CS 311 Project Report

Group Members

Abdullah Bin Zubair	2023037
Abdullah Yasin	2023049
Zain Wajid	2023775

System Resource Monitor Implementation

1. Introduction to the Problem

Modern operating systems require efficient methods to monitor system resources such as CPU usage, memory consumption, process counts, and system load. While Linux provides various tools to access this information, these methods either:

1. Parse text files from /proc filesystem (inefficient for frequent access)
2. Use multiple system calls to gather different pieces of information
3. Lack real-time integration in a single, atomic operation

The need for a unified, efficient system call that can retrieve all critical system resource information in a single kernel-space operation is essential for:

- Performance monitoring applications
- System administration tools
- Resource management daemons
- Embedded systems requiring low-overhead monitoring

Objective

To design and implement a custom Linux kernel system call named sys_resmon that:

- Retrieves comprehensive system resource information in one call
- Minimizes kernel-user space context switches
- Provides real-time, accurate data
- Can be accessed by user-space applications efficiently

Scope

This project involves:

1. Modifying the Linux kernel source code (version 6.1.60)
2. Implementing a new system call sys_resmon (syscall number 451)
3. Compiling and installing the custom kernel
4. Developing a user-space monitoring application
5. Testing and validating the implementation

2. The Unique Function - sys_resmon

System Call Specification

System Call Number: 451
Function Name: `sys_resmon`
Architecture: ARM64 (aarch64)
Prototype:

```
long sys_resmon(struct resource_info __user *info);

struct resource_info {
    unsigned long total_ram;      // Total RAM in KB
    unsigned long free_ram;       // Free RAM in KB
    unsigned long used_ram;       // Used RAM in KB
    unsigned long total_swap;     // Total swap space in KB
    unsigned long free_swap;      // Free swap space in KB
    unsigned long procs;          // Number of active processes
    unsigned long uptime;         // System uptime in seconds
    unsigned long load_1;         // 1-minute load average * 65536
    unsigned long load_5;         // 5-minute load average * 65536
    unsigned long load_15;        // 15-minute load average * 65536
};
```

File Location: `kernel/custom_syscalls/sys_resmon.c`

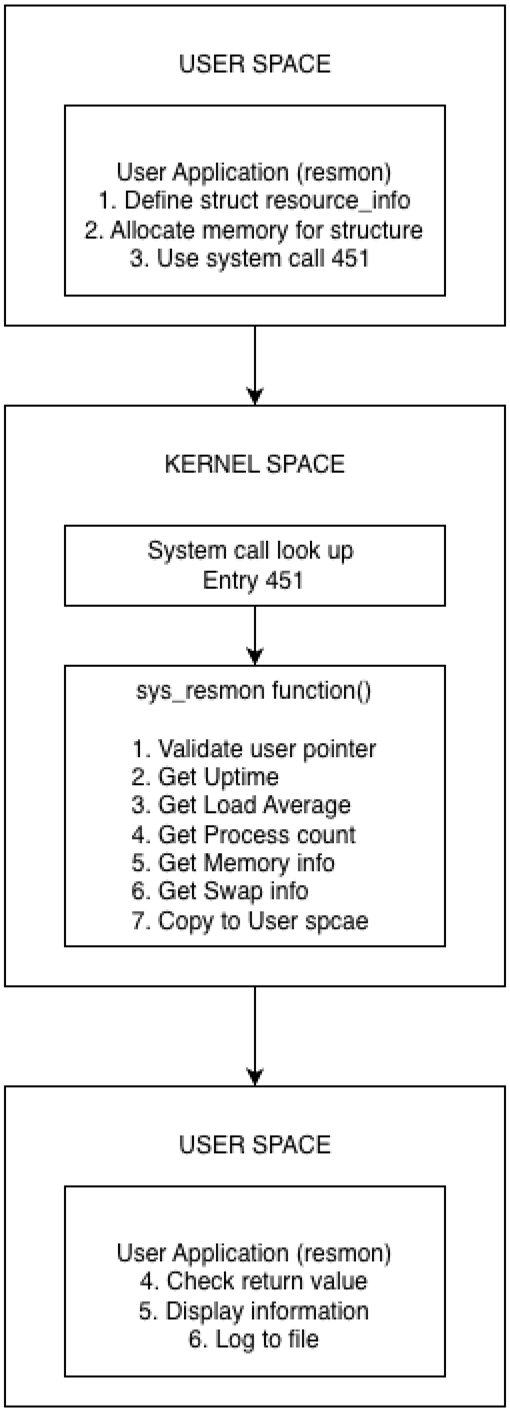
```
#include <linux/kernel.h>
#include <linux/syscalls.h>
#include <linux/uaccess.h>
#include <linux/mm.h>
#include <linux/sched/signal.h>
#include <linux/sched/stat.h>
#include <linux/sched/loadavg.h>
#include <linux/swap.h>
#include <linux/timekeeping.h>
#include <linux/sched.h>

struct resource_info {
    unsigned long total_ram;
    unsigned long free_ram;
    unsigned long used_ram;
    unsigned long total_swap;
    unsigned long free_swap;
    unsigned long procs;
    unsigned long uptime;
    unsigned long load_1;
    unsigned long load_5;
    unsigned long load_15;
};

SYSCALL_DEFINE1(resmon, struct resource_info __user *, info)
{
    struct resource_info kinfo;
    struct sysinfo si;
    struct timespec64 tp;
    unsigned long loads[3];

    // Validate user pointer
    if (!info)
        return -EINVAL;

    // Initialize structures
    memset(&kinfo, 0, sizeof(kinfo));
    memset(&si, 0, sizeof(si));

    // Get system uptime
    ktime_get_boottime_ts64(&tp);
    kinfo.uptime = tp.tv_sec;

    // Get load averages (1, 5, 15 minutes)
    get_avenrun(loads, 0, SI_LOAD_SHIFT - FSHIFT);
    kinfo.load_1 = loads[0];
    kinfo.load_5 = loads[1];
    kinfo.load_15 = loads[2];

    // Get process count
    kinfo.procs = nr_threads;

    // Get memory information
    si_meminfo(&si);
    kinfo.total_ram = si.totalram * (si.mem_unit / 1024);
    kinfo.free_ram = si.freeram * (si.mem_unit / 1024);
    kinfo.used_ram = kinfo.total_ram - kinfo.free_ram;

    // Get swap information
    si_swapinfo(&si);
    kinfo.total_swap = si.totalswap * (si.mem_unit / 1024);
    kinfo.free_swap = si.freeswap * (si.mem_unit / 1024);

    // Copy data to user space
    if (copy_to_user(info, &kinfo, sizeof(kinfo)))
        return -EFAULT;

    return 0;
}
```

Key Features

1. Atomic Operation: All data collected in single kernel execution
2. Kernel Functions Used:
   - `ktime_get_boottime_ts64()` - System uptime
   - `get_avenrun()` - Load averages
   - `nr_threads` - Process count
   - `si_meminfo()` - Memory statistics
   - `si_swapinfo()` - Swap statistics
   - `copy_to_user()` - Safe kernel-to-user data transfer
3. Error Handling:
   - Validates user-space pointer
   - Returns `-EINVAL` for invalid parameters
   - Returns `-EFAULT` for memory copy failures
4. Memory Safety:
   - Uses `memset()` to initialize structures
   - Employs `copy_to_user()` for safe data transfer
   - Checks all kernel function return values

3. Diagram Flow

3.1 System Call flow



3.2 Function Call Hierarchy

```
main()
  │
  ├──→ signal(SIGINT, sig_handler)    [Setup signal handler]
  │
  ├──→ syscall(SYS_RESMON, &info)     [Call sys_resmon]
  │     │
  │     └──→ KERNEL SPACE
  │           │
  │           ├──→ sys_resmon()
  │           │     │
  │           │     ├──→ ktime_get_boottime_ts64()
  │           │     ├──→ get_avenrun()
  │           │     ├──→ si_meminfo()
  │           │     ├──→ si_swapinfo()
  │           │     └──→ copy_to_user()
  │           │
  │           └──→ Return to USER SPACE
  │
  ├──→ display_info(&info)          [Display data]
  │     │
  │     ├──→ printf() [Clear screen]
  │     ├──→ strftime() [Format time]
  │     └──→ printf() [Show metrics]
  │
  ├──→ log_to_file(&info)           [Log data]
  │     │
  │     ├──→ fopen()
  │     ├──→ fprintf()
  │     └──→ fclose()
  │
  └──→ sleep(interval)              [Wait]
        │
        └──→ Loop back to syscall()
```

3.3 Kernel Modification Points

```
Linux Kernel Source Tree (6.1.60)
│
├──── kernel/
│     ├──── Makefile                    [MODIFIED]
│     │     └──── Added: obj-y += custom_syscalls/sys_resmon.o
│     │
│     └──── custom_syscalls/            [CREATED]
│           └──── sys_resmon.c          [NEW FILE]
│                 └──── SYSCALL_DEFINE1(resmon, ...)
│
├──── include/
│     ├──── linux/
│     │     └──── syscalls.h            [MODIFIED]
│     │           └──── Added: asmlinkage long sys_resmon(...)
│     │
│     └──── uapi/asm-generic/
│           └──── unistd.h              [MODIFIED]
│                 ├──── Added: #define __NR_resmon 451
│                 ├──── Added: __SYSCALL(__NR_resmon, sys_resmon)
│                 └──── Modified: #define __NR_syscalls 452
│
└──── arch/arm64/
      └──── (Uses generic syscall table from include/uapi/asm-generic/)
```

4. Commands to Execute Program

Complete Command Sequence (Quick Reference)

```
PREREQUISITES
sudo apt update && sudo apt install -y build-essential libncurses-dev bison flex libssl-dev
libelf-dev bc dwarves git fakeroot

DOWNLOAD KERNEL
cd ~ && mkdir kernel_work && cd kernel_work
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.60.tar.xz
tar -xf linux-6.1.60.tar.xz && cd linux-6.1.60

IMPLEMENT SYSCALL
mkdir -p kernel/custom_syscalls
nano kernel/custom_syscalls/sys_resmon.c  # Create syscall
nano kernel/Makefile                       # Add obj-y line
nano include/uapi/asm-generic/unistd.h     # Add syscall entry
nano include/linux/syscalls.h              # Add declaration

CONFIGURE AND COMPILE
cp /boot/config-$(uname -r) .config
yes "" | make oldconfig
nano .config                       # Fix certificates
make -j$(nproc)                    # Compile (1-3 hours)

INSTALL KERNEL
sudo make modules_install
sudo make install
sudo update-grub
sudo reboot                        # Boot into 6.1.60

CREATE APPLICATION
cd ~ && mkdir resmon_tool && cd resmon_tool
nano resmon.c                      # Create application
gcc -o resmon resmon.c             # Compile
sudo touch /var/log/resmon.log     # Create log
sudo chmod 666 /var/log/resmon.log # Set permissions
```

Resmon.c code

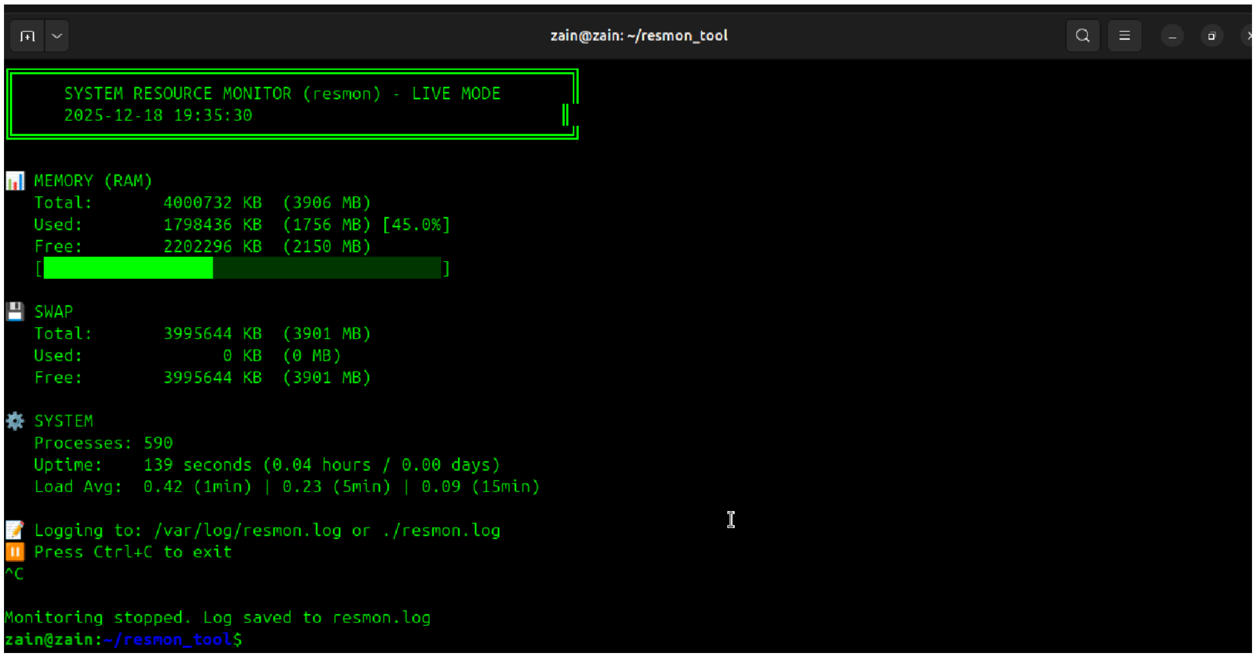
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <time.h>
#include <string.h>
#include <errno.h>
#include <signal.h>

#define SYS_RESMON 451

struct resource_info {
    unsigned long total_ram;
    unsigned long free_ram;
    unsigned long used_ram;
    unsigned long total_swap;
    unsigned long free_swap;
    unsigned long procs;
    unsigned long uptime;
    unsigned long load_1;
    unsigned long load_5;
    unsigned long load_15;
};

volatile sig_atomic_t keep_running = 1;

void sig_handler(int sig) {
    keep_running = 0;
}

void log_to_file(struct resource_info *info) {
    FILE *fp = fopen("/var/log/resmon.log", "a");
    if (!fp) {
        fp = fopen("resmon.log", "a");
        if (!fp) {
            return;
        }
    }

    time_t now = time(NULL);
    char timestamp[64];
    strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", localtime(&now));
    fprintf(fp, "[%s] RAM: %lu/%lu KB (%.1f%%) | Swap: %lu/%lu KB | Procs: %lu | Load: %.2f,
%.2f, %.2f\n",
            timestamp,
            info->used_ram, info->total_ram,
            (info->used_ram * 100.0) / info->total_ram,
            (info->total_swap - info->free_swap), info->total_swap,
            info->procs,
            info->load_1 / 65536.0,
            info->load_5 / 65536.0,
            info->load_15 / 65536.0);

    fclose(fp);
}

void display_info(struct resource_info *info) {
    // Clear screen and move cursor to top
    printf("\033[2J\033[H");

    time_t now = time(NULL);
    char timestamp[64];
    strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", localtime(&now));


printf("╔══════════════════════════════════════════════════════════
══╗\n");
    printf("║      SYSTEM RESOURCE MONITOR (resmon) - LIVE MODE        ║\n");
    printf("║      %s                              ║\n", timestamp);

printf("╚══════════════════════════════════════════════════════════
══╝\n\n");

    printf("📊 MEMORY (RAM)\n");
    printf("   Total:    %10lu KB  (%lu MB)\n", info->total_ram, info->total_ram / 1024);
    printf("   Used:     %10lu KB  (%lu MB) [%.1f%%]\n",
           info->used_ram, info->used_ram / 1024,
           (info->used_ram * 100.0) / info->total_ram);
    printf("   Free:     %10lu KB  (%lu MB)\n",
           info->free_ram, info->free_ram / 1024);

    // RAM usage bar
    int bar_width = 40;
    int filled = (int)((info->used_ram * bar_width) / info->total_ram);
    printf("   [");
    for (int i = 0; i < bar_width; i++) {
        if (i < filled) printf("█");
        else printf("░");
    }
    printf("]\n\n");

    printf("💾 SWAP\n");
    printf("   Total:    %10lu KB  (%lu MB)\n", info->total_swap, info->total_swap / 1024);
    printf("   Used:     %10lu KB  (%lu MB)\n",
           info->total_swap - info->free_swap,
           (info->total_swap - info->free_swap) / 1024);
    printf("   Free:     %10lu KB  (%lu MB)\n\n",
           info->free_swap, info->free_swap / 1024);

    printf("⚙️  SYSTEM\n");
    printf("   Processes: %lu\n", info->procs);
    printf("   Uptime:    %lu seconds (%.2f hours / %.2f days)\n",
           info->uptime, info->uptime / 3600.0, info->uptime / 86400.0);
    printf("   Load Avg:  %.2f (1min) | %.2f (5min) | %.2f (15min)\n",
           info->load_1 / 65536.0,
           info->load_5 / 65536.0,
           info->load_15 / 65536.0);

    printf("\n📝 Logging to: /var/log/resmon.log or ./resmon.log\n");
    printf("⏸️  Press Ctrl+C to exit\n");
}

int main(int argc, char *argv[]) {
    struct resource_info info;
    long ret;
    int interval = 2;  // Default 2 seconds

    // Handle command line argument for interval
    if (argc > 1) {
        interval = atoi(argv[1]);
        if (interval < 1) interval = 2;
    }

    // Setup signal handler for clean exit
    signal(SIGINT, sig_handler);
    signal(SIGTERM, sig_handler);

    // First check if syscall works
    ret = syscall(SYS_RESMON, &info);
    if (ret < 0) {
        fprintf(stderr, "ERROR: System call failed!\n");
        fprintf(stderr, "Error code: %ld (%s)\n", ret, strerror(errno));
        fprintf(stderr, "\nTroubleshooting:\n");
        fprintf(stderr, "1. Check if custom kernel is running: uname -r\n");
        fprintf(stderr, "2. Verify syscall exists: cat /proc/kallsyms | grep sys_resmon\n");
        fprintf(stderr, "3. Check syscall number matches (should be 451)\n");
        return 1;
    }

    // Continuous monitoring loop
    while (keep_running) {
        ret = syscall(SYS_RESMON, &info);

        if (ret < 0) {
            fprintf(stderr, "\nERROR: System call failed during monitoring\n");
            break;
        }

        display_info(&info);
        log_to_file(&info);

        sleep(interval);
    }

    printf("\n\nMonitoring stopped. Log saved to resmon.log\n");
    return 0;
}
```

RUN PROGRAM

```
./resmon                        # Execute monitor
```

5. Expected Output

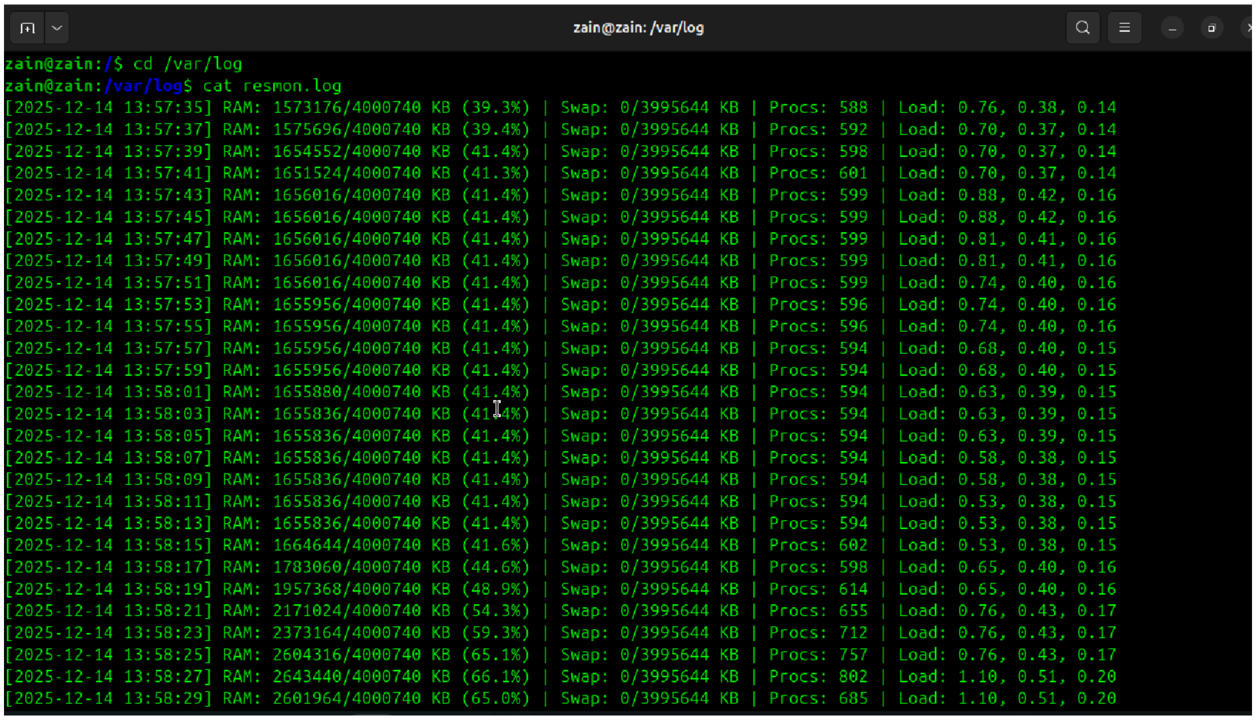
5.1 Terminal Output



zain@zain: ~/resmon_tool

```
    SYSTEM RESOURCE MONITOR (resmon) - LIVE MODE
    2025-12-18 19:35:30

📊 MEMORY (RAM)
   Total:      4000732 KB  (3906 MB)
   Used:       1798436 KB  (1756 MB) [45.0%]
   Free:       2202296 KB  (2150 MB)
   [                                         ]

💾 SWAP
   Total:      3995644 KB  (3901 MB)
   Used:             0 KB  (0 MB)
   Free:       3995644 KB  (3901 MB)

⚙️ SYSTEM
   Processes: 590
   Uptime:    139 seconds (0.04 hours / 0.00 days)
   Load Avg:  0.42 (1min) | 0.23 (5min) | 0.09 (15min)

📝 Logging to: /var/log/resmon.log or ./resmon.log
⏸️ Press Ctrl+C to exit
^C

Monitoring stopped. Log saved to resmon.log
zain@zain:~/resmon_tool$
```

5.2 Log File Output



zain@zain: /var/log

```
zain@zain:/$ cd /var/log
zain@zain:/var/log$ cat resmon.log
[2025-12-14 13:57:35] RAM: 1573176/4000740 KB (39.3%) | Swap: 0/3995644 KB | Procs: 588 | Load: 0.76, 0.38, 0.14
[2025-12-14 13:57:37] RAM: 1575696/4000740 KB (39.4%) | Swap: 0/3995644 KB | Procs: 592 | Load: 0.70, 0.37, 0.14
[2025-12-14 13:57:39] RAM: 1654552/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 598 | Load: 0.70, 0.37, 0.14
[2025-12-14 13:57:41] RAM: 1651524/4000740 KB (41.3%) | Swap: 0/3995644 KB | Procs: 601 | Load: 0.70, 0.37, 0.14
[2025-12-14 13:57:43] RAM: 1656016/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 599 | Load: 0.88, 0.42, 0.16
[2025-12-14 13:57:45] RAM: 1656016/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 599 | Load: 0.88, 0.42, 0.16
[2025-12-14 13:57:47] RAM: 1656016/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 599 | Load: 0.81, 0.41, 0.16
[2025-12-14 13:57:49] RAM: 1656016/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 599 | Load: 0.81, 0.41, 0.16
[2025-12-14 13:57:51] RAM: 1656016/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 599 | Load: 0.74, 0.40, 0.16
[2025-12-14 13:57:53] RAM: 1655956/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 596 | Load: 0.74, 0.40, 0.16
[2025-12-14 13:57:55] RAM: 1655956/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 596 | Load: 0.74, 0.40, 0.16
[2025-12-14 13:57:57] RAM: 1655956/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.68, 0.40, 0.15
[2025-12-14 13:57:59] RAM: 1655956/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.68, 0.40, 0.15
[2025-12-14 13:58:01] RAM: 1655880/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.63, 0.39, 0.15
[2025-12-14 13:58:03] RAM: 1655836/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.63, 0.39, 0.15
[2025-12-14 13:58:05] RAM: 1655836/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.63, 0.39, 0.15
[2025-12-14 13:58:07] RAM: 1655836/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.58, 0.38, 0.15
[2025-12-14 13:58:09] RAM: 1655836/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.58, 0.38, 0.15
[2025-12-14 13:58:11] RAM: 1655836/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.53, 0.38, 0.15
[2025-12-14 13:58:13] RAM: 1655836/4000740 KB (41.4%) | Swap: 0/3995644 KB | Procs: 594 | Load: 0.53, 0.38, 0.15
[2025-12-14 13:58:15] RAM: 1664644/4000740 KB (41.6%) | Swap: 0/3995644 KB | Procs: 602 | Load: 0.53, 0.38, 0.15
[2025-12-14 13:58:17] RAM: 1783060/4000740 KB (44.6%) | Swap: 0/3995644 KB | Procs: 598 | Load: 0.65, 0.40, 0.16
[2025-12-14 13:58:19] RAM: 1957368/4000740 KB (48.9%) | Swap: 0/3995644 KB | Procs: 614 | Load: 0.65, 0.40, 0.16
[2025-12-14 13:58:21] RAM: 2171024/4000740 KB (54.3%) | Swap: 0/3995644 KB | Procs: 655 | Load: 0.76, 0.43, 0.17
[2025-12-14 13:58:23] RAM: 2373164/4000740 KB (59.3%) | Swap: 0/3995644 KB | Procs: 712 | Load: 0.76, 0.43, 0.17
[2025-12-14 13:58:25] RAM: 2604316/4000740 KB (65.1%) | Swap: 0/3995644 KB | Procs: 757 | Load: 0.76, 0.43, 0.17
[2025-12-14 13:58:27] RAM: 2643440/4000740 KB (66.1%) | Swap: 0/3995644 KB | Procs: 802 | Load: 1.10, 0.51, 0.20
[2025-12-14 13:58:29] RAM: 2601964/4000740 KB (65.0%) | Swap: 0/3995644 KB | Procs: 685 | Load: 1.10, 0.51, 0.20
```