Stopping systemd services under memory pressure

Monday, 15 December 2025 - ⏱ 10 min

Do you have your "favourite" server that is responsible for just a tiny number too many things ?

I know I have one, it's my CI runner, but also serves as a platform to experiment with local LLMs, but also is a Nix builder for all other machines I have. Lots of responsiblities ! Luckily, I don't need it to do all of those things at the same time, it just didn't know that previously. I would run a local build, it would get sent to that machine, but that machine would have e.g. an LLM loaded in memory and the build would OOM. I would then need to go there, stop the LLM service and restart the build. What if a computer could do it all by itself ?!

I present to you - a simple service that monitors available memory and based on that starts and stops other services dynamically. The code is in Nix, that helps with keeping the service code and definition in one place, making sure paths make sense etc. However, there is nothing fundamentally Nix specific to the service itself, you can just take the Bash code and use it anywhere.

Speaking of usage (from Nix) you can use it like so; if you want to, say, stop LLM server if available mem is under 8GB and start it again when it's over whatever LLM needs (in my case it's 64GB):

```nix
services.custom.memoryManager = {
  enable = true;
  stopThresholdMB = 8192;
  startThresholdMB = config.services.llm.memoryRequirementMB;
  checkIntervalSeconds = 30;
  managedServices = [ "llama-cpp-fast" ];
};
```

And this gives you

```
Dec 15 12:05:22 memory-manager: Memory: 7543MB available |
Decision: STOP (below 8192MB)
Dec 15 12:05:22 memory-manager: LOW MEMORY: Stopping llama-cpp-
fast
Dec 15 12:05:23 memory-manager: llama-cpp-fast stopped
successfully
Dec 15 12:08:53 memory-manager: Memory: 88902MB available |
Decision: START (above 65536MB)
Dec 15 12:08:53 memory-manager: MEMORY OK: Starting llama-cpp-fast
Dec 15 12:08:54 memory-manager: llama-cpp-fast started
successfully
```

And here is the module definition itself. The code is not the prettiest but it does help a lot already !

```
{
  config,
  lib,
  pkgs,
  ...
}:

with lib;

let
  cfg = config.services.custom.memoryManager;

  monitorScript = pkgs.writeShellScript "memory-manager" ''
    set -euo pipefail

    STOP_THRESHOLD_MB=${toString cfg.stopThresholdMB}
    START_THRESHOLD_MB=${toString cfg.startThresholdMB}
    CHECK_INTERVAL=${toString cfg.checkIntervalSeconds}
    MANAGED_SERVICES="${concatStringsSep " " cfg.managedServices}"
    STATE_DIR="/run/memory-manager"
    DEBUG=${if cfg.debug then "true" else "false"}

    mkdir -p "$STATE_DIR"

    log() {
      echo "$1"
    }
    get_available_memory_mb() {
      ${pkgs.gawk}/bin/awk '/^MemAvailable:/ { printf "%d", $2 /
1024 }' /proc/meminfo
    }

    is_service_stopped_by_us() {
      local service="$1"
      [ -f "$STATE_DIR/$service.stopped" ]
    }

    mark_service_stopped() {
      local service="$1"
      touch "$STATE_DIR/$service.stopped"
    }

    mark_service_started() {
      local service="$1"
      rm -f "$STATE_DIR/$service.stopped"
    }

    stop_service() {
      local service="$1"

      if ${pkgs.systemd}/bin/systemctl is-active --quiet
"$service"; then
        log "LOW MEMORY: Stopping $service"

        ${pkgs.systemd}/bin/systemctl disable --runtime --now
"$service" 2>/dev/null || true

        if ! ${pkgs.systemd}/bin/systemctl is-active --quiet
"$service"; then
          mark_service_stopped "$service"
          log "$service stopped successfully"
        else
          log "WARNING: Failed to stop $service"
        fi
      fi
    }

    start_service() {
      local service="$1"
      if is_service_stopped_by_us "$service"; then
        log "MEMORY OK: Starting $service"
        ${pkgs.systemd}/bin/systemctl enable --runtime "$service"
2>/dev/null || true
        ${pkgs.systemd}/bin/systemctl start "$service" 2>/dev/null
|| true

        if ${pkgs.systemd}/bin/systemctl is-active --quiet
"$service"; then
          mark_service_started "$service"
          log "$service started successfully"
        else
          log "WARNING: Failed to start $service"
        fi
      fi
    }

    log "Memory manager started"
    log "Stop threshold: ''${STOP_THRESHOLD_MB}MB, Start
threshold: ''${START_THRESHOLD_MB}MB"
    log "Managed services: $MANAGED_SERVICES"
    log "Check interval: ''${CHECK_INTERVAL}s, Debug: $DEBUG"

    while true; do
      AVAILABLE_MB=$(get_available_memory_mb)

      if [ "$AVAILABLE_MB" -lt "$STOP_THRESHOLD_MB" ]; then
        DECISION="STOP (below ''${STOP_THRESHOLD_MB}MB)"
        log "Memory: ''${AVAILABLE_MB}MB available | Decision:
$DECISION"
        for service in $MANAGED_SERVICES; do
          stop_service "$service"
        done
      elif [ "$AVAILABLE_MB" -gt "$START_THRESHOLD_MB" ]; then
        DECISION="START (above ''${START_THRESHOLD_MB}MB)"
        NEEDS_START=false
        for service in $MANAGED_SERVICES; do
          if is_service_stopped_by_us "$service"; then
            NEEDS_START=true
            break
          fi
        done
        if [ "$NEEDS_START" = "true" ]; then
          log "Memory: ''${AVAILABLE_MB}MB available | Decision:
```

```
$DECISION"
          for service in $MANAGED_SERVICES; do
            start_service "$service"
          done
        else
          log "Memory: ''${AVAILABLE_MB}MB available | Decision:
OK (services running)"
        fi
      else
        if [ "$DEBUG" = "true" ]; then
          DECISION="WAIT (between ''${STOP_THRESHOLD_MB}MB and
''${START_THRESHOLD_MB}MB)"
          SERVICE_STATUS=""
          for service in $MANAGED_SERVICES; do
            if ${pkgs.systemd}/bin/systemctl is-active --quiet
"$service"; then
              SERVICE_STATUS="$SERVICE_STATUS $service=running"
            elif is_service_stopped_by_us "$service"; then
              SERVICE_STATUS="$SERVICE_STATUS $service=stopped-by-
us"
            else
              SERVICE_STATUS="$SERVICE_STATUS $service=stopped"
            fi
          done
          log "Memory: ''${AVAILABLE_MB}MB available | Decision:
$DECISION | Services:$SERVICE_STATUS"
        fi
      fi

      sleep "$CHECK_INTERVAL"
    done
  '';
in
{
  options.services.custom.memoryManager = {
    enable = mkEnableOption "memory pressure manager that
stops/starts services based on available memory";

    stopThresholdMB = mkOption {
      type = types.int;
      default = 8192;
      description = "Stop managed services when available memory
drops below this threshold (in MB)";
      example = 4096;
    };

    startThresholdMB = mkOption {
      type = types.int;
      default = 16384;
      description = "Restart managed services when available
memory rises above this threshold (in MB). Should be higher than
stopThresholdMB to prevent flapping.";
      example = 32768;
    };

    checkIntervalSeconds = mkOption {
      type = types.int;
      default = 10;
      description = "How often to check memory availability (in
seconds)";
      example = 30;
    };

    managedServices = mkOption {
      type = types.listOf types.str;
      default = [ ];
      description = "List of systemd service names to manage
(without .service suffix)";
      example = [ "llama-cpp-fast" ];
    };

    debug = mkOption {
      type = types.bool;
      default = false;
      description = "Enable debug logging (logs WAIT states on
every check interval)";
    };
  };

  config = mkIf cfg.enable {
    assertions = [
      {
        assertion = cfg.startThresholdMB > cfg.stopThresholdMB;
        message = "services.custom.memoryManager.startThresholdMB
must be greater than stopThresholdMB to prevent flapping";
      }
      {
        assertion = cfg.managedServices != [ ];
    message = "services.custom.memoryManager.managedServices
must not be empty when enabled";
   }
  ];

  systemd.services.memory-manager = {
   description = "Memory Pressure Service Manager";
   wantedBy = [ "multi-user.target" ];
   after = [ "multi-user.target" ];

   serviceConfig = {
    Type = "simple";
    Restart = "always";
    RestartSec = "10s";
    ExecStart = "${monitorScript}";
    RuntimeDirectory = "memory-manager";
    RuntimeDirectoryMode = "0755";
   };
  };
 };
}
```

Cyryl Plotnicki

Monday, 15 December 2025 - □ 10 min

#systemd #linux #nix #nixos

Related systemd content

- Booting Gentoo with LUKS+LVM2+systemd

← Long posts in Mastodon, but now declarative !