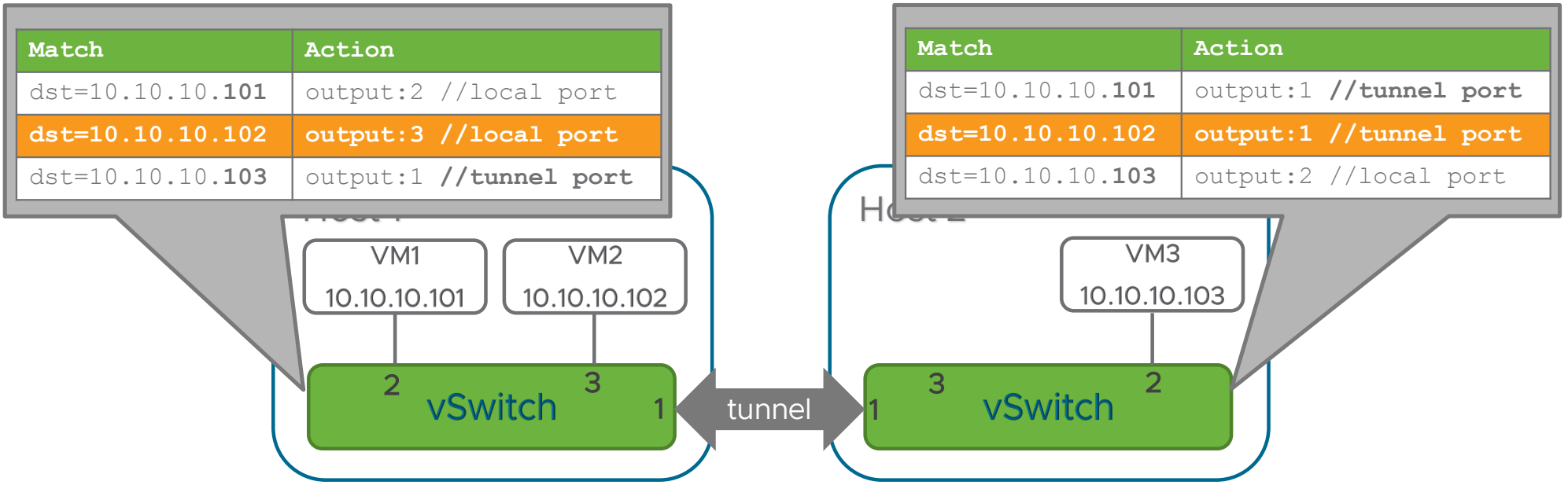
Controller

Table: VM

vm	ip	host	port
VM1	10.10.10.101	Host 1	2
VM2	10.10.10.102	Host 1	3
VM3	10.10.10.103	Host 2	2

Match	Action
dst=10.10.10.**101**	output:2 //local port
dst=10.10.10.102	**output:3 //local port**
dst=10.10.10.**103**	output:1 **//tunnel port**

Match	Action
dst=10.10.10.**101**	output:1 **//tunnel port**
dst=10.10.10.102	**output:1 //tunnel port**
dst=10.10.10.**103**	output:2 //local port



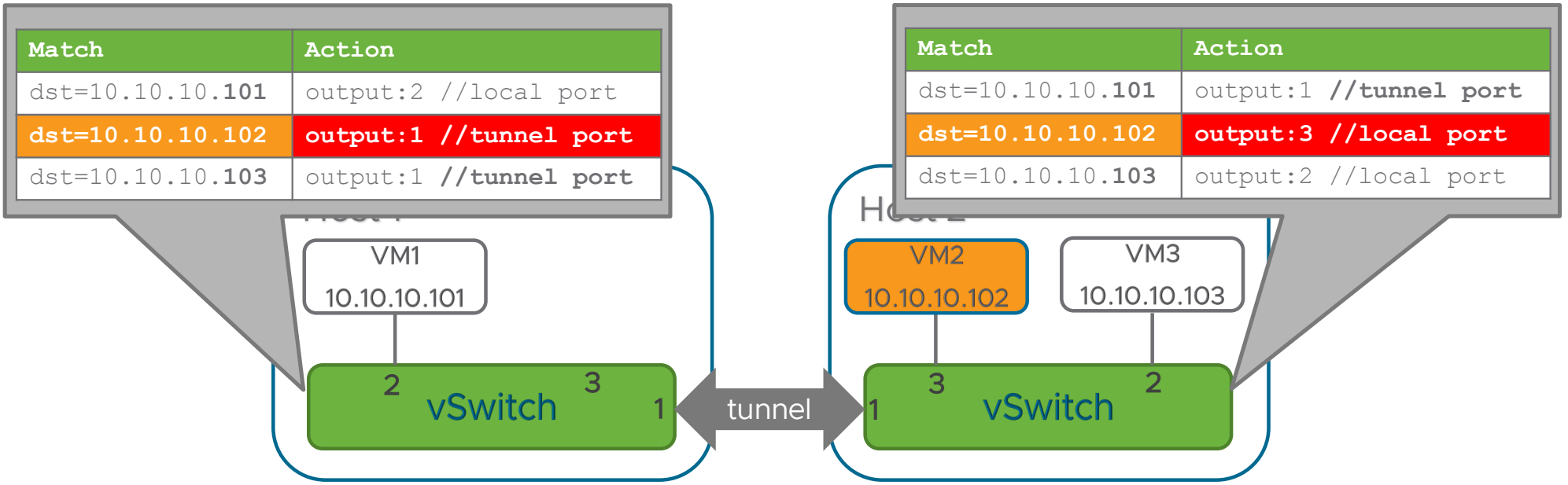
Controller

Table: VM

vm	ip	host	port
VM1	10.10.10.101	Host 1	2
VM2	10.10.10.102	Host 2	3
VM3	10.10.10.103	Host 2	2

Match	Action
dst=10.10.10.**101**	output:2 //local port
dst=10.10.10.102	**output:1 //tunnel port**
dst=10.10.10.**103**	output:1 **//tunnel port**

Match	Action
dst=10.10.10.**101**	output:1 **//tunnel port**
dst=10.10.10.102	**output:3 //local port**
dst=10.10.10.**103**	output:2 //local port



どのようにフローテーブルをアップデートするか？

案1： 全てのフローを1から再計算する

```
for h in Hosts:
  for vm in VM:
    if vm.host == h:
      h.add_flow(match="dst={vm.ip}",action="output:{vm.port}")
    else:
      h.add_flow(match="dst={vm.ip}",action="output:{tunport}")
```

- CPU サイクルの無駄遣い
- ネットワーク帯域の無駄遣い
- 設定遅延の増大

どのようにフローテーブルをアップデートするか？

案2： インクリメンタルな更新のアルゴリズムを作る

```
for h in Hosts:
  if old_vm.host == h:
    h.del_flow(…)
  else:
    h.del_flow(…)
  if new_vm.host == h:
    h.add_flow(match="dst={new_vm.ip}",
               action="output:{new_vm.port}")
  else:
    h.add_flow(match="dst={new_vm.ip}",
               action="output:{tunport}")
```

どのようにフローテーブルをアップデートするか？

案2： インクリメンタルな更新のアルゴリズムを作る

- **複雑になりがち**
  - エラー処理、部分障害、バッチ更新、等々…
- **テーブルごとに専用のアルゴリズムが必要**
  - ovn_northd には47個、NSX には数百のテーブルが存在する

```
for h in Hosts:
  if new_vm.host == h:
    h.add_flow(match="dst={new_vm.ip}",
               action="output:{new_vm.port}")
  else:
    h.add_flow(match="dst={new_vm.ip}",
               action="output:{tunport}")
```

解決策

- コントロール -> データプレンのマッピング設定を宣言的行う
- マッピングを差分更新するために汎用的な推論エンジンを用いる
- プログラマは差分更新について心配する必要なし

”Differential Datalog”

Datalog

一階述語論理にもとづいた宣言的論理プログラミング言語（DSL）

- しばしば Query Language として使われる

文法的には Prolog に似ているが、Prolog と違い、

- Function symbol がない
- 停止することが保証されている
- 説の順序は無関係
- リストの概念がない
- カット（！）や fail オペレータがない

Datalog 入門

Table: Parent

parent	child
Anakin	Luke
Anakin	Leia

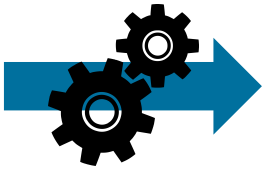


Table: Sibling

sibling1	sibling2
Luke	Leia
Leia	Luke

Head

Body

```
Sibling(c1,c2) :- Parent(p,c1),
                  Parent(p,c2),
                  c1 != c2.
```

Datalog 入門

Table: Host

id
1
2

Table: VM

vm	ip	host	port
VM1	10.10.10.101	1	2
VM2	10.10.10.102	1	3
VM3	10.10.10.103	2	2

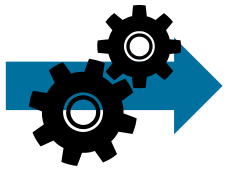


Table: Flow

host	match	action
1	dst=10.10.10.101	output:2
1	dst=10.10.10.102	output:3
1	dst=10.10.10.103	output:1
2	dst=10.10.10.101	output:1
2	dst=10.10.10.102	output:1
2	dst=10.10.10.103	output:2

Datalog 入門

Table: Host

id
1
2

Table: VM

vm	ip	host	port
VM1	10.10.10.101	1	2
VM2	**10.10.10.102**	2	3
VM3	10.10.10.103	2	2

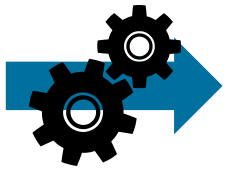


Table: Flow

host	match	action
1	dst=10.10.10.101	output:2
1	**dst=10.10.10.102**	**output:1**
1	dst=10.10.10.103	output:1
2	dst=10.10.10.101	output:1
1	**dst=10.10.10.102**	**output:3**
2	dst=10.10.10.103	output:2

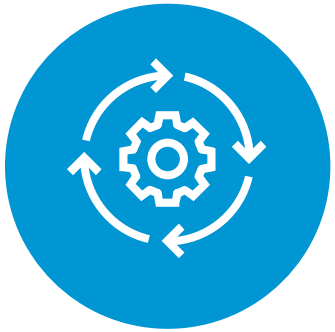
```
Flow(h,"dst={ip}","output:{p}") :-
    Host(h),
    VM(_, ip, h, p).
Flow(h,"dst={ip}","output:1") :-
    Host(h),
    VM(_, ip, other_host, _),
    other_host != h.
```

Differential Datalog (a.k.a. DDlog)

Differential Datalog	Differential Dataflow にもとづく Datalog エンジン
Differential Dataflow	大量のデータを効率よく処理し、任意の入力集合の変化に素早く対応することができるデータ並列プログラミングフレームワーク
Timely Dataflow	低遅延な巡回データフロー計算モデル

DDlog による効果

よりメンテナンスし易いコード



インクリメンタルな更新による性能の向上

C vs DDlog

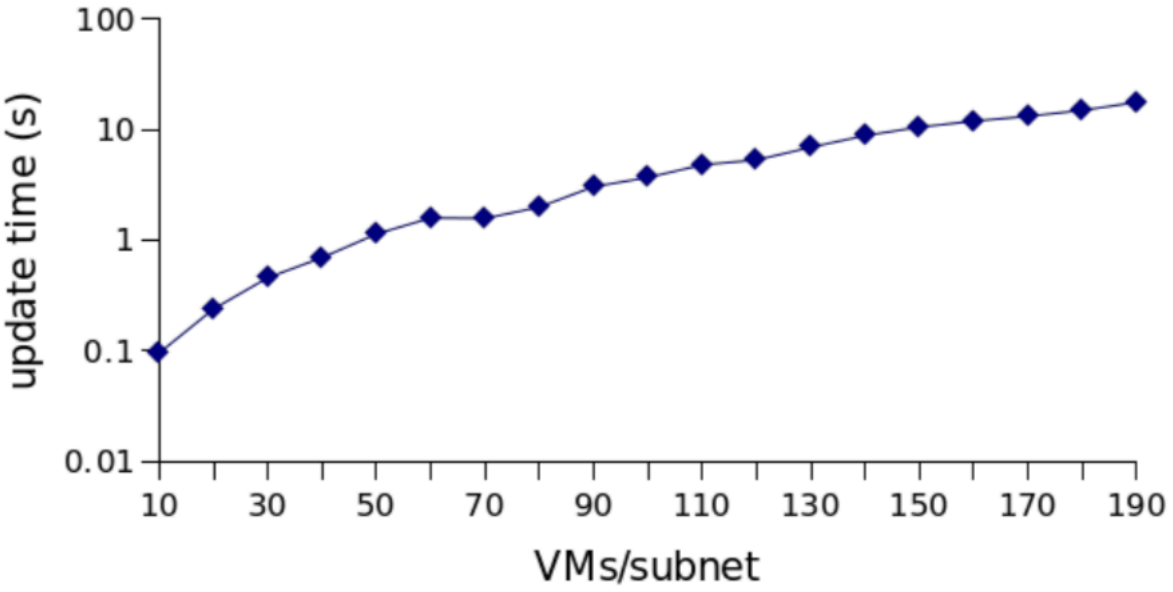
```
struct band_entry {
    int64_t rate;
    int64_t burst_size;
    const char *action;
};

static int
band_cmp(const void *band1_, const void *band2_)
{
    const struct band_entry *band1p = band1_;
    const struct band_entry *band2p = band2_;

    if (band1p->rate != band2p->rate) {
        return band1p->rate > band2p->rate ? -1 : 1;
    } else if (band1p->burst_size != band2p->burst_size) {
        return band1p->burst_size > band2p->burst_size ? -1 : 1;
    } else {
        return strcmp(band1p->action, band2p->action);
    }
}

static bool
bands_need_update(const struct nbrec_meter *nb_meter,
                  const struct sbrec_meter *sb_meter)
{
    if (nb_meter->n_bands != sb_meter->n_bands) {
        return true;
    }

    /* A single band is the most common scenario, so speed up that
     * check. */
    if (nb_meter->n_bands == 1) {
        struct nbrec_meter_band *nb_band = nb_meter->bands[0];
        struct sbrec_meter_band *sb_band = sb_meter->bands[0];

        return !(nb_band->rate == sb_band->rate
                 && nb_band->burst_size == sb_band->burst_size
                 && !strcmp(sb_band->action, nb_band->action));
    }

    /* Place the Northbound entries in sorted order. */
    struct band_entry *nb_bands;
    nb_bands = xmalloc(sizeof *nb_bands * nb_meter->n_bands);
    for (size_t i = 0; i < nb_meter->n_bands; i++) {
        struct nbrec_meter_band *nb_band = nb_meter->bands[i];

        nb_bands[i].rate = nb_band->rate;
        nb_bands[i].burst_size = nb_band->burst_size;
        nb_bands[i].action = nb_band->action;
    }
    qsort(nb_bands, nb_meter->n_bands, sizeof *nb_bands, band_cmp);

    /* Place the Southbound entries in sorted order. */
    struct band_entry *sb_bands;
    sb_bands = xmalloc(sizeof *sb_bands * sb_meter->n_bands);
    for (size_t i = 0; i < sb_meter->n_bands; i++) {
        struct sbrec_meter_band *sb_band = sb_meter->bands[i];

        sb_bands[i].rate = sb_band->rate;
        sb_bands[i].burst_size = sb_band->burst_size;
        sb_bands[i].action = sb_band->action;
    }
    qsort(sb_bands, sb_meter->n_bands, sizeof *sb_bands, band_cmp);

    bool need_update = false;
    for (size_t i = 0; i < nb_meter->n_bands; i++) {
        if (nb_bands[i].rate != sb_bands[i].rate
            || nb_bands[i].burst_size != sb_bands[i].burst_size
            || strcmp(nb_bands[i].action, nb_bands[i].action)) {
            need_update = true;
            goto done;
        }
    }

done:
    free(nb_bands);
    free(sb_bands);

    return need_update;
}
```

```
/* Each entry in the Meter and Meter_Band tables in OVN_Northbound have
 * a corresponding entries in the Meter and Meter_Band tables in
 * OVN_Southbound.
 */
static void
sync_meters(struct northd_context *ctx)
{
    struct shash sb_meters = SHASH_INITIALIZER(&sb_meters);

    const struct sbrec_meter *sb_meter;
    SBREC_METER_FOR_EACH (sb_meter, ctx->ovnsb_idl) {
        shash_add(&sb_meters, sb_meter->name, sb_meter);
    }

    const struct nbrec_meter *nb_meter;
    NBREC_METER_FOR_EACH (nb_meter, ctx->ovnnb_idl) {
        bool new_sb_meter = false;

        sb_meter = shash_find_and_delete(&sb_meters, nb_meter->name);
        if (!sb_meter) {
            sb_meter = sbrec_meter_insert(ctx->ovnsb_txn);
            sbrec_meter_set_name(sb_meter, nb_meter->name);
            new_sb_meter = true;
        }

        if (new_sb_meter || bands_need_update(nb_meter, sb_meter)) {
            struct sbrec_meter_band **sb_bands;
            sb_bands = xcalloc(nb_meter->n_bands, sizeof *sb_bands);
            for (size_t i = 0; i < nb_meter->n_bands; i++) {
                const struct nbrec_meter_band *nb_band = nb_meter->bands[i];

                sb_bands[i] = sbrec_meter_band_insert(ctx->ovnsb_txn);

                sbrec_meter_band_set_action(sb_bands[i], nb_band->action);
                sbrec_meter_band_set_rate(sb_bands[i], nb_band->rate);
                sbrec_meter_band_set_burst_size(sb_bands[i],
                                                nb_band->burst_size);
            }
            sbrec_meter_set_bands(sb_meter, sb_bands, nb_meter->n_bands);
            free(sb_bands);
        }

        sbrec_meter_set_unit(sb_meter, nb_meter->unit);
    }

    struct shash_node *node, *next;
    SHASH_FOR_EACH_SAFE (node, next, &sb_meters) {
        sbrec_meter_delete(node->data);
        shash_delete(&sb_meters, node);
    }
    shash_destroy(&sb_meters);
}
```

```
/* Meter_Band table */
for (mb in nb.Meter_Band) {
    sb.Out_Meter_Band(.uuid_name = uuid2str(mb._uuid),
                      .action = mb.action,
                      .rate = mb.rate,
                      .burst_size = mb.burst_size)
}

/* Meter table */
for (meter in nb.Meter) {
    sb.Out_Meter(.name = meter.name,
                 .unit = meter.unit,
                 .bands = set_map_uuid2str(meter.bands))
}
```

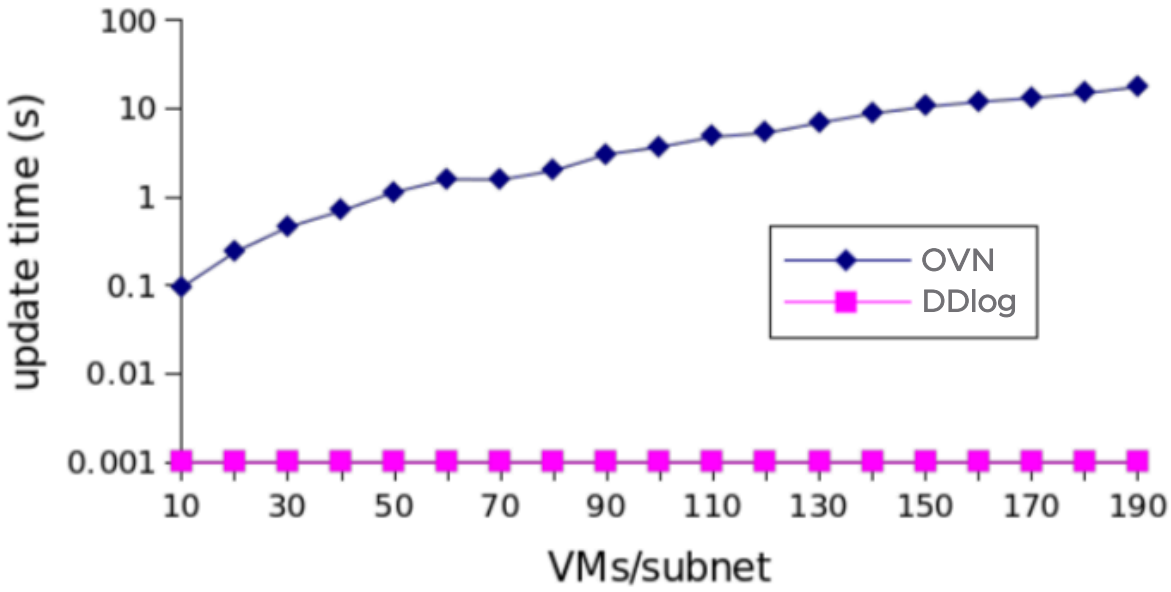
DDlog によるパフォーマンスの改善

論理ポート追加にかかる時間



DDlog によるパフォーマンスの改善

論理ポート追加にかかる時間



今後の予定と課題

予定

- ovn-northd と ovn-controller のロジックを DDlog で実装
- Test、Test、and Test !!
  - C バージョンと DDlog バージョンで同じフローテーブルが計算されるかを比較
- NSX-T での適用？

課題

- 新しいプログラミング・パラダイムの学習コスト
- 新しいツールチェーン（Rust、Haskel、デバッガ、etc.）
- C 実装への追従

References

- “RFC: incremental computation for OVN with DDlog”, discussion on ovs-discuss mailing list,
  - https://mail.openvswitch.org/pipermail/ovs-discuss/2018-November/047665.html
- “OVN Controller Incremental Processing”, Han Zhou, Open vSwitch 2018 Fall Conference
  - http://www.openvswitch.org/support/ovscon2018/6/1350-zhou.pptx
- ”Toward Leaner, Faster ovn-northd”, Leonid Ryzhyk, OVS Orbit Episode 58
  - https://ovsorbit.org/#e58
- “DDlog in OVN – Current State and Future Challenges”, OVS and OVN 2019 Fall Conference
  - https://www.youtube.com/watch?v=Sf6QjyoZk34&feature=youtu.be
- Differential Datalog
  - https://github.com/ryzhyk/differential-datalog
  - https://research.vmware.com/files/attachments/0/0/0/0/0/8/8/ddlog.pdf
- Timely Dataflow & Differential Dataflow
  - https://github.com/TimelyDataflow
  - https://www.youtube.com/watch?v=nRp3EQy6ccw

Thank You