Routing Protocol Multi-core Support

Overview

RouterOS v7 is capable of splitting tasks between multiple processes.
There is one "**main**" task, which can start/stop sub-tasks and process data between those sub-tasks. Each sub-task can allocate "private" (only accessible by this particular task) and "shared" memory (accessible by all route tasks).

List of tasks that can be split:

- Handling of "print" command;
- Entire OSPF protocol handling;
- Entire RIP protocol handling;
- Static configuration handling;
- Routing Policy configuration;
- BGP connections and configuration handling;
- BGP receive (one task per peer or grouped by specific parameters);
- BGP send (one task per peer or grouped by specific parameters);
- FIB update.

BGP Sub-Tasks

BGP receive and send can be split into sub-tasks by specific parameters, for example, it is possible to run input per each peer or group all peer inputs and run them in the main process. This split by sub-tasks is controlled with **`input.affinity`** and **`output.affinity`** parameter configuration in **`/routing/bgp/template`**.It is possible to boost performance by playing with affinity values on devices with fewer cores since sharing data between tasks is a bit slower than processing the same data within one task. For example, on single-core or two-core devices running input and output in the main or instance process will boost performance.

> BGP can have up to 100 unique processes.

All currently used tasks and their allocated private/shared memory can be monitored using the command:

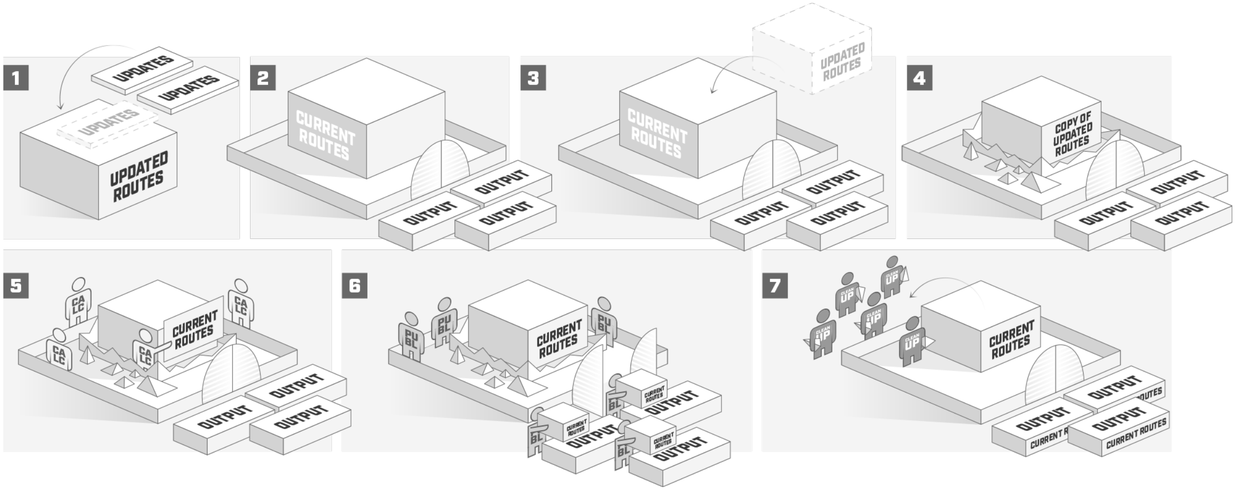
```
/routing/stats/process/print
```

Sample Output:

```
[admin@BGP_MUM] /routing/stats/process> print interval=1
Columns: TASKS, PRIVATE-MEM-BLOCKS, SHARED-MEM-BLOCKS, PSS, RSS, VMS, RETIRED, ID, PID, RPID, PROCESS-TIME,
KERNEL-TIME, CUR-BUSY, MAX-BUSY, CUR-CALC, MAX-CALC
   #  TASKS                        PRIVATE-M  SHARED-M  PSS        RSS      VMS      R  ID       PID  R
PROCESS-  KERNEL-TI  CUR-  MAX-BUSY  CUR-  MAX-CALC
   0  routing tables               11.8MiB    20.0MiB   19.8MiB    42.2MiB  51.4MiB  7  main     195  0
15s470ms  2s50ms     20ms  1s460ms   20ms  35s120ms

rib

      connected
networks

   1  fib                          2816.0KiB  0         8.1MiB     27.4MiB  51.4MiB     fib      255  1
5s730ms   7m4s790ms        23s350ms        23s350ms
   2  ospf                         512.0KiB   0         3151.0KiB  14.6MiB  51.4MiB     ospf     260  1
20ms      100ms            20ms            20ms
      connected
networks

   3  fantasy                      256.0KiB   0         1898.0KiB  5.8MiB   51.4MiB     fantasy  261  1
40ms      60ms             20ms            20ms
   4  configuration and reporting  4096.0KiB  512.0KiB  9.2MiB     28.4MiB  51.4MiB     static   262  1
3s210ms   40ms             220ms           220ms
   5  rip                          512.0KiB   0         3151.0KiB  14.6MiB  51.4MiB     rip      259  1
50ms      90ms             20ms            20ms
      connected
networks

   6  routing policy configuration  768.0KiB   768.0KiB  2250.0KiB  6.2MiB   51.4MiB     policy   256  1
70ms      50ms             20ms            20ms
   7  BGP service                  768.0KiB   0         3359.0KiB  14.9MiB  51.4MiB     bgp      257  1
4s260ms   8s50ms           30ms            30ms
      connected
networks

   8  BFD service                  512.0KiB   0         3151.0KiB  14.6MiB  51.4MiB     12       258  1
80ms      40ms             20ms            20ms
      connected
networks

   9  BGP Input 10.155.101.232     8.2MiB     6.8MiB    17.0MiB    39.1MiB  51.4MiB     20       270  1
24s880ms  3s60ms           18s550ms        18s550ms
      BGP Output
10.155.101.232

  10  Global memory                           256.0KiB                                global     0  0
```

Routing Table Update Mechanism

Illustration below tries to explain in more user friendly form on how routing table update mechanism is working.



Routing protocols continuously loop through following procedures:

- "**main**" process waits for updates from other sub tasks (1);
- "**main**" starts to calculate new routes (2..4) if:
  - update from sub task is received;
  - protocol has not published all routes;
  - configuration has changed or link state has changed.
- during new route calculation (5) following event occur:
  - all received updates are applied to the route;
  - gateway reachability is being determined;
  - recursive route is being resolved;
- "**publish**" event is called where "**current**" routes are being published. During this phase, "**current**" routes will not change, but protocols can still receive and send updates (6).
- Do cleanup and free unused memory (7). In this step everything that is no longer used in new "**current**" table is removed (routes, attributes, etc.).

Consider "**updated**" and "**current**" as two copies of routing table, where "**current**" table (2) is the one used at the moment and "**updated**" (1) is table of candidate routes to be published in the next publish event (3 and 4). This method prevents protocols to fill memory with buffered updates while "**main**" process is doing "**publish**", instead protocols sends the newest update directly to "main" process which then copies new update in "**updated**" table. A bit more complicated is OSPF, it internally has similar process to select current OSPF routes which then are sent to the "**main**" for further processing.