BIND 9 Memory Management

(You can't have too much memory, or can you?)

Carsten Strotmann and the ISC Team

Welcome

Welcome to our Webinar on BIND 9 Memory Management

(many thanks to Cathy Almond, Ondřej Surý and Petr Špaček for their insight on the topic)

In this Webinar

- Memory in BIND 9.11, BIND 9.16 and BIND 9.18
- Configuration that influences BIND 9 memory usage
- Memory Management in Unix/Linux
- How to measure memory consumption of BIND 9
- How to restrict memory available to a process in Linux

All content © 2022 Internet Systems Consortium, Inc.

Memory usage across recent BIND 9 versions

BIND 9.11 vs 9.16 vs 9.18

- BIND 9 is evolving - after 20 years, larger parts of the code base are being updated or replaced
  - The operating system and hardware environment BIND 9 is working in has changed over the last 20 years (Multi-Core hardware, NUMA machines, new CPU architectures, standardized operating system APIs)
  - Custom, platform specific code is being replaced by functions provided by standard libraries

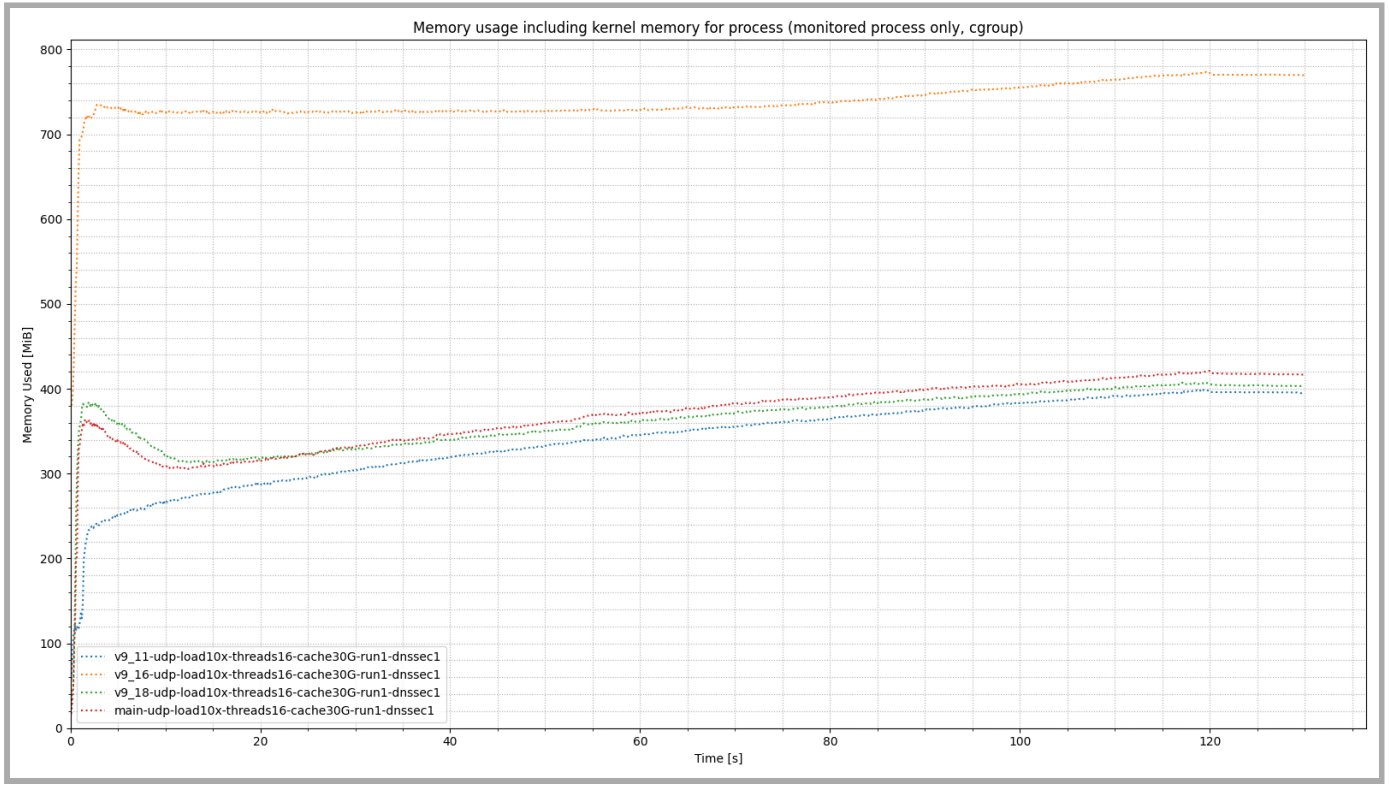
BIND 9.11 vs 9.16 vs 9.18

- In BIND 9.16, parts of the network code has been replaced by a more modern version
  - To better perform on modern systems
  - Faster TCP connections
  - Required for DNS-over-TLS and DNS-over-HTTPS
- In BIND 9.16, the functions processing incoming requests are running the new code, the functions sending requests still use old code
  - This could result in higher memory usage compared to previous BIND 9 versions

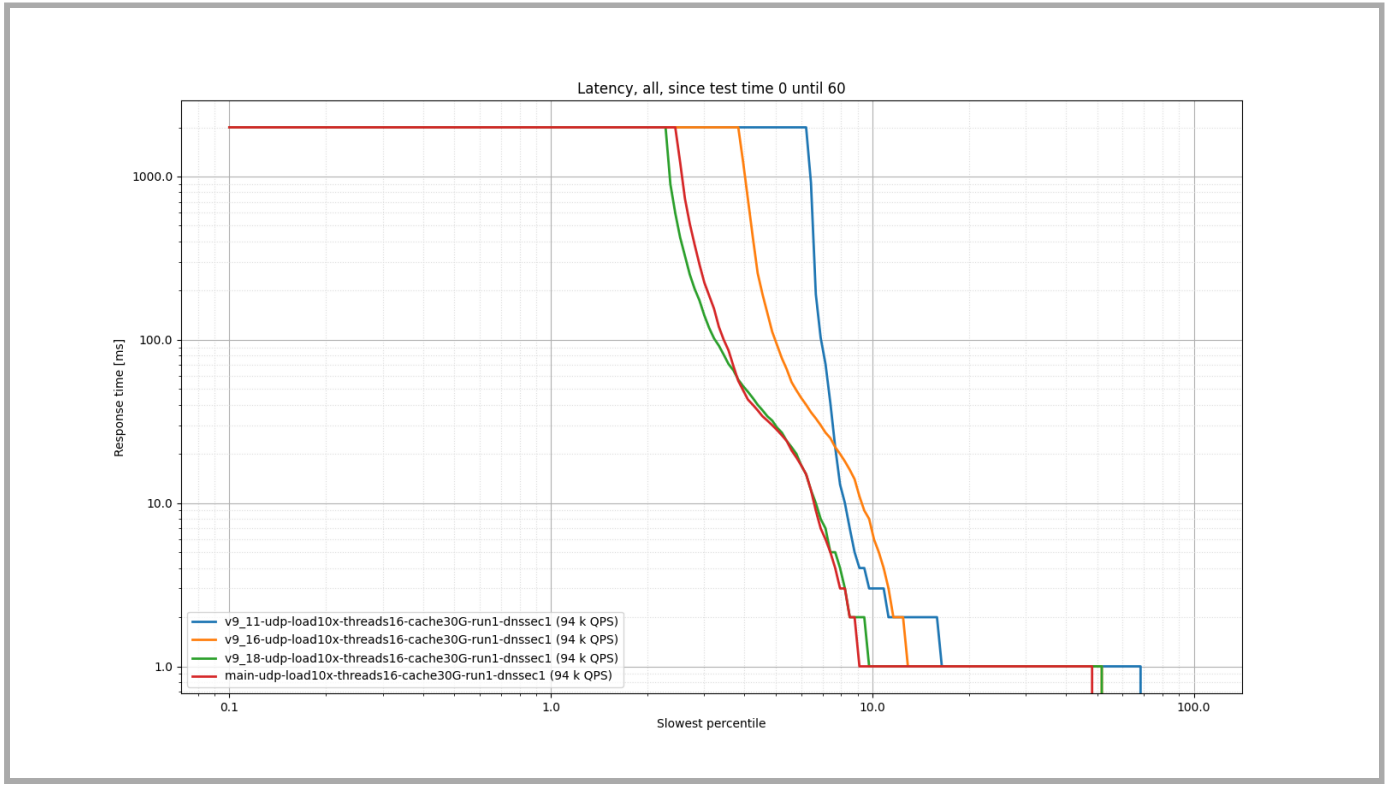
BIND 9.11 vs 9.16 vs 9.18

- In BIND 9.18, the old network code has been removed (incoming and outgoing requests are done from the new code)
  - Memory usage is back to the pre-9.16 level
  - Performance has increased

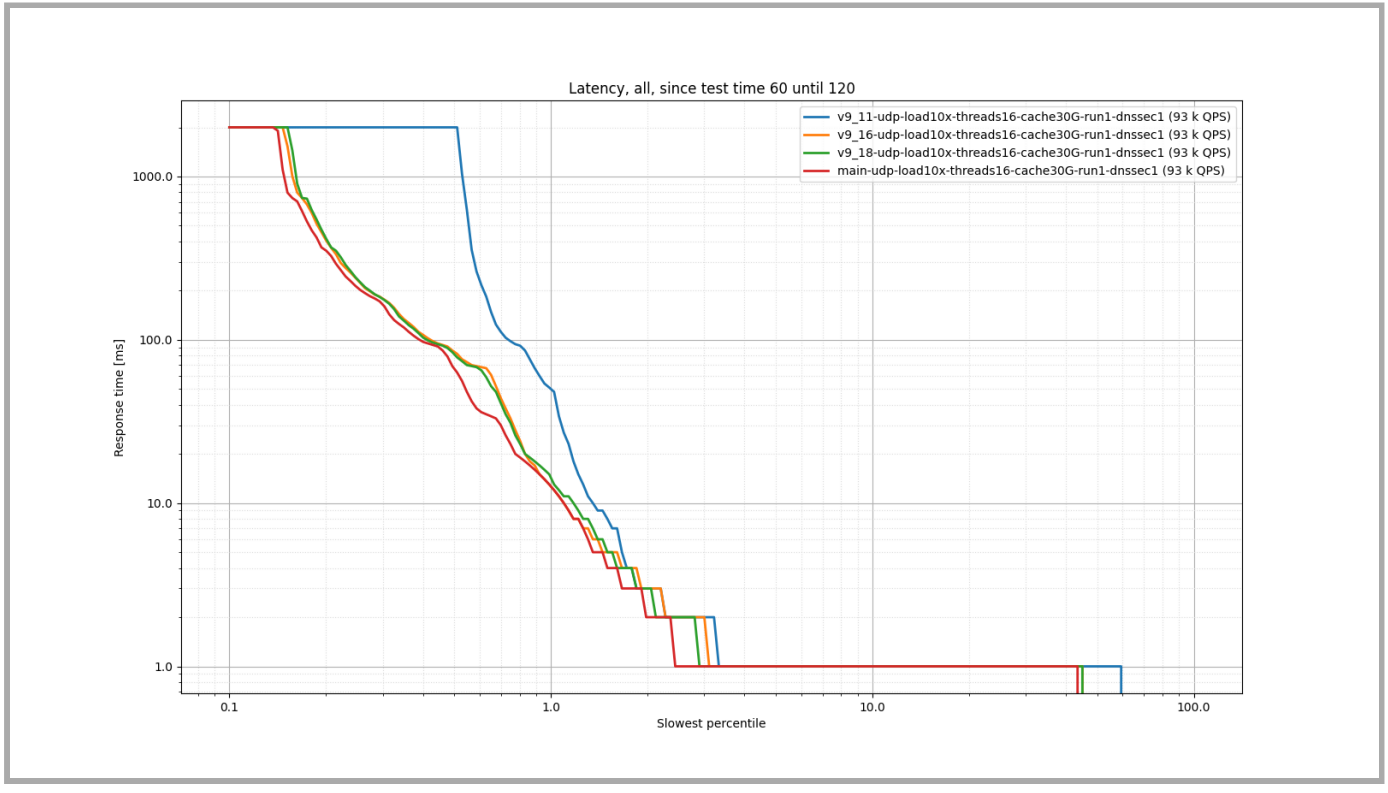
Memory usage of BIND 9 versions



Query latency of BIND 9 versions (Cold cache)



Query latency of BIND 9 versions (Warm cache)



Unix/Linux Memory Management

Virtual Memory

- Memory on Unix/Linux systems can come from different sources
  - Real memory (RAM)
  - Virtual memory (memory backed by a storage device)

Anonymous vs. named Memory

- In a Unix/Linux system, memory can be either named or anonymous
  - Content of *named* memory is coming from files on a storage device (hard disk)
    - Application binary
    - Dynamic link libraries (`.so` files)
    - Memory mapped files
  - Anonymous memory is not backed by a file (the programs *heap* memory)

Resident vs. virtual memory

- Memory requested from an application is added to the pool of *committed* memory
  - But the memory is not *allocated* immediately
  - The operating system allocates the memory on first use (page fault on access)
- The memory requested by an application (virtual memory size = VSZ) is not the real memory used by the process
- The *resident memory size* (RSS) is the amount of memory allocated **and** in RAM (process code, libraries, heap and stack)

Memory Overcommit

- Linux (and other Unix systems) will overcommit memory
  - The OS allows more memory being requested by applications than is available in the system (physical memory plus swap memory)
  - This is possible because application usually don't use all the memory they request

Memory Overcommit and the Linux OOM-Killer

- Once the real memory becomes tight, the OOM (out-of-memory) killer will identify an application that is using much memory and has not been active lately - and will kill that process
  - There is an Systemd OOM killer and an Linux-Kernel OOM killer
  - On a dedicated DNS server, it is best to disable the systemd OOM killer and also disable memory overcommit

```
echo 2 > /proc/sys/vm/overcommit_memory
```

Memory Overcommit and the Linux OOM-Killer

- When disabling overcommit, the machine should have plenty physical RAM and a paging device (swap)

- Linux memory overcommit https://www.marcusfolkesson.se/blog/linux-memory-overcommit/
- Overcommit Accounting https://www.kernel.org/doc/html/v5.1/vm/overcommit-accounting.html
- Disabled systemd-oomd https://utcc.utoronto.ca/~cks/space/blog/linux/SystemdOomdNowDisabled

Paging and Swap-Space

- The operating system can decide to write memory pages to storage to reclaim physical memory
  - Paging (swapping) can happen even when free memory is still available
  - In Linux, the *swappiness* kernel parameter controls how eager the operating system will page out memory to storage (higher value = memory is paged early)

```
# sysctl vm.swappiness
vm.swappiness = 60
```

- Background: In defence of swap: common misconceptions https://chrisdown.name/2018/01/02/in-defence-of-swap.html

Paging and swap space

- When possible, the BIND 9 data (zone files, DNS cache) should not be paged out
  - Reduce the *swappiness* parameter and restrict paging via Systemd CGroup settings (see later chapter)
- The use of *zram* (https://en.wikipedia.org/wiki/Zram) can be a good alternative to disk based swap space

Memory Allocators

- Applications request memory from (and return memory to) the operating system via standard library functions (`malloc` and `free`)
  - The default memory allocators need to be generic
    - Work reasonable well for applications requesting (few) large blocks of memory
    - Work reasonable well for applications requesting (many) small blocks of memory
    - Work on single core machines
    - Work on multi core / multi cpu machines

Memory Allocators

- Older BIND 9 versions implemented their own memory management, seldom or never giving the small memory allocations back
  - This has been common in memory allocators 20 years ago

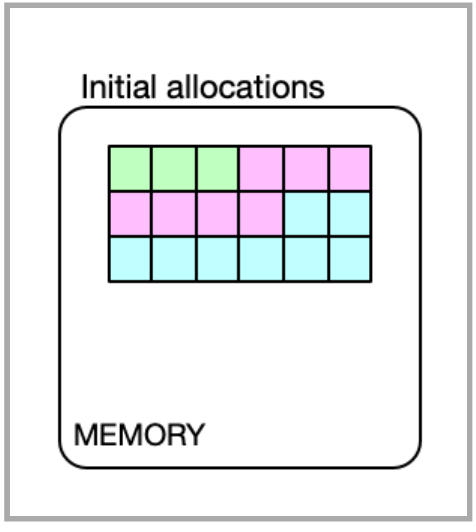
> *For simple allocators, a free(3) does not return memory to the operating system; rather, memory is kept to serve future allocations. This means that the process resident memory can only grow, which is normal.*

Brandon Gregg - Systems Performance 2nd Ed. Chapter 7.3.3

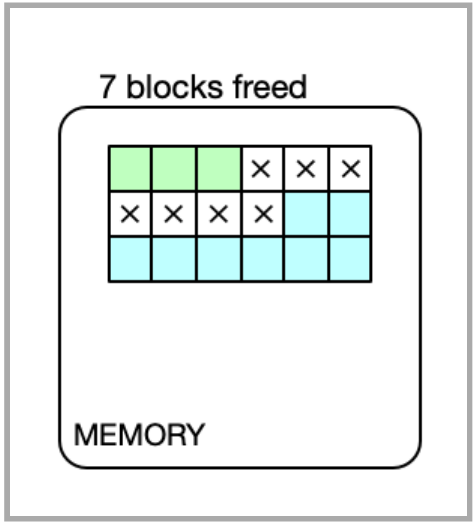
Memory fragmentation

- Similar to file-systems, memory can fragment
  - Blocks of memory are allocated and freed
  - Free blocks might be too small for new allocations, creating unused holes in the memory space
  - The application consumes more memory than it is using (worst case +50%)
    - Example: if you configure the BIND 9 DNS cache in a DNS resolver to be 2 GB, BIND 9 might need up to 50% more memory to keep 2 GB DNS cache in memory

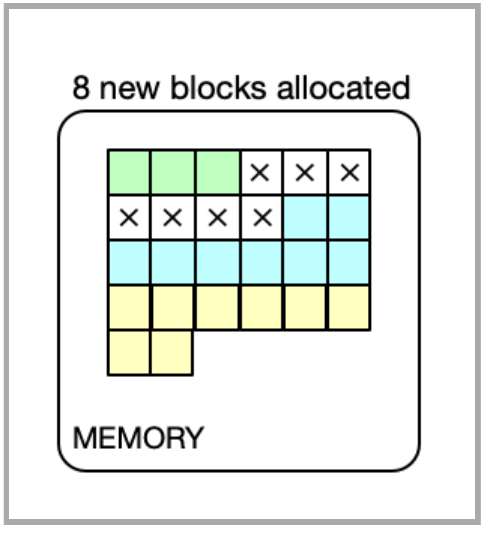
Memory fragmentation



Memory fragmentation



Memory fragmentation



Tuning the memory allocator

- There are many ways to tune the memory allocator in Unix/Linux systems
- Tuning results greatly differ between types of physical hardware and operating system versions
  - Don't tweak the memory allocator without measurements

Don't speculate - measure!

Glibc malloc (Linux)

- Unless otherwise configured, BIND 9.18 will use the memory allocator in the GNU libc library
  - The GLIBC allocator works well for many application workloads
  - It is not optimized for a high number small allocations (such as for a BIND 9 resolver cache)

- GLIBC Memory Allocation Tunables https://www.gnu.org/software/libc/manual/html_node/Memory-Allocation-Tunables.html
- Tuning glibc Memory Behavior https://devcenter.heroku.com/articles/tuning-glibc-memory-behavior

jemalloc (Linux/FreeBSD and others)

- *jemalloc* is an alternative memory allocator originally created in the FreeBSD project: https://jemalloc.net/
- Starting from BIND 9.18, BIND 9 will compile with the *jemalloc* allocator when available in the system
  - *jemalloc* shows better memory usage and better performance for BIND 9 compared to the standard GLibc memory allocator
  - *jemalloc* can be used with older BIND 9 versions as well

How to use alternative allocators with BIND 9

- BIND 9.18 available in *bleeding-edge* Linux distributions (Fedora 37, Debian unstable, Arch Linux) installs *jemalloc* as a dependency

Compiling BIND 9 with jemalloc

- When compiling BIND 9.18+, the `configure` script will detect *jemalloc* and will compile BIND 9 with *jemalloc* support
  - Compiling BIND 9 with *jemalloc* support gives the best support this memory allocator
  - Install the *jemalloc* header files (package `libjemalloc-dev` or `jemalloc-devel`) before compiling BIND 9

```
cd bind-9.18.9
./configure --with-jemalloc=detect ...
```

ISC BIND 9 packages

- ISC provides the latest BIND 9 versions as binary installer packages for popular Linux distributions (CentOS, Ubuntu, Fedora, Debian)
- These packages have more recent versions of BIND 9, and come with support for *jemalloc* compiled
- ISC BIND 9 packages: https://kb.isc.org/docs/isc-packages-for-bind-9

Example: Installing BIND 9 with jemalloc on CentOS 7

```
yum install epel-release
yum install dnf
dnf install 'dnf-command(copr)'
dnf copr enable isc/bind
dnf install isc-bind
```

Preloading jemalloc

- Even older versions of BIND 9, or BIND 9.18 not compiled with *jemalloc* support, can make use of the *jemalloc* memory allocator through *preloading*
- *preloading* will load a library into memory before the application is loaded
  - The functions in that *preloaded* library will overlay functions of the same name in the original binary (in the case of *jemalloc*, the `malloc` and `free` functions)
- Preloading is done by specifying the libraries to be preloaded in the `LD_PRELOAD` environment variable

Preloading jemalloc on Debian/Ubuntu

- On Debian/Ubuntu, the `LD_PRELOAD` variable is set in the file `/etc/default/named`

```
#
# run resolvconf?
RESOLVCONF=no

# startup options for the server
OPTIONS="-u bind"

LD_PRELOAD="/usr/lib/x86_64-linux-gnu/libjemalloc.so.2"
```

Preloading jemalloc on RHEL/CentOS/Fedora

- On Red Hat compatible systems, the `LD_PRELOAD` variable is set in the file `/etc/sysconfig/named`

```
# BIND named process options
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
#
# OPTIONS="whatever"      --  These additional options will be passed to named
#                             at startup. Don't add -t here, enable proper
#                             -chroot.service unit file.
#
[...]
LD_PRELOAD="/usr/lib64/libjemalloc.so.2"
```

How to verify jemalloc support in BIND 9 (Linux)

- When *jemalloc* is available on the system, and support for *jemalloc* is either compiled into BIND 9 or *jemalloc* is preloaded, the `libjemalloc.so.2` library file is mapped into the BIND 9 `named` process space
- The memory mappings of a process can be inspected in the `/proc` pseudo file-system

```
# cat /proc/$(pgrep named)/maps | grep alloc
7faf23e00000-7faf23e08000 r--p 00000000 fd:00 26663227                   /usr/lib64/libjemalloc.so.2
7faf23e08000-7faf23e8f000 r-xp 00008000 fd:00 26663227                   /usr/lib64/libjemalloc.so.2
7faf23e8f000-7faf23e9e000 r--p 0008f000 fd:00 26663227                   /usr/lib64/libjemalloc.so.2
7faf23e9e000-7faf23e9f000 ---p 0009e000 fd:00 26663227                   /usr/lib64/libjemalloc.so.2
7faf23e9f000-7faf23ea5000 r--p 0009e000 fd:00 26663227                   /usr/lib64/libjemalloc.so.2
7faf23ea5000-7faf23ea6000 rw-p 000a4000 fd:00 26663227                   /usr/lib64/libjemalloc.so.2
```

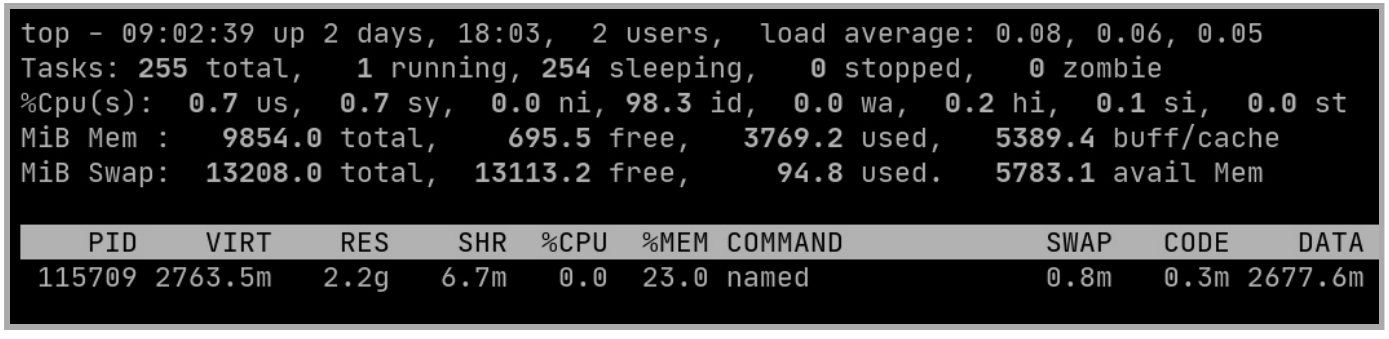
How to measure memory consumption

'top' and similar tools

- The Linux tool `top` is often used to monitor the memory usage of processes
  - However the numbers shown by `top` are sometimes misleading, as it does not show the different types of memory by default
- The `atop` command offers more memory management details (growth or shrinkage of virtual memory) and has the ability to log the data over time (start `atop` as a background daemon)

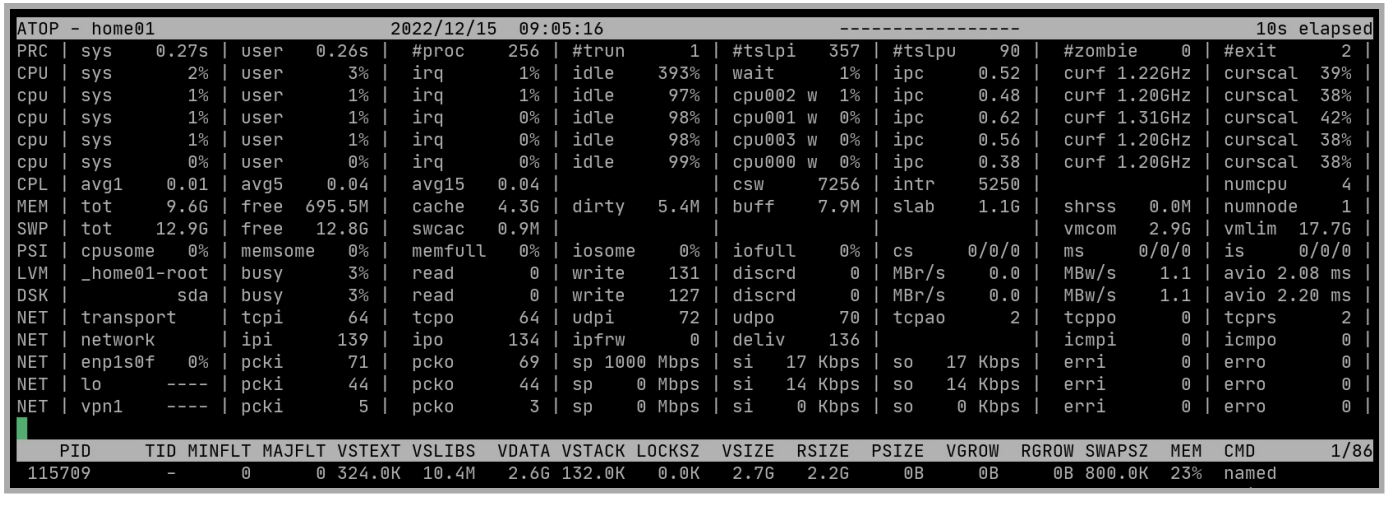
Linux top memory display

- `top` configured to show extra memory fields (Virtual, Resident, Shared, Swap, Code and Data)



Linux atop memory display

- `atop` configured to show key memory indicators



Memory display with pmap

- The command `pmap` can be used to display a detailed view of all memory allocations of a process

```
# pmap -p $(pgrep named) | head
147848:   /usr/sbin/named -u named -c /etc/named.conf
000055b0ac704000     88K r---- /usr/sbin/named
000055b0ac71a000    324K r-x-- /usr/sbin/named
000055b0ac76b000    136K r---- /usr/sbin/named
000055b0ac78d000     12K r---- /usr/sbin/named
000055b0ac790000     12K rw--- /usr/sbin/named
000055b0ac793000     28K rw---   [ anon ]
[...]
00007f6a12674000      8K r---- /usr/lib64/ld-linux-x86-64.so.2
00007f6a12676000      8K rw--- /usr/lib64/ld-linux-x86-64.so.2
00007ffca0288000    132K rw---   [ stack ]
00007ffca02f5000     16K r----   [ anon ]
00007ffca02f9000      8K r-x--   [ anon ]
ffffffffff600000      4K --x--   [ anon ]
 total          2850848K
```

Memory display is ps_mem

- `ps_mem` is a python script that will aggregate the memory allocation information for a process

```
# pip install ps_mem
# ps_mem -S -p $(pgrep named)
 Private  +   Shared  =  RAM used   Swap used   Program

  2.2 GiB + 175.5 KiB =   2.2 GiB     0.0 KiB   named
-------------------------------------------------
                          2.2 GiB     0.0 KiB
=================================================
```

- Source: https://github.com/pixelb/ps_mem

Using the BIND 9 statistics channel

- The BIND 9 statistics channel contains detailed information about the BIND 9 memory allocations
  - This is the *inside* view from within BIND 9
  - Documentation: https://bind9.readthedocs.io/en/v9_18_9/reference.html#statistics-channels-block-definition-and-usage

Enabling the statistics port on SELinux machines

- On Linux systems with SELinux enabled (Red Hat EL, Fedora, CentOS …), the SELinux policy needs to allow BIND 9 to offer the statistics channel on a `http_port`:

```
# setsebool named_tcp_bind_http_port=on
# semanage port -l | grep "^http_port"
http_port_t                    tcp      8066, 80, 81, 443, 488, 8008, 8009, 8443, 9000
```

BIND 9 statistics channel

- Inside the BIND 9 configuration file `named.conf`, configure the statistics channel

```
statistics-channels {
        inet 127.0.0.1 port 8008 allow { localhost; };
};
```

Test the statistics channel

```
# curl -j http://127.0.0.1:8008/json | jq ".memory" | less
{
  "TotalUse": 2692737846,
  "InUse": 2252771540,
  "Malloced": 2252880844,
  "ContextSize": 109304,
  "Lost": 0,
  "contexts": [
    {
      "id": "0x7f44eaeab000",
      "name": "main",
      "references": 325,
      "total": 8305658,
      "inuse": 7207085,
      "maxinuse": 0,
      "malloced": 7215493,
      "maxmalloced": 7289413,
      "pools": 0,
      "hiwater": 0,
      "lowater": 0
[...]
```

Aggregating the memory statistics with memory-json.py

- `memory-json.py` is a Python script provided by ISC that will read the BIND 9 JSON statistics and will print out the memory allocations for the different parts of BIND 9:

```
# curl -j http://127.0.0.1:8008/json > bind9-stats.json
# ./memory-json.py bind9-stats.json
main: 6.9MiB 6.9MiB
zonemgr-pool: 2.1GiB 2.1GiB
clientmgr: 512.0B 33.3KiB
cache: 56.4KiB 72.9KiB
cache_heap: 74.1KiB 90.5KiB
ADB: 320.9KiB 337.3KiB
SUMMARY
INUSE: 2.1GiB == 2.1GiB
MALLOCED: 2.1GiB == 2.1GiB
```

https://gitlab.isc.org/isc-projects/bind9/-/wikis/uploads/e9398e64964bbd68e7715c594dadbd3e/memory-json.py

Measuring the working set

- The *working set* is the total (physical) memory a process requires to work
  - How To Measure the Working Set Size on Linux https://www.brendangregg.com/blog/2018-01-17/measure-working-set-size.html
  - Working Set Size Estimation https://www.brendangregg.com/wss.html

Measuring the working set

- Brendan Gregg has created tools to estimate the working set size of running processes: https://github.com/brendangregg/wss
- The `wss.pl` tool measures the memory pages that are *in use* during the measurement time (in this example, 11MB have been used from 2.2GB during the 10 second measurement time)

```
# ./wss.pl $(pgrep named) 10
Watching PID 150109 page references during 10 seconds...
Est(s)     RSS(MB)    PSS(MB)    Ref(MB)
10.039     2277.16    2273.77      11.86
```


All content © 2022 Internet Systems Consortium, Inc.

Configuration settings and Memory consumption

Resolver: Cache size

- Cache-size: The BIND 9 configuration option `max-cache-size` takes a percentage (default 90%) or a fixed amount of memory to be used for the record cache (positive **and** negative caching)
- Never set this option to 100%, as BIND 9 needs memory besides the cache (and other applications on the system will need memory as well)

Resolver: Cache size

- In an ideal world, the cache should be large enough to store all DNS records until their TTL expires
  - Cleanup of old entries base on TTL is work than finding and freeing still valid entries
  - Depending on the hardware architecture, a (too) large cache can result in an real-world performance degradation (measure!)
  - 2GB - 5GB cache size are a good starting point on an 64-bit architecture

Resolver: TTL cache times

- The upper bound for the TTL of cache entries can be configured with the `max-cache-ttl` (positive results) and `max-ncache-ttl` (negative query results) options
  - Adjusting the max. negative caching time can lead to less cache usage when there are many negative responses (example: `max-ncache-ttl 3600`)

Resolver: stale caching

- BIND 9 can serve *stale* (expired) records from the cache in case all upstream authoritative servers are not responding
- Current BIND 9 versions (9.18 and below) do also cache stale negative responses (NXDOMAIN/NODATA) for `max-stale-ttl` time.
  - This will change in a future version of BIND 9, see https://gitlab.isc.org/isc-projects/bind9/-/issues/3386

Resolver: Aggressive use of DNSSEC-Validated cache

- Starting with BIND 9.18.0, a DNSSEC enabled BIND 9 resolver will (by default) synthesize negative responses from an validated NSEC or NSEC3 chain in the cache
  - This prevents an extra round-trip to the authoritative server (performance win and DDoS protection)
  - It also prevents an extra NXDOMAIN entry in the resolver cache

Documentation: https://bind9.readthedocs.io/en/v9_18_9/reference.html#namedconf-statement-synth-from-dnssec

Resolver: Split-Horizon

- When using multiple *views* within a BIND 9 resolver, each view will have it's own cache (adjust `max-cache-size` accordingly, as it configures the size of each cache in each view)
- BIND 9 views can share the resolver cache, with the `attach-cache` option a view can be configured to re-use the cache of a different view

Resolver: Local Root Zone

- Serving a copy of the Internet Root-Zone from the resolver (see RFC 8806 "Running a Root Server Local to a Resolver" https://www.rfc-editor.org/rfc/rfc8806)
- The DNS resolver will be able to respond authoritatively to single label names
  - This will speed up the name resolution process and will prevent negative caching of answers for non-existing single label names
- The configuration `synth-from-dnssec` has a similar effect, as the root zone is DNSSEC signed, without the need to transfer the root zone content to the resolver

Resolver: Be authoritative for local/private zones

- If the network uses a private DNS name-space (not delegated in the Internet DNS), the DNS resolver should be authoritative for these zones
  - This prevents leakage of internal information
  - It speeds up resolution for local/private names
  - Prevents caching of negative responses from the Internet for local/private names
  - Don't forget private IP-Address ranges (reverse resolution)

Resolver: response policy zones (RPZ)

- Response policy zones can be large - monitor the memory footprint of these zones (esp. when subscribing to external RPZ sources)
- When re-writing responses, the option `qname-wait-recurse no` will prevent normal recursion (and caching) of the requested name(s)

Authoritative Server: Share zones across views

- In case the same zone must be visible in multiple views, this zone content can be shared across views with the `in-view` statement.
  - See https://bind9.readthedocs.io/en/v9_18_9/reference.html#multiple-views

```
view internal {
    match-clients { 10/8; };
    zone example.com {  type primary; file "example.com-zone"; };
};

view external {
    match-clients { any; };
    zone example.com { in-view internal; };
};
```

Restricting memory

Using Systemd and CGroups

- On Linux, Systemd can use the Linux CGroups feature to restrict the resource usage of a process
- These memory usage restrictions are configured in the systemd unit file (`/etc/systemd/system/named.service`)

```
[...]
MemoryAccounting=True
MemoryHigh=3G
MemoryMax=4G
MemorySwapMax=0M
MemoryDenyWriteExecute=True
[...]
```

Using Systemd and CGroups

- `MemoryAccounting` enables the memory accounting feature for this process group
- `MemoryHigh`: Specify the throttling limit on memory usage of the processes in this unit. Memory usage may go above the limit if unavoidable, but the processes are heavily slowed down and memory is taken away aggressively in such cases.
- `MemoryMax`: This is the hard limit, the process in this unit cannot allocate more memory. Calls to `malloc` will fail.
- `MemorySwapMax`: Specify the absolute limit on swap usage of the executed processes in this unit. For BIND 9, we don't want any part of it to be paged out.
- Documentation: https://www.freedesktop.org/software/systemd/man/systemd.resource-control.html

Using Systemd and CGroups

- Systemd memory restrictions will be shown in the `systemctl status` output

```
# systemctl status named
● named.service - Berkeley Internet Name Domain (DNS)
     Loaded: loaded (/etc/systemd/system/named.service; enabled; preset: disabled)
     Active: active (running) since Thu 2022-12-15 11:34:24 CET; 1h 13min ago
    Process: 150105 ExecStartPre=/bin/bash -c if [ ! "$DISABLE_ZONE_CHECKING" == "yes" ]; then /usr/bin
    Process: 150107 ExecStart=/usr/sbin/named -u named -c ${NAMEDCONF} $OPTIONS (code=exited, status=0/
   Main PID: 150109 (named)
      Tasks: 10 (limit: 20)
     Memory: 2.2G (high: 3.0G max: 4.0G swap max: 0B available: 796.3M)
        CPU: 40.256s
     CGroup: /system.slice/named.service
             └─150109 /usr/sbin/named -u named -c /etc/named.conf
```

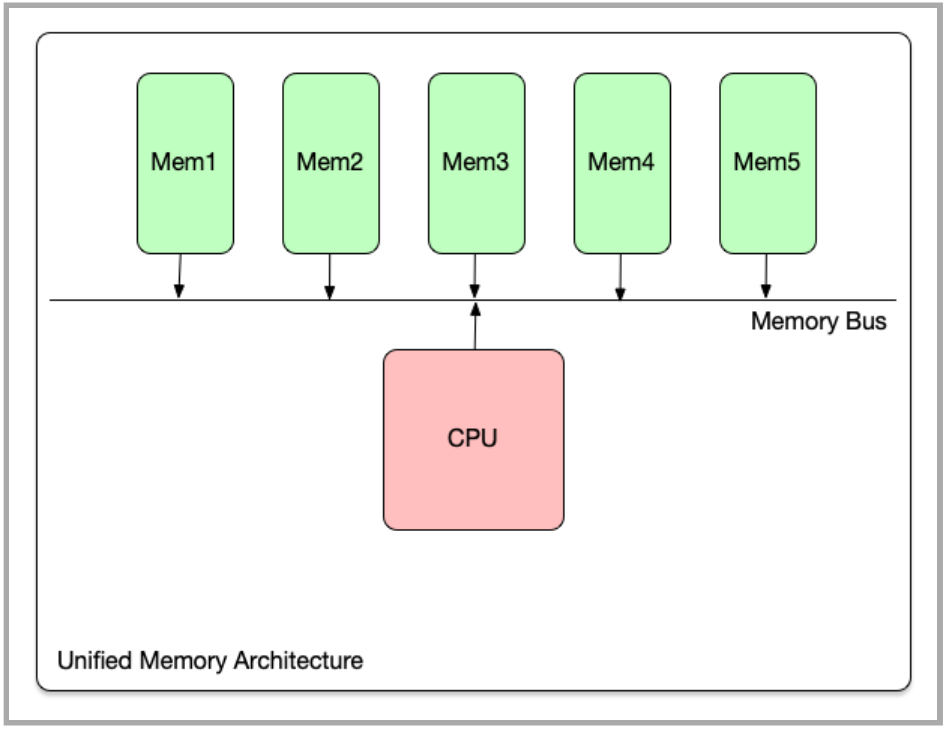

All content © 2022 Internet Systems Consortium, Inc.

BIND 9, Memory and NUMA machines

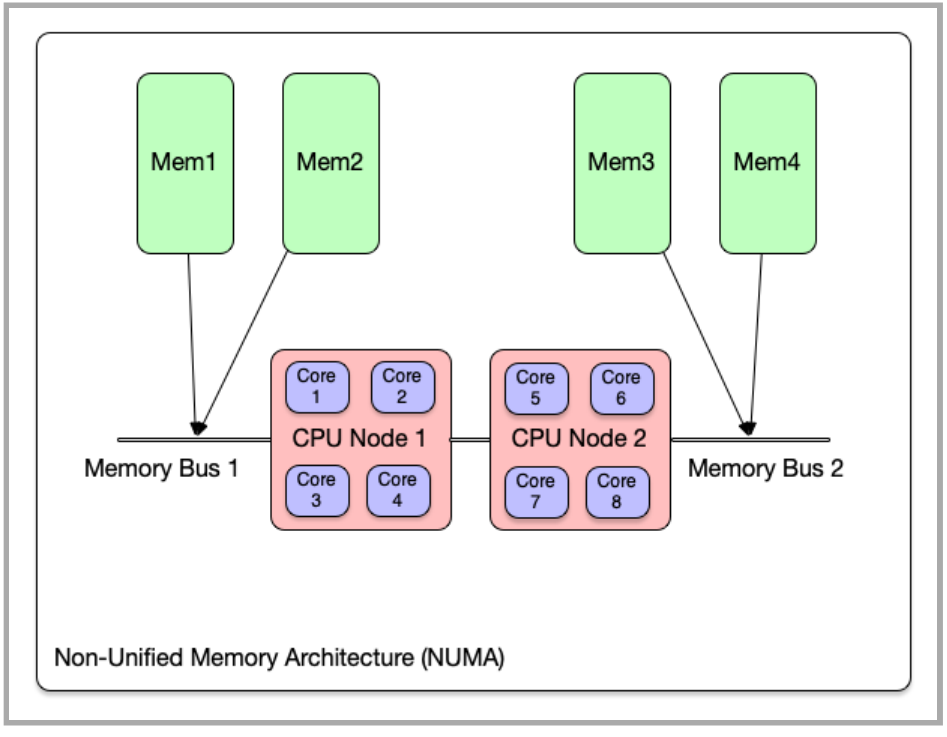
UMA and NUMA machines

- *NUMA* stands for *Non-Unified Memory Access* - it describes a computer system where main memory (RAM) has different access times based on the CPU core
  - AMD Ryzen/Zen is a popular NUMA architecture

UMA and NUMA machines



UMA and NUMA machines



BIND 9 on a NUMA machine running Linux

- To optimize performance, BIND 9 can be restricted to on use specific CPU cores and memory banks
  - This is best done with configuration options in the systemd unit file (s https://www.freedesktop.org/software/systemd/man/systemd.resour control.html)
  - The tool `numastat` will display statistics on the utilization of NUMA memory

```
[...]
CPUAccounting=True
CPUAffinity=0,24,48,72
AllowedCPUs=0,24,48,72
NUMAPolicy = interleave
NUMAMask = 1,2,3-8
AllowedMemoryNodes = 5-8, 12-14
```

BIND 9 on a NUMA machine running Linux

- To fully utilize a NUMA machine with many cores, multiple BIND 9 instances (each on it's own NUMA node) can be faster than one large BIND 9 process spanning multiple NUMA nodes
  - BIND 9 on Linux and FreeBSD can bind to a port already in use by another instance of BIND 9 with the `reuseport true;` option. The Linux kernel will distribute the incoming queries to the different BIND 9 processes https://bind9.readthedocs.io/en/v9_18_8/reference.html#namedconf statement-reuseport

All content © 2022 Internet Systems Consortium, Inc.

Resources

- BIND 9 Memory Explained https://gitlab.isc.org/isc-projects/bind9/-/wikis/BIND-9-Memory-Explained
- Book: Systems Performance: Enterprise and the Cloud, 2nd Edition (2020) https://www.brendangregg.com/systems-performance-2nd-edition-book.html
- Josh Haberman: one malloc to rule them all https://blog.reverberate.org/2009/02/one-malloc-to-rule-them-all.html
- Malloc Replacements https://web.archive.org/web/20110607183215/http://blog.pavlov.net/2007/11/21/malloc-replacements/

All content © 2022 Internet Systems Consortium, Inc.

Resources

- JEMalloc Paper "A Scalable Concurrent malloc(3) Implementation for FreeBSD"
  http://people.freebsd.org/~jasone/jemalloc/bsdcan2006/jemalloc.pdf
- Hoard Allocator: http://hoard.org/
- Brenda Gregg `wss` tool to measure the working set memory size
  https://github.com/brendangregg/wss

All content © 2022 Internet Systems Consortium, Inc.

Resources

- GLibc Malloc Tunable Parameters https://www.gnu.org/software/libc/manual/html_node/Malloc-Tunable-Parameters.html
- Tuning glibc Memory Behavior https://devcenter.heroku.com/articles/tuning-glibc-memory-behavior
- Limitations of Tuning glibc malloc https://lpc.events/event/11/contributions/1006/attachments/857/1626/limitations_malloc.pdf
- Configuring CPU Affinity and NUMA policies using systemd https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_monitoring_and_updating_the_kernel/assembly_configuring-cpu-affinity-and-numa-policies-using-systemd_managing-monitoring-and-updating-the-kernel

All content © 2022 Internet Systems Consortium, Inc.

Resources

- Measuring Efficiency of Aggressive Use of DNSSEC-Validated Cache (RFC 8198) by Petr Špaček https://indico.dns-oarc.net/event/28/contributions/509/attachments/479/786/DNS-OARC-28-presentation-RFC8198.pdf
- Performance effects of DNSSEC validation https://www.isc.org/docs/2022-oarc38-dnssec.pdf
- DNSSEC validation: performance killer? (also includes information of memory overhead) https://www.isc.org/blogs/dnssec-validation-performance-july-2022/

All content © 2022 Internet Systems Consortium, Inc.

Upcoming ISC Webinars

- We will have more webinars in 2023 - please send in your ideas

All content © 2022 Internet Systems Consortium, Inc.

Questions and Answers

All content © 2022 Internet Systems Consortium, Inc.

Happy Holidays and a peaceful new year 2023!

All content © 2022 Internet Systems Consortium, Inc.