RISng - Technical Overview

James Aldridge, Daniel Karrenberg, Henk Uijterwaal, Arife Vural, Matthew Williams

RIPE NCC New Projects Group
Contents

• “RIS Classic” - The original version
• Problems
• Potential solutions
• “RISng”
• Work in progress
• Summary
• Questions
“RIS Classic” - Overview

BGPd → Periodic dumps of RIB and Updates

Raw Data → rsync → MySQL

abcoude

dbinsert

rsync

Collect & Schedule

MySQL → UI

rrcXX

rsync

Raw Data → Replication

MySQL

halfweg

Collect & Schedule
“RIS Classic”

- RRCs only run Zebra bgpd
  - All other processing done centrally
- All database insertion done centrally
  - abcoude.ripe.net
  - Solaris / Sun SPARC Ultra Enterprise 420R
  - MySQL insertion script written in Perl
    - Text output from “route_btoa -m” (Update dumps)
    - Binary processing (RIB dumps)
- Front-end server handles user interface, etc
  - halfweg.ripe.net
  - Linux
  - Replicated copy of database
Problems

• Database insertion of data from 9 route collectors on a single central machine is slow
  – Little headroom to allow for abnormal cases
  – Can sometimes take more than 24 hours to insert a single day’s data
  – Little capacity to add more RRCs or full BGP feeds

• Limited attributes are stored in the database:
  – Only first 255 characters of AS Path stored
  – Other BGP attributes (communities, MEDs, etc.) ignored
“RIS Classic”
Database insertion times

>24 hours to insert one day’s data
Considered Solutions

- New, faster hardware to replace or supplement abcoude.ripe.net
- Make better use of existing hardware
- Database redesign
- Use other database than MySQL

- But we don’t want to spend more money than necessary

- RISng is the result…
“RISng”

• **Aims**
  – Improve scalability
  – Easier software maintenance
  – Store more complete route attributes

• **New database structure**
  – Remove arbitrary limit on AS Path
  – Store additional attributes

• **Perform database insertion locally on (otherwise mostly idle) route collectors**

• **New database insertion process**
  – written in C instead of Perl
Single software version

- Up to now, software maintenance has been more difficult than necessary
  - 6 different versions of FreeBSD (FreeBSD 3.5 onwards)
  - Almost every individual RRC needs its own software build
- Aim to bring all RRCs to the same OS version (FreeBSD 4.6.2)
  - Bootable CDROM created and shipped to hosts
  - Contains a snapshot of complete RRC system
  - Allows remote (SSH) access for manual configuration and subsequent maintenance if necessary
- Upgrades are in progress (3 out of 9 RRCs upgraded so far)
  - But some boxes have problems:
    - RRC01 (@ LINX) fails to boot the new FreeBSD kernel
    - RRC04 (@CERN) has a faulty CDROM drive
- Subsequent software updates can use rsync
RISng Overview

• We take advantage of unused processor resources in each route collector:
  – Data is now inserted into a local database on each RRC and replicated to central servers

• Software changes
  – Zebra BGPd
  – Insertion process
RISng - Overview

**Diagram**

- **BGPd** -> **MySQL** via `dbinsert`
- **Raw Data** -> **MySQL** via `rsync`
- **MySQL** -> **UI** via replication
- **MySQL** -> **MySQL** via replication

**Paths**

- BGPd to MySQL: `dbinsert`
- Raw Data to MySQL: `rsync`
- MySQL to UI: Replication

**Repositories**

- abcoude
- halfweg
RISng - Changes to Zebra BGPd

• Normally Zebra dumps RIB and Updates at times relative to when the process started
  – RIB dumped at 00:00, 08:00, 16:00 hours
  – Updates dumped at xx:00, xx:15, xx:30, xx:45
• Small change made to Zebra’s bgpd/bgp_dump.c to achieve this
  – First dump starts immediately after start-up as before
  – Subsequent dumps follow the above pattern
RISng - New Database Insertion Process

• Rewritten in C
• Runs locally on each RRC
  – Processing time not influenced by what load other RRC’s data insertion may be generating
  – Local database insertion removes delay caused by once hourly rsync of dump files to central machine
• Database is replicated to central servers
  – Simplifies porting of front-end applications to use new database
New Database Structure

• Arbitrary length restrictions removed
• Many new attributes stored
  – More information allows better diagnosis of routing problems
  – RIS users can now see whether what appeared to be duplicate announcements really are identical or whether some other attribute (MED, Community, etc.) has changed.
  – Prevents time being wasted tracking down the wrong problem.
## New Database - Prefix Table

<table>
<thead>
<tr>
<th></th>
<th>RIS Classic</th>
<th>RISng</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prefix</strong></td>
<td>Prefix string</td>
<td></td>
</tr>
<tr>
<td><strong>Start</strong></td>
<td>First address in prefix range</td>
<td></td>
</tr>
<tr>
<td><strong>End</strong></td>
<td>Last address in prefix range</td>
<td></td>
</tr>
<tr>
<td><strong>First time seen</strong></td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td><strong>Last time seen</strong></td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td><strong>IP version</strong></td>
<td>Never used</td>
<td>New</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed</td>
<td></td>
</tr>
<tr>
<td><strong>RRC</strong></td>
<td>RRC Number</td>
<td></td>
</tr>
<tr>
<td><strong>Origin AS</strong></td>
<td>New</td>
<td></td>
</tr>
</tbody>
</table>
# New Database - AS Path Table

<table>
<thead>
<tr>
<th></th>
<th>RIS Classic</th>
<th>RISng</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AS Path</strong></td>
<td>255 characters</td>
<td>Unlimited</td>
</tr>
<tr>
<td><strong>First time seen</strong></td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td><strong>Last time seen</strong></td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Never used</td>
<td>Removed</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td></td>
<td>New: Length of “raw” BGP AS Path attribute</td>
</tr>
<tr>
<td><strong>RRC</strong></td>
<td></td>
<td>RRC Number</td>
</tr>
</tbody>
</table>
# New Database - Peer Table

<table>
<thead>
<tr>
<th>IP Address</th>
<th>RIS Classic</th>
<th>RISng</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS Number</td>
<td>Peer IP Address</td>
<td>Peer AS Number</td>
</tr>
<tr>
<td>Status</td>
<td>Peer Status (up or down)</td>
<td>New</td>
</tr>
<tr>
<td>IP version</td>
<td></td>
<td>Timestamp</td>
</tr>
<tr>
<td>First time seen</td>
<td></td>
<td>Timestamp</td>
</tr>
<tr>
<td>Last time seen</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### New Database - Attributes Table

#### All New!

<table>
<thead>
<tr>
<th>New Database - Attributes Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Origin Type</strong></td>
</tr>
<tr>
<td><strong>Next Hop</strong></td>
</tr>
<tr>
<td><strong>MED</strong></td>
</tr>
<tr>
<td><strong>Community</strong></td>
</tr>
<tr>
<td><strong>IP Version</strong></td>
</tr>
<tr>
<td><strong>“Rest”</strong></td>
</tr>
<tr>
<td><strong>RRC</strong></td>
</tr>
<tr>
<td><strong>First time seen</strong></td>
</tr>
<tr>
<td><strong>Last time seen</strong></td>
</tr>
<tr>
<td><strong>RISng</strong></td>
</tr>
<tr>
<td>‘IGP’, ‘EGP’, ‘Unknown’</td>
</tr>
<tr>
<td>IP address</td>
</tr>
<tr>
<td>Multi-Exit Discriminator</td>
</tr>
<tr>
<td>Unlimited length string</td>
</tr>
<tr>
<td>4 or 6</td>
</tr>
<tr>
<td>Any other attributes we choose to store</td>
</tr>
<tr>
<td>RRC Number</td>
</tr>
<tr>
<td>Timestamp</td>
</tr>
<tr>
<td>Timestamp</td>
</tr>
</tbody>
</table>
## New Database
### RIB & Updates Tables

<table>
<thead>
<tr>
<th>Field</th>
<th>RIS Classic</th>
<th>RISng</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTC</td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td>Prefix ID</td>
<td>Reference to entry in prefix table</td>
<td>Reference to entry in prefix table</td>
</tr>
<tr>
<td>AS Path ID</td>
<td>Reference to entry in AS path table</td>
<td>Reference to entry in AS path table</td>
</tr>
<tr>
<td>Attribute ID</td>
<td>Reference to entry in Attributes table</td>
<td>Reference to entry in Attributes table</td>
</tr>
<tr>
<td>Type</td>
<td>‘U’ (Advertisement)</td>
<td>‘U’ (Advertisement)</td>
</tr>
<tr>
<td></td>
<td>‘W’ (Withdrawal)</td>
<td>‘W’ (Withdrawal)</td>
</tr>
<tr>
<td></td>
<td>‘S’ (BGP state change)</td>
<td>‘S’ (BGP state change)</td>
</tr>
<tr>
<td>RRC ID</td>
<td>RRC Number</td>
<td></td>
</tr>
<tr>
<td>Origin AS</td>
<td>Origin AS (‘U’ entries only)</td>
<td>IPv4 or IPv6</td>
</tr>
<tr>
<td>IP version</td>
<td>Reference to entry in Peer table</td>
<td>Reference to entry in Peer table</td>
</tr>
<tr>
<td>Peer ID</td>
<td>Not Used</td>
<td></td>
</tr>
<tr>
<td>Pstate</td>
<td>Not Used</td>
<td></td>
</tr>
<tr>
<td>Nstate</td>
<td>Not Used</td>
<td></td>
</tr>
</tbody>
</table>
RISng - Ongoing Work

• Improve handling of RIB dump
  – RISng insertion is currently slower than before…
    … but we don’t use any of the Perl tricks to bypass MySQL which the “Classic” insertion process uses
  – Better indexing of database?
  – Intermediate processing of RIB dump followed by single pass to update MySQL tables?
  – Is MySQL the best way of handling this?

• Store even more attributes in the database

• Store IPv6 entries in the database
Credits

• Dan Ardelean for providing the libbgpdump building block for reading Zebra’s BGP dump files.

• RRC Hosts for hands-on support during the sometimes problematic software upgrade process
Summary

• Scalability and software maintenance issues improved

• More attributes stored in database

• More work still to do…

• Test drive the new database:
  – http://www.ris.ripe.net/risng/
Questions?