High Availability Software Update for IRR

Yasuhiro Shirasaki

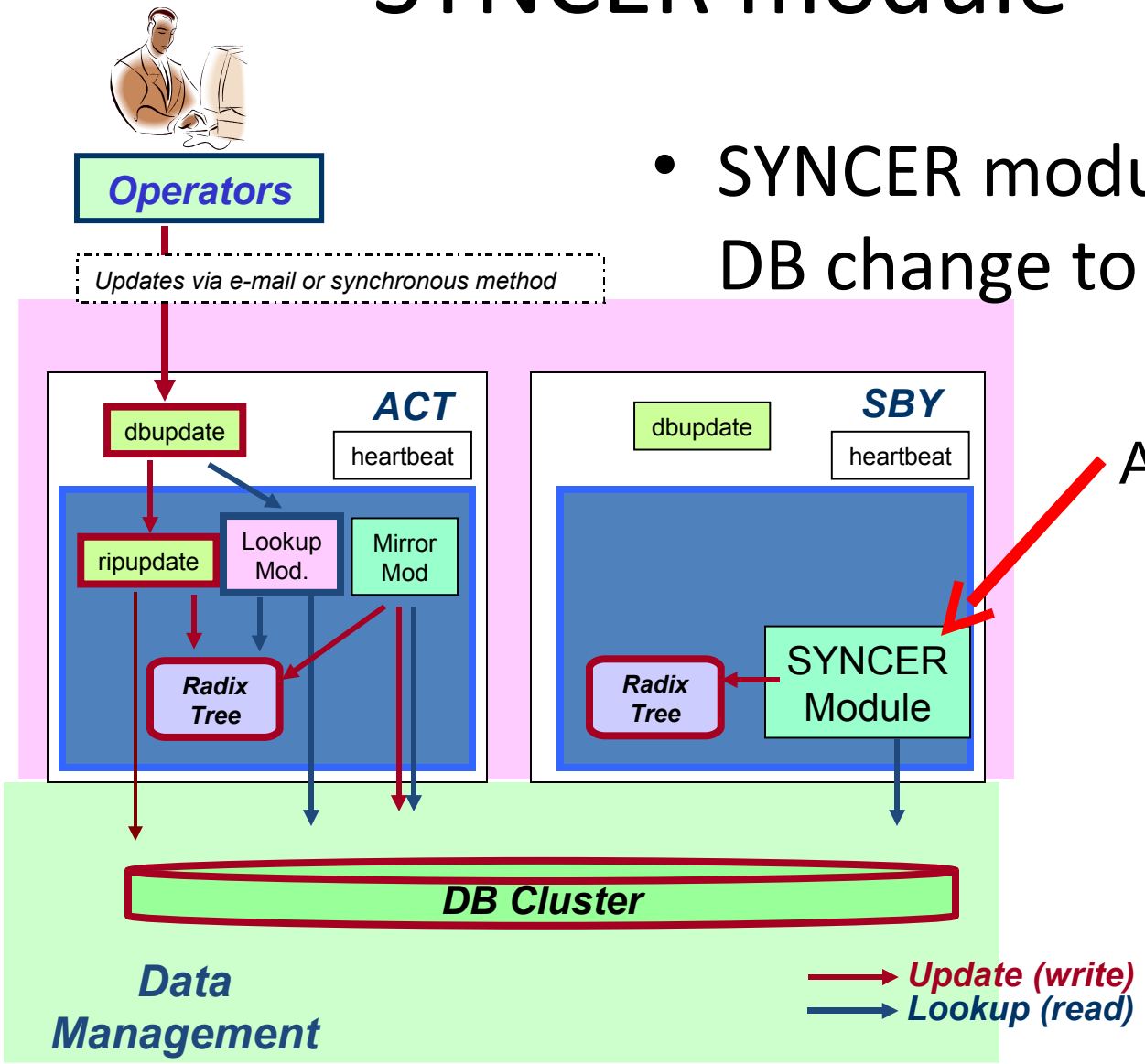
Tomoya Yoshida

Anti Prefix Hijacking Project / NTT Communications

High Availability IRR Software Development

- We use RIPE whois-server software for IRR
  - As core module of anti BGP prefix hijacking system
- We are working on
  - SYNCER module (Redundancy, Radix Tree)
  - Minimizing down time
  - Optimization (SQL)

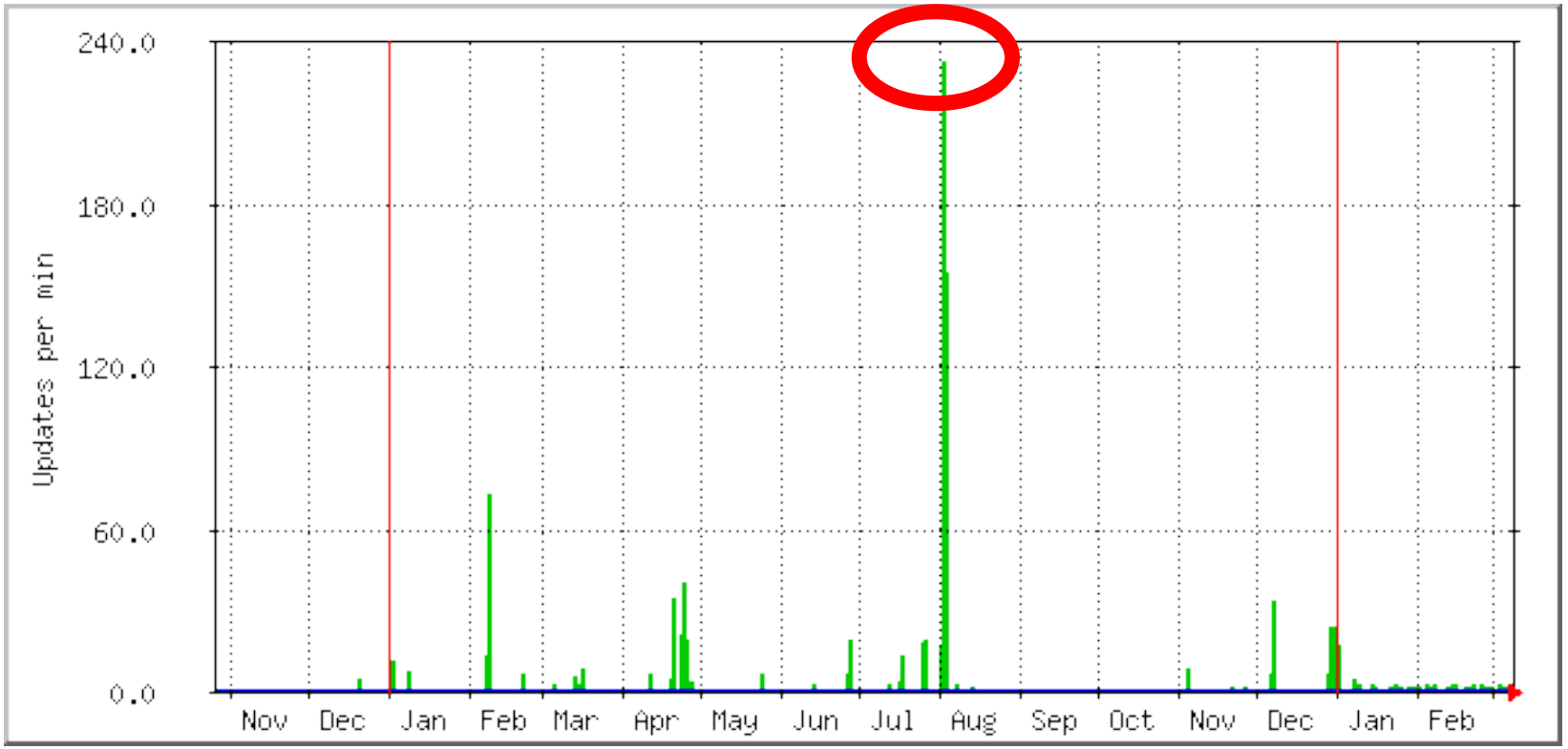
SYNCER module



- SYNCER module reflects DB change to Radix Tree

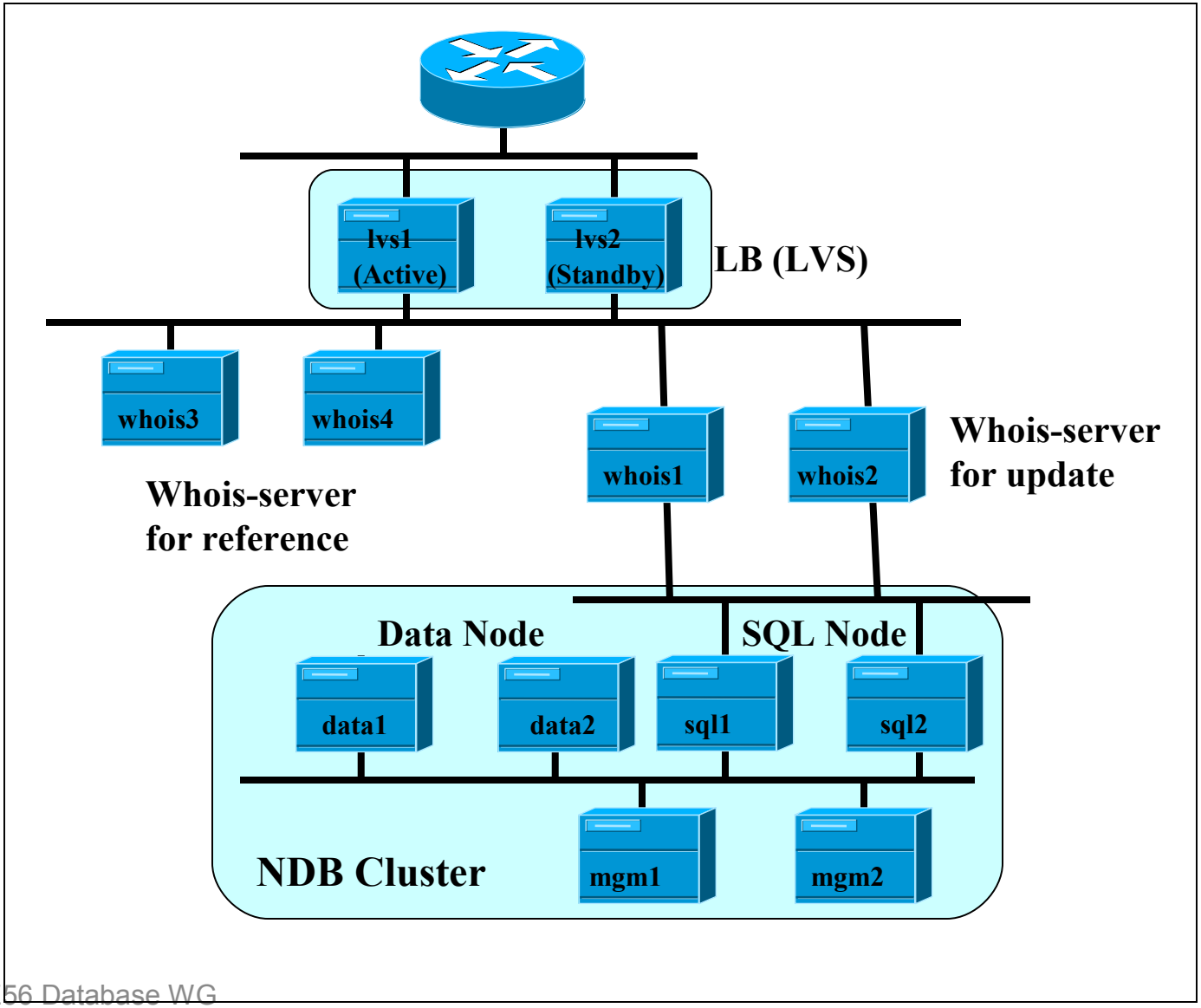
Whois Update Statistics

- RIPE whois
  - Peak: 240 updates/minute (2updates/min average)

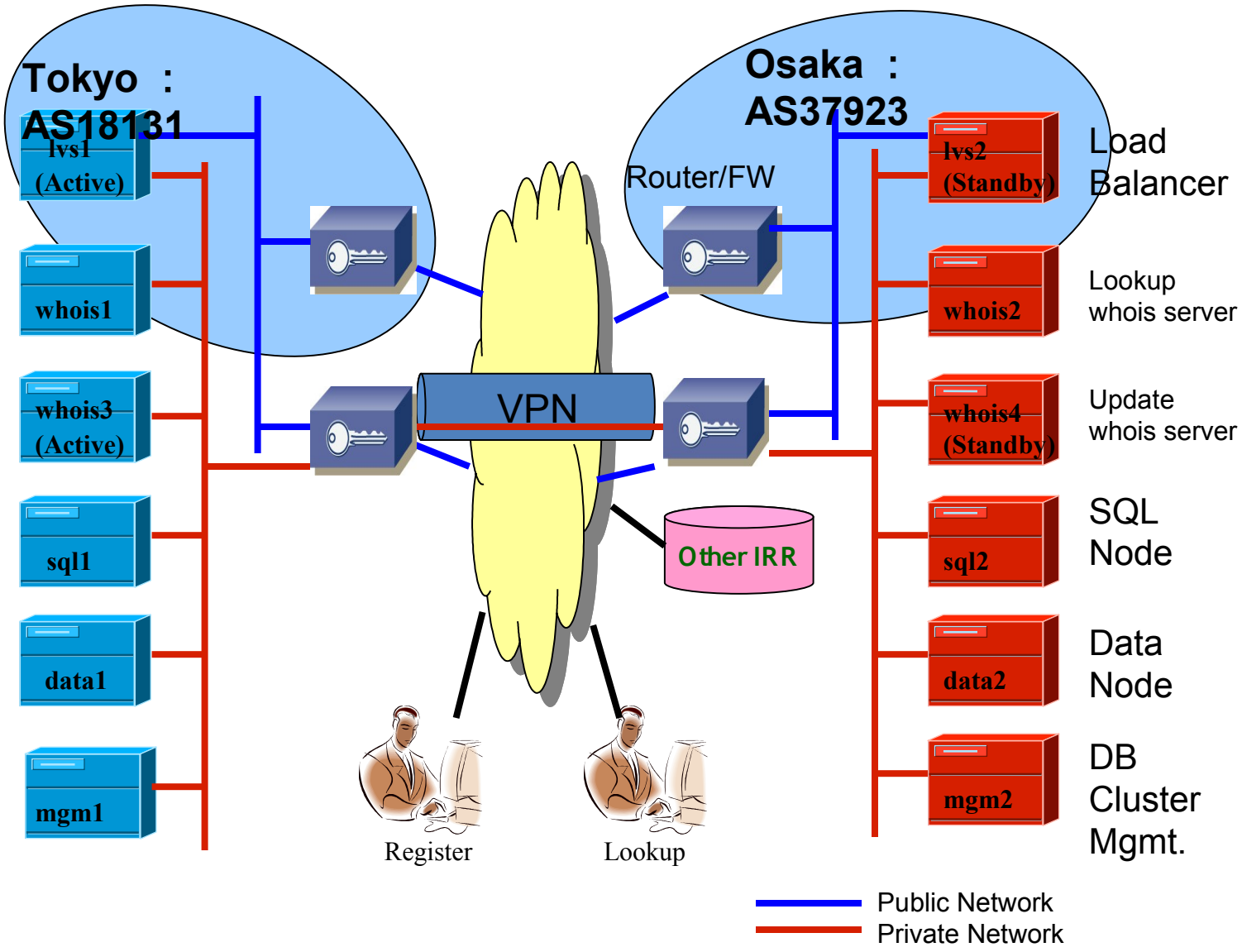


RIPE whois database statistics
http://www.ripe.net/info/stats/db/mrtg/updaterate.html

Single Site Topology



Dual Site Topology



Performance Measurement

- Addition and deletion
  - With hundreds of route objects

```
route:          10.99.1.0/24
descr:          IRR Experiment routes
origin:         AS65000
mnt-by:         IAC-SAMPLE-MNT
changed:        yasuhiro@knight.route-police.jp 20071125
source:         SAMPLE

route:          10.99.2.0/24
descr:          IRR Experiment routes
origin:         AS65000
mnt-by:         IAC-SAMPLE-MNT
changed:        yasuhiro@knight.route-police.jp 20071125
source:         SAMPLE
```

Results

- Single site

	RIPE DB		Single Site
	Peak	Ave.	
ADD	240	2	198
DEL	240	2	300

(Objects / min)

- Dual sites
  - Synchronous update

Dual Site	
Normal DB	Modified DB
20	26
33	39

  - Distance: 500km, RTT: 8msec

Down time minimization

- Radix Tree (RX) rebuilding issue
  - Rebuilding RX in a boot sequence takes more than 5 minutes for RIPE-DB
  - Service interruption by blocking DB updates while rebuilding RX
  - NDBCLUSTER storage engine doesn’t support Multi Version Concurrency Control (MVCC) mechanism
    - No “snapshot”
    - Future planned
- Current Workaround
  - Make a copy of table in MEMORY storage from slow NDBCLUSTER storage to minimize service interruption

Results

		Test1	Test2	Test3	Test4	Test5	Ave.
Original Code	LOCK duration	29.6	29.3	29.9	29.5	29.1	29.5
	RX rebuild	29.7	29.3	29.9	29.5	29.2	29.5
Modified Code	LOCK duration	26.1	25.7	25.1	24.7	25.5	25.4
	RX rebuild	30.7	30.1	29.6	29.1	30.0	29.9

Unit: seconds

- 14% shorter locking time with RADB
  - Total time of RX rebuild is not change
- Requires more memories on SQL node

SQL Optimizations

- Whois-server consists ~ 60 SQL queries
  - We tested with NDB Cluster
  - 3 Optimizations
    - BEGIN-COMMIT Transaction
    - Omit useless sub-table creation
    - Optimization of id_table creation

Base SQL

```
DROP TABLE IF EXISTS ID_63097_1082161504;

CREATE TEMPORARY TABLE ID_63097_1082161504 ( id INT PRIMARY KEY NOT NULL, recursive BOOL DEFAULT
    0 ) TYPE=HEAP;

DROP TABLE IF EXISTS ID_63097_1082161504_S;
CREATE TEMPORARY TABLE ID_63097_1082161504_S  ( id int ) TYPE=HEAP;
DROP TABLE IF EXISTS ID_63097_1082161504_S;

INSERT INTO ID_63097_1082161504 values (3962212,0);

ALTER TABLE ID_63097_1082161504 ADD COLUMN gid INT NOT NULL DEFAULT 0;
UPDATE ID_63097_1082161504 SET gid=id;
SELECT STRAIGHT_JOIN last.object_id, last.sequence_id, last.object, last.object_type, last.pkey, recursive, gid
    FROM ID_63097_1082161504 IDS, last, last glast, object_order, object_order gorder WHERE
    (IDS.gid=glast.object_id AND glast.object_type=gorder.object_type) AND (IDS.id=last.object_id AND
    last.object_type=object_order.object_type) ORDER BY  recursive, object_order.order_code;
DROP TABLE IF EXISTS ID_63097_1082161504;
```

Wrap with BEGIN-COMMIT

```
BEGIN;
DROP TABLE IF EXISTS ID_63097_1082161504;

CREATE TEMPORARY TABLE ID_63097_1082161504 ( id INT PRIMARY KEY NOT NULL, recursive BOOL DEFAULT
    0 ) TYPE=HEAP;

DROP TABLE IF EXISTS ID_63097_1082161504_S;
CREATE TEMPORARY TABLE ID_63097_1082161504_S  ( id int ) TYPE=HEAP;
DROP TABLE IF EXISTS ID_63097_1082161504_S;

INSERT INTO ID_63097_1082161504 values (3962212,0);

ALTER TABLE ID_63097_1082161504 ADD COLUMN gid INT NOT NULL DEFAULT 0;
UPDATE ID_63097_1082161504 SET gid=id;
SELECT STRAIGHT_JOIN last.object_id, last.sequence_id, last.object, last.object_type, last.pkey, recursive, gid
    FROM ID_63097_1082161504 IDS, last, last glast, object_order, object_order gorder WHERE
    (IDS.gid=glast.object_id AND glast.object_type=gorder.object_type) AND (IDS.id=last.object_id AND
    last.object_type=object_order.object_type) ORDER BY  recursive, object_order.order_code;
DROP TABLE IF EXISTS ID_63097_1082161504;
COMMIT;
```

IMPACT: 1.4% faster than base code

Omit useless sub table

```
BEGIN;
DROP TABLE IF EXISTS ID_63097_1082161504;

CREATE TEMPORARY TABLE ID_63097_1082161504 ( id INT PRIMARY KEY NOT NULL, recursive BOOL DEFAULT
    0 ) TYPE=HEAP;
# Omit useless subtable
# DROP TABLE IF EXISTS ID_63097_1082161504_S;
# CREATE TEMPORARY TABLE ID_63097_1082161504_S  ( id int ) TYPE=HEAP;
# DROP TABLE IF EXISTS ID_63097_1082161504_S;

INSERT INTO ID_63097_1082161504 values (3962212,0);

ALTER TABLE ID_63097_1082161504 ADD COLUMN gid INT NOT NULL DEFAULT 0;
UPDATE ID_63097_1082161504 SET gid=id;
SELECT STRAIGHT_JOIN last.object_id, last.sequence_id, last.object, last.object_type, last.pkey, recursive, gid
    FROM ID_63097_1082161504 IDS, last, last glast, object_order, object_order gorder WHERE
    (IDS.gid=glast.object_id AND glast.object_type=gorder.object_type) AND (IDS.id=last.object_id AND
    last.object_type=object_order.object_type) ORDER BY  recursive, object_order.order_code;
DROP TABLE IF EXISTS ID_63097_1082161504;
COMMIT;
```

IMPACT: 20.8% faster than base code

Optimization of id_table creation

```
BEGIN;
DROP TABLE IF EXISTS ID_63097_1082161504;
# Added gid INT
CREATE TEMPORARY TABLE ID_63097_1082161504 ( id INT PRIMARY KEY NOT NULL, recursive BOOL DEFAULT
    0, gid INT NOT NULL DEFAULT 0 ) TYPE=HEAP;
# Omit useless subtable
# DROP TABLE IF EXISTS ID_63097_1082161504_S;
# CREATE TEMPORARY TABLE ID_63097_1082161504_S  ( id int ) TYPE=HEAP;
# DROP TABLE IF EXISTS ID_63097_1082161504_S;
# Insert gid here
INSERT INTO ID_63097_1082161504 values (3962212,0,3962212);
# gid is already added
# ALTER TABLE ID_63097_1082161504 ADD COLUMN gid INT NOT NULL DEFAULT 0;
# UPDATE ID_63097_1082161504 SET gid=id;
SELECT STRAIGHT_JOIN last.object_id, last.sequence_id, last.object, last.object_type, last.pkey, recursive, gid
    FROM ID_63097_1082161504 IDS, last, last glast, object_order, object_order gorder WHERE
    (IDS.gid=glast.object_id AND glast.object_type=gorder.object_type) AND (IDS.id=last.object_id AND
    last.object_type=object_order.object_type) ORDER BY  recursive, object_order.order_code;
DROP TABLE IF EXISTS ID_63097_1082161504;
COMMIT;
```

IMPACT: 24% faster than base code

Results

- ADD: 150%
- DELETE: 48% faster than the base code
  - On single site configuration

	Base code	Optimized
ADD	156	390
DEL	370	550

(Objects / min)

Next step

- We are happy to share these code to improve the RIPE DB
- Evaluation by RIPE NCC DB team would be appreciated ☺

Thank you

Yasuhiro Shirasaki
yasuhiro@nttv6.jp

Tomoya Yoshida
yoshida@nttv6.jp