Performance optimization for VoIP services

Henning Westerholt

FOSDEM Brussels

February 2023

Agenda

- About
- How to not achieve great performance
- Approach
- Tools
- Contact

About GILAWA

- We offer services for Real-Time Communication platforms
  - Consulting and Management
  - Administration/Developer trainings
  - Development and IT Operations
- Kamailio experience since 2007
- Independent and neutral service provider
  - No own end-user products
  - No vendor contracts
- Our customer are Internet Service Providers and Telephone Provider
- Germany, Europe, North-America, Asia and Middle-East

How to not achieve great performance

- Real-life customer example
  - We need to make a routing decision for an incoming INVITE
  - Kamailio forks a process with “exec” module
  - Exec module starts a Perl script
  - Perl script is interpreted and access over Perl database layer a remote database
  - Perl returns database return value back to Kamailio
  - Kamailio parses return value and forward SIP messages to destination
  - Return values, database query and exit status are logged in Proxy and Perl script
- As soon as you reach a (small) number of concurrent calls, this breaks down

Addressing performance problems

- Of course most performance problems are less obvious then this example
- The first step should be to formulate a goal
  - We want to achieve X concurrent calls per second in a certain scenario
  - We need to support Y REGISTER messages per second per server
  - We like to handle Z connected user-agents over TLS
- Compare against current situation from production load, past incidents or (ideally) performance test results

Common causes of bottle-necks (1/2)

- CPU performance
  - CPU over-commitment on virtual platform, 1 virtual core != 1 physical core
  - Issues with supporting services on production system (cfg mgmt, monitoring..)
  - Not suitable worker configuration for Kamailio (defaults are usually fine)
- Memory related issues
  - Not suitable memory pool configuration for Kamailio (increase defaults)
  - Insufficient memory for database cache pool
  - Insufficient memory for HTTP API service
  - In special use cases Kamailio memory manager might be not optimal (use FM instead of default)

Common causes of bottle-necks (2/2)

- Most problems are usually related to I/O performance
  - Slow DNS lookups (Use the Kamailio DNS cache or dnsmasq)
  - Optimize write operations (Kamailio location cache, asterisk deactivate qualify)
  - Slow local disk performance for logging (restrict logging to a sensible amount)
  - To many database queries to a remote database (use caching or local replication)
  - To many requests to a remote API service (use caching)
  - Extensive CDR writing (move to asynchronous handling)
  - Avoid forking of new processes for error or overload conditions
  - …

Tools

- Performance tests
  - sipp (use dedicated test host)
  - pjsua client and libraries for scripting
  - Specialized performance test frameworks (usually hown-grown or closed source)
  - Custom tools to test database, HTTP API performance
- Performance analysis
  - Common linux system tools (htop, iotop, netstat, iostat, vmstat etc..)
  - Monitoring tools like icinga, grafana etc..
  - Quality related VoIP tools (Homer, voipmonitor)
  - Kamailio benchmark module, custom Kamailio logging with PVs

Thank you

- Thank you – any questions?

- Hope to see you at Kamailio World 2023!
  - Call for papers is now open
  - June 5-7, 2023, Berlin, Germany
  - https://www.kamailioworld.com/

Thank you

Henning Westerholt

GILAWA Ltd

hw@gilawa.com

https://gilawa.com/