Introduction to Cloud COMPUTING

Giuliano Taffoni - I.N.A.F.

"Virtualization"

2023 @ Università di Trieste

Outline

Intro to Benchmarking
a
Linux platform

Benchmark
Tutorial

Overview

- Why should I measure the performance?
- What are the main component to address performances?
- A benchmarking methodology
- HW vs VM vs Container: the Cloud
- Tools and services

Performance measure

"…maintaining optimal server performance is crucial for ensuring smooth operations and meeting business objectives."

- measure how efficiently a server handles workloads,
- identify bottlenecks,
- optimize system resources.

Performance analysis and benchmarking

Why Performance Analysis is Important

- Maximizing Efficiency: Understanding how the server utilizes its CPU, memory, storage, and network resources allows for fine-tuning and making better use of the hardware's full potential.
- Cost Optimization: By identifying underutilized or overburdened resources, businesses can make informed decisions about scaling or redistributing workloads, reducing unnecessary costs related to hardware upgrades or cloud services.
- Troubleshooting Issues: Performance bottlenecks can cause slowdowns or system failures. Regular analysis helps in proactively identifying and addressing these issues before they impact users or critical applications.

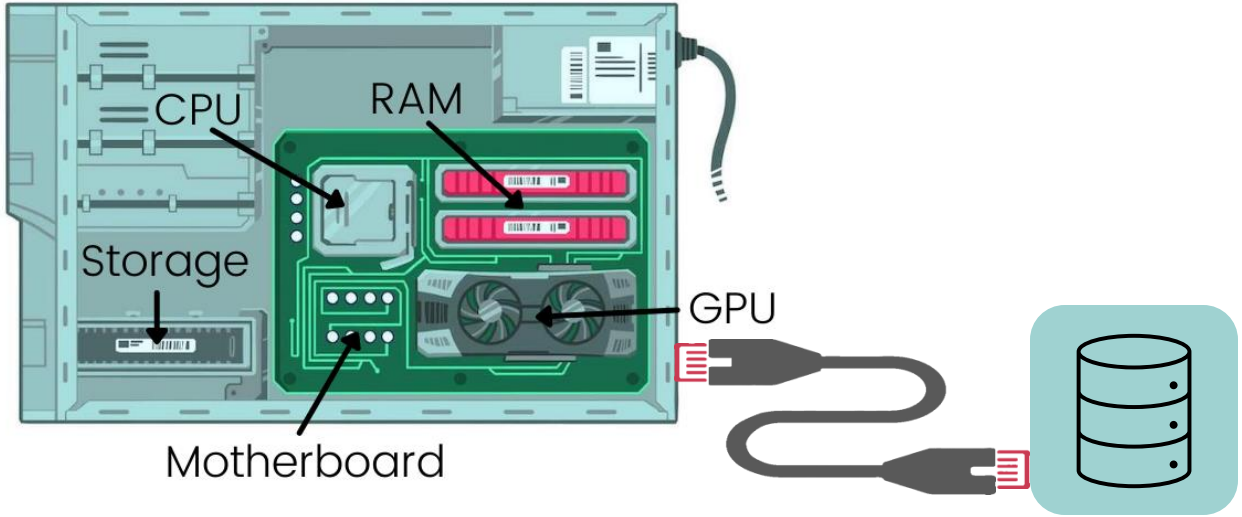
Why Performance Analysis is Important

- Capacity Planning: Benchmarking and analysis provide data for predicting how servers will behave under increasing workloads, helping with capacity planning and infrastructure scaling.
- System Stability and Reliability: A well-performing server is less prone to crashes and instability. Ensuring that it can handle peak loads and performance spikes is key to maintaining high availability.
- Security Considerations: Poor performance can sometimes indicate deeper security vulnerabilities like denial-of-service (DoS) attacks or malware that uses excessive resources.

Benefits of Benchmarking

- Identifying Hardware Limitations: Benchmarking helps in understanding the hardware's limits and provides a basis for upgrading or optimizing hardware components.
- Optimal Resource Allocation for Virtual Machines: For environments running virtual machines (VMs), analysing the performance of both physical hardware and VMs helps in allocating resources appropriately between them. This ensures that VMs don't overutilize or underutilize the underlying hardware.
- Comparing System Upgrades: When upgrading hardware or software, benchmarking provides a clear way to compare the old and new setups, ensuring that the upgrades provide measurable benefits.
- Improving System Performance: After identifying performance issues, administrators can tune configurations, optimize the operating system, or apply specific hardware changes to improve overall performance.

Which components should I test?



There are a lot of tools available on the Internet that can perform system testing on Linux. We can use them to run different tests and benchmark various components, such as the Central Processing Unit (CPU), the Graphics Processing Unit (GPU), memory, database, and more.

Critical components

CPU (Central Processing Unit) Performance:

- Why Test?: The CPU handles all computational tasks. Measuring its performance helps in understanding how efficiently it handles processes and threads, and whether there are bottlenecks in processing power.
- Key Metrics: CPU utilization, load average, number of processes, context switches, and interrupt rates.
- Tools: stress-ng, sysbench, htop, mpstat, HPL, HPCG, SPEC

Memory (RAM) Performance:

- Why Test?: Memory is critical for storing active data and processes. Performance issues in memory, such as excessive swapping or memory leaks, can severely affect server speed.
- Key Metrics: Memory usage, swap activity, page faults, cache hit/miss ratio.
- Tools: vmstat, free, memtester, stress-ng.

Critical components

Disk I/O (Input/Output) and Storage Performance:

- Why Test?: Disk I/O performance affects how quickly data is read from or written to disk, impacting databases, file systems, and applications. Disk bottlenecks can cause slow system response times. Modern servers use different types of storage like RAID arrays or SSDs, which perform differently under various workloads.
- Key Metrics: Read/write speed, IOPS (Input/Output Operations Per Second), disk latency, queue length.
- Tools: fio, iostat, hdparm, dd,

Network Performance:

- Why Test?: For servers involved in network-heavy operations (e.g., web servers, database servers), network throughput and latency are critical factors. Network bottlenecks can result in slow or dropped connections.
- Key Metrics: Bandwidth utilization, latency, packet loss, jitter.
- Tools: iperf, netstat, ping, iftop.

Additional Considerations

Contextual Testing (Workload-specific):

- If the server is running specific applications, such as a web server, database, or containerized services, benchmark those workloads directly. For example, benchmarking a web server might involve tools like *ApacheBench* or *siege* to simulate web traffic.

Power and Energy Consumption:

- In some scenarios, particularly in data centers or environments with strict energy consumption regulations, power efficiency can be a key performance metric. Tools like *powertop* can help in measuring power consumption during different workloads.

Comparison Between Host and VM Performance:

- Measure performance on the physical hardware and compare it to the VM's performance to understand how virtualization overhead affects the system. Pay close attention to resource contention and how resources are shared between VMs.

Measure performance

Measure the overall performance of a system (including network and memory)

Common tools used in HPC are:

1. HPLinpack http://top500.org
2. STREAM http://www.cs.virginia.edu/stream/
3. HPC Challenge http://icl.cs.utk.edu/hpcc
4. Graph 500 http://graph500.org
5. NAS Parallel Benchmarks http://www.nas.nasa.gov/publications/ npb.html

HP Linpack

The most famous HPC benchmark – used for the “Top500” ranking

Solve a system of n linear equations using Gaussian elimination (matrix is dense)

For a given problem size N, HPL performs floating point operations proportional to N*N*N operations while performing memory reads and write proportional to N*N.
That means, if the problem size doubles, the number of floating-point operations increases by a factor of 8 while the number of memory operations only grows by a factor of 4.
This property of HPL means that it performs very well on systems that have many flop functional units relative to data movement support. The result is that HPL tends to represent a very optimistic performance prediction that can only be matched by a small number of real scientific applications.

HPCG

HPCG uses a simple implementation of the Conjugate Gradients and multigrid algorithms. It generates and uses sparse data structures that have a very low compute-to-data-movement ratio.

HPCG has a floating-point operations rate proportional to N and a memory access rate also proportional to N.
This property means that HPCG performance is strongly influenced by memory bandwidth.

https://github.com/hpcg-benchmark/hpcg/

Central Processing Unit

It is responsible for executing all the processes, calculations, and tasks. A server's overall speed, responsiveness, and ability to handle concurrent workloads are highly dependent on the efficiency and power of its CPU.

In a virtualized environment, CPU testing becomes even more critical since physical CPU resources are shared between the virtual machines (VMs) running on the host. The virtual CPUs (vCPUs) allocated to each VM must be carefully monitored to ensure that the host and all guest VMs operate efficiently without resource contention.

Testing the CPU

Floating point operations per second (FLOPS, flops or flop/s).

$$\mathbf{FLOPS} = \mathbf{cores} \times \frac{\mathbf{cycles}}{\mathbf{second}} \times \frac{\mathbf{FLOPs}}{\mathbf{cycle}}.$$

Floating-point operations per clock cycle per core is CPU dependent (e.g. Intel FF32 = 64)…there is much more here but…it will be discussed in HPC course

When testing CPU performance, several key metrics provide insight into how well the CPU handles the system's workload.

Key CPU Performance Metrics

1. CPU Utilization (%): Measures the percentage of CPU capacity being used. High sustained utilization may indicate a CPU bottleneck.
   Consideration: Utilization near 100% for long periods may point to insufficient CPU resources.
2. Load Average: Represents the average number of processes waiting for CPU time. A high load average relative to the number of CPU cores indicates that processes are queuing up.
   Consideration: For multi-core systems, the load average should be compared to the number of cores. For example, a load average of 4 on a 4-core machine is normal, but a load average of 10 would suggest the CPU is overburdened.
3. Context Switches: Tracks how often the CPU switches between processes. Excessive context switching can degrade performance, as the CPU spends more time switching between tasks than processing them.
4. Interrupt Rates: Measures the number of hardware interrupts the CPU has to handle. High interrupt rates can indicate hardware issues or excessive device activity.

Key CPU Performance Metrics

5. CPU Wait Time (I/O Wait): The percentage of time the CPU is idle but waiting for I/O operations (like disk or network) to complete. High I/O wait time can indicate that the CPU is being held back by slow I/O devices.

6. CPU Frequency: In dynamic frequency scaling environments (such as Intel's SpeedStep or AMD's Cool'n'Quiet), the CPU frequency can vary based on load. Monitoring this ensures the CPU is operating at the correct speed under load.

7. vCPU Overcommitment (for VMs): In virtualized environments, it's important to measure how many virtual CPUs are allocated across VMs relative to the physical CPUs (pCPUs) on the host. Overcommitment occurs when the number of vCPUs exceeds the available pCPUs, which can cause performance degradation due to context switching between VMs.

HOW to test it

1. Baseline Testing

Start by measuring CPU performance under typical workloads to establish a baseline. This allows you to understand how the CPU performs under normal conditions.

2. Stress Testing

Stress testing pushes the CPU to its limits, simulating high usage scenarios to identify how the system handles extreme conditions. This can help find the upper bounds of the CPU's capabilities.

Some tools to install

htop:
An interactive system monitor that provides real-time information on CPU utilization, load average, and the number of running processes. It visually represents per-core utilization.
Usage: Simply run htop to monitor the system in real time.

mpstat:
Part of the sysstat package, mpstat provides detailed CPU usage statistics, breaking down the usage by user processes, system processes, and I/O wait time.

- Usage:

```
> mpstat -P ALL 1
```

This will display CPU usage statistics for all CPU cores every second.

Base line testing

HPL testing

> apt install hpcc

Tweak HPL parameters: https://www.advancedclustering.com/act_kb/tune-hpl-dat-file/
HPL Calculator: https://hpl-calculator.sourceforge.net/

(You may instead use HPCG)

Stress tools

stress-ng:
A powerful tool designed to stress test a CPU by generating various types of load. It can create stress on CPU, cache, and memory.
Usage Example:

```
stress-ng --cpu 4 --timeout 60s
```

This command will stress 4 CPU cores for 60 seconds.

sysbench:
A versatile benchmarking tool that can be used to stress CPU and perform performance tests on various server subsystems.
Usage Example:

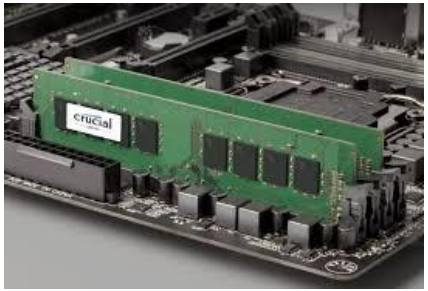
```
sysbench --test=cpu --cpu-max-prime=20000 run
```

This command will benchmark the CPU by calculating prime numbers up to 20,000.

CPU perf analysis

1. High CPU Utilization: If CPU utilization is consistently high (above 80-90%), it might be necessary to increase CPU resources (e.g., upgrade to a higher clock speed or more cores) or optimize the running applications.
2. High Load Average: Load average exceeding the number of available CPU cores suggests that the CPU cannot keep up with the demand. This could be a sign of inadequate CPU resources or inefficient application design.
3. High Context Switching: Excessive context switching may indicate too many processes competing for CPU time. It might be necessary to tune the operating system's scheduling policies or reduce the number of concurrent processes.
4. High Interrupt Rates: High interrupt rates can indicate excessive hardware interrupts (e.g., from network or disk I/O). It may be necessary to investigate whether hardware components are properly configured.
5. vCPU Overcommitment: In a virtualized environment, overcommitting vCPUs (assigning more vCPUs than the available physical CPUs) can lead to poor performance. Monitoring and adjusting the vCPU allocation across VMs can help alleviate this issue.

Testing the memory



Memory is one of the most vital components of a server, affecting how quickly applications can access and manipulate data. Inadequate memory resources, poor memory management, or issues with memory speed can lead to severe performance bottlenecks. For example, if the system runs out of physical memory, it may start using swap space (disk-based memory), which can significantly degrade performance.

Testing memory performance ensures that the system can handle workloads effectively without encountering delays due to insufficient or slow memory. This is particularly important in environments running databases, virtual machines, or in-memory data stores (like Redis).

Key Memory Performance Metrics

Memory Usage: Measures how much RAM is actively being used by processes versus how much is available. Excessive usage can indicate the need for more memory or optimized software configurations.

Swap Activity: Swap usage occurs when the system runs out of RAM and starts using disk space as an extension of physical memory. Frequent swap activity slows down performance significantly.

Page Faults: A page fault occurs when a program accesses a portion of memory that isn't currently mapped to physical RAM, causing the system to retrieve it from disk (if it's swapped out). High page fault rates can signal inefficient memory usage or insufficient RAM.

Key Memory Performance Metrics

Memory Bandwidth: The rate at which data can be read from or written to memory. Insufficient bandwidth can cause bottlenecks, especially in memory-intensive applications.

Cache Hit/Miss Ratio: CPUs rely on fast-access memory caches (L1, L2, L3) to speed up processing. The cache hit ratio indicates how often the CPU successfully retrieves data from cache versus fetching it from slower main memory.

NUMA (Non-Uniform Memory Access) Performance: In multi-socket systems, memory access speed can vary depending on which CPU core is accessing the memory (due to NUMA architecture). NUMA-aware benchmarks can help identify whether memory access latency is a concern.

How to Test Memory Performance

1. Baseline Testing -- Start by monitoring how the system utilizes memory under normal conditions. This will provide a baseline against which you can compare performance under stress or after configuration changes.

2. Stress Testing -- Memory stress tests push the system to its limits, ensuring that the server can handle memory-heavy workloads. It simulates conditions where the system might run out of RAM or need to allocate large amounts of memory to specific applications.

3. Virtual Machine Memory Testing-- In virtualized environments, testing memory performance within virtual machines is key. Virtual machines often share physical memory (using techniques like memory ballooning), so it's important to ensure that VMs don't suffer from memory starvation or excessive swapping.

Memory Benchmarking Tools

memtester: A straightforward memory stress-testing tool designed to allocate and test memory for errors. It's useful for checking system stability and detecting faulty memory modules.
Usage Example:

```
memtester 512M 5
```

sysbench (memory mode): It tests memory bandwidth by continuously reading and writing to memory.
Usage Example:

```
sysbench --test=memory --memory-total-size=10G run
```

Memory monitor Tools

vmstat: A system monitor that displays memory usage, swap activity, page faults, and more. It provides real-time monitoring of memory performance under various workloads.
Usage Example:

```
vmstat 1
```

numactl: Useful for systems with NUMA architectures, numactl provides insights into memory access patterns and performance. You can test memory bandwidth and latency between different NUMA nodes.
Usage Example:

```
numactl --hardware
```

Analyzing Performance Results

High Memory Usage: If memory usage is constantly high, the system may be running too many memory-intensive applications, or it may require more RAM. Excessive memory usage can lead to swapping, which should be avoided.

Swap Activity: Swap activity should be minimized because it involves using disk space (which is much slower than RAM) as memory. If swap usage is high, it could indicate that the system is running out of physical memory.

High Page Fault Rate: A high rate of page faults suggests that the system frequently retrieves data from the disk, rather than from memory. This could be a result of insufficient RAM, or it could indicate inefficient memory usage by applications.

Analyzing Performance Results

Low Memory Bandwidth: If memory bandwidth is low (especially when tested under stress conditions), the system may be constrained by memory speeds. This could suggest a need for faster memory (higher clock speeds) or optimizations in memory usage patterns.

NUMA-related Latency: In multi-socket systems with NUMA architectures, high latency may indicate that processes are frequently accessing memory from a different NUMA node. This can be mitigated by optimizing the placement of processes or making applications NUMA-aware.

Cache Misses: High cache miss rates indicate that the CPU is often unable to retrieve data from its fast-access caches and must instead fetch it from main memory, slowing down performance. Improving cache utilization through code optimizations or CPU tuning may help.

Benchmarking mem in VMs

Ballooning: Many hypervisors use a technique called memory ballooning to dynamically allocate memory between virtual machines. Monitoring how this process impacts VM memory performance is critical, especially in environments where memory is overcommitted.

Overcommitting Memory: In virtualized environments, it's possible to allocate more virtual memory across VMs than the host physically has. While this allows better utilization of resources, it can lead to performance issues if the actual memory demand exceeds the host's capacity.

Swap in VMs: Just as on a physical server, excessive swapping in a virtual machine can cause performance degradation. Monitoring the VM's memory usage and swap activity can help ensure that the VM has adequate memory resources allocated.

Disk and Storage Performance

Disk I/O (Input/Output) is a critical factor in server performance, especially for databases, web servers, and applications that rely on frequent disk access. Slow disk performance can lead to bottlenecks where applications are waiting on data retrieval or writing operations, causing delays and system slowdowns. In addition, storage subsystems like RAID arrays and SSDs have different performance characteristics that must be optimized for the server's workload.

Cloud and Virtualized environments also introduce additional complexity since multiple virtual machines may share the same physical storage, increasing the potential for resource contention.

Key Performance Metrics

Read/Write Throughput: Measures the speed at which data is read from or written to the disk, usually measured in MB/s or GB/s. High throughput is critical for applications that require fast access to large datasets.

IOPS (Input/Output Operations Per Second): IOPS measures how many individual read or write operations the disk can perform per second. High IOPS is important for transactional databases and applications that require frequent small I/O operations.

Disk Latency: The time it takes for a single I/O operation to complete. Low latency is important for reducing response times in applications that rely on quick data access.

Key Performance Metrics

Queue Depth: Represents the number of I/O operations waiting to be processed by the disk. A high queue depth can indicate that the disk is overwhelmed and may be a sign that more I/O capacity is needed.

RAID Performance: Many servers use RAID to improve performance, reliability, or both. RAID configurations like RAID 0, 1, 5, and 10 have different performance characteristics and trade-offs between redundancy and speed.

SSD vs HDD Performance: Solid-State Drives (SSDs) provide much faster IOPS and lower latency compared to Hard Disk Drives (HDDs). Benchmarking should take into account the type of storage being used.

TRIM (SSD Maintenance): TRIM is a feature in SSDs that helps maintain performance over time by managing how data is erased and stored.

How to Test Disk perf

- Baseline Testing: Measure disk performance under normal workloads to establish baseline performance.
- Stress Testing: Push the disk subsystem to its limits to identify peak performance and potential bottlenecks.
- Virtualization: In virtualized environments, test both host and VM disk performance.

Benchmarking Tools

- fio (Flexible I/O Tester)
- iostat (I/O Statistics)
- hdparm (Disk Tuning and Benchmarking)
- dd (Disk Data Copying and Benchmarking)
- bonnie++ (File System Benchmarking)
- mdadm (RAID Management and Benchmarking)
- IOZone

Some examples

```
$ fio --name=readtest --rw=read --bs=4k --size=1G --
numjobs=4 --runtime=60 --group_reporting
```

This command runs a read test using 4KB block sizes on a 1GB file with 4 parallel jobs, running for 60 seconds.

```
bonnie++ -d /path/to/directory -s 4G
```

A tool designed to test file system performance, including file creation, deletion, and sequential/random I/O. It provides detailed reports on read/write throughput and CPU usage during I/O operations.

Some examples

```
dd if=/dev/zero of=testfile bs=1G count=1 oflag=dsync
```

While primarily used as a file-copying utility, dd can be used to benchmark sequential read and write speeds by reading from or writing to disk.

```
hdparm -t /dev/sda
```

Primarily used for testing and tuning the performance of physical hard drives and SSDs. hdparm can measure read speeds and tweak disk parameters for optimization.

Some examples

```
dd if=/dev/zero of=testfile bs=1G count=1 oflag=dsync
```

While primarily used as a file-copying utility, dd can be used to benchmark sequential read and write speeds by reading from or writing to disk.

```
hdparm -t /dev/sda
```

Primarily used for testing and tuning the performance of physical hard drives and SSDs. hdparm can measure read speeds and tweak disk parameters for optimization.

Some examples

IOzone performs 13 types of test (Read, Write, Re-read, Re-write, Random Read, Random Write, Backward Read, Record Re-Write, Stride Read, Fread, Fwrite, Freread, Frewrite).
If you are executing iozone test on a database server, you can focus on the 1st 6 tests, as they directly impact the database performance.
IOZone allows to vary the number of threads used and can be used to test parallel file systems on multi node environments.

```
$ ./iozone -a -b output.xls
```

Analyzing Disk Performance Results

Low Throughput: May indicate disk fragmentation, misconfiguration, or nearing capacity.

High Latency: May suggest I/O contention or hardware limitations.

Low IOPS: Disk subsystem may not handle frequent small I/O operations well.

RAID Performance: Different RAID levels offer different trade-offs between speed and redundancy.

Performance on VMs

I/O Contention: Multiple VMs sharing the same physical disk may cause performance degradation.

Thin Provisioning: Allocating more virtual disk space than physically available can lead to performance issues.

Key concepts to know…

Main tools to measure performance and monitor the status

that’s all, have fun