PERFORMANCE

Performance Considerations for Netframe Deployments

How to think about CPU, memory, storage I/O, and network performance on Netframe, with patterns for benchmarking your own workloads

May 2026 · Version 1.0

Contents

Overview

Netframe is built on KVM, the same hypervisor that runs the majority of the public cloud and the bulk of Linux-based enterprise virtualisation worldwide. Under most workload profiles, the overhead of virtualisation on KVM is small: typically a few percent on CPU-bound workloads, a few percent on memory-bound workloads, and within single-digit percent on well-tuned storage and network paths.

What this means in practice is that performance on Netframe is rarely about the hypervisor and almost always about the choices made around it: the hardware spec, the storage topology, the network design, and the guest OS configuration. This document is a short, practical overview of those choices and a starting point for benchmarking your own workloads to validate them.

It does not contain published benchmark numbers. The reason is that meaningful performance numbers are workload-specific, and the only numbers worth quoting are the ones you have measured against your own workloads on your own hardware. The patterns below describe how to get those numbers in a defensible way.

CPU

KVM's CPU virtualisation is implemented in hardware via Intel VT-x or AMD-V. The overhead is in the single-digit percent range for most workloads. Two factors matter more than the hypervisor itself.

CPU pinning and NUMA placement. Modern multi-socket servers have non-uniform memory access: a CPU accessing memory attached to its own socket is materially faster than the same CPU accessing memory attached to the other socket. For VMs that are sensitive to memory latency (databases, real-time workloads), pinning the VM's vCPUs to a single NUMA node and aligning memory allocation with that node typically delivers a measurable improvement. Netframe Manager exposes the placement controls; the default scheduler does the right thing for most workloads without explicit pinning.

vCPU over-commit. Allocating more vCPUs in total across VMs than the host has physical cores is normal and works well for typical mixed workloads. Aggressive over-commit ratios (above 4:1) start to show in tail latency, particularly for interactive workloads. The right ratio for your environment is best determined empirically against the workload mix you will actually run.

For benchmarking, the workloads worth measuring on the CPU dimension are: a representative compute-heavy workload (typically a database query mix or a build pipeline), a representative latency-sensitive workload (typically an interactive application's response time), and a representative aggregate (overall throughput at expected concurrency).

Memory

Memory performance on KVM is close to bare metal for most workloads. The features worth understanding are memory overcommit, ballooning, and Kernel Samepage Merging (KSM).

Overcommit. Allocating more memory across VMs than the host physically has is supported, with paging to disk as the fallback when actual usage exceeds physical memory. Conservative environments avoid overcommit; cost-optimised environments use it deliberately for workloads with predictable memory profiles. Tail latency degrades sharply when overcommit drives the host into swap; this should be monitored as a leading indicator.

Ballooning. Allows the host to reclaim memory from VMs that are not actively using their full allocation. Useful for environments where VM memory sizing is approximate and reclaim is needed to fit additional VMs. The trade-off is that ballooning can interact poorly with applications that pre-allocate large buffers (JVM heap, database buffer pool); test for your specific workloads.

KSM. Detects identical memory pages across VMs and merges them, reducing physical memory consumption. Most useful when running many VMs with the same guest OS and similar workloads (typical VDI scenarios). Has a CPU cost; disable in environments where the cost outweighs the savings.

For benchmarking, the dimensions worth measuring are: working-set fit in physical memory (does the host swap?), allocation latency under contention (how long does a memory allocation take when the host is busy?), and overall VM memory headroom (how much margin do you have before tail latency degrades?).

Storage I/O

Storage is the dimension where the platform-level choices matter most. Netframe uses NFS as the shared storage backend; the performance characteristics of an NFS-based environment are well understood and predictable, but they depend heavily on the appliance, the network, and the mount configuration.

The factors worth tuning explicitly:

Network path. A 10 GbE network supports roughly 1.0–1.2 GB/s of throughput per host before becoming the bottleneck. A 25 GbE or 100 GbE path raises that ceiling. For dense compute hosts with multiple I/O-heavy VMs, the storage network is usually the first bottleneck encountered.

NFS version and mount options. Modern NFSv4.1 with parallel I/O and appropriate read-ahead settings significantly outperforms NFSv3 with conservative defaults. Validate the mount options against your appliance vendor's recommendations.

Appliance tier. NVMe-backed appliances deliver materially lower latency than SATA-backed appliances for small-block random I/O. Match the tier to the workload: tier-1 databases on NVMe, archive workloads on bulk SATA.

Caching. Most enterprise NFS appliances have substantial read and write caches. Cache sizing and the write-cache flush policy are the largest single performance levers available on the storage appliance side.

For benchmarking, three patterns:

- **Sequential throughput.** Measures the network path and the storage backend's bulk performance. Tools: `fio` with sequential read and write workloads at 1MB block size.
- **Random IOPS at small block sizes.** Measures the storage backend's ability to service the kind of I/O databases produce. Tools: `fio` with random read/write at 4KB and 8KB block sizes.
- **Mixed workload latency.** Measures real-world performance under the workload mix you will actually run. Tools: workload-specific (HammerDB for OLTP, pgbench for PostgreSQL, your application's own benchmark mode).

The single most useful number to know about your storage path is the 99th-percentile latency at 70% of your expected I/O load. If that number is acceptable, the platform is sized correctly. If it is not, no amount of CPU or memory tuning will compensate.

Network

VM network performance on Open vSwitch is close to bare metal for most workloads. The hypervisor adds a small per-packet overhead; for bandwidth-heavy workloads, the relevant ceiling is the physical NIC and the switch fabric, not OVS.

Factors worth understanding:

NIC offload features. Modern NICs offload TCP segmentation, checksum calculation, and other per-packet work from the CPU. Make sure these are enabled (they are by default in the Netframe-hardened image) and that the bonding mode being used supports them.

VirtIO drivers in guests. Linux guests use VirtIO networking by default with very low overhead. Windows guests should have the VirtIO drivers installed (Netframe Converter handles this for standard images during migration). Windows guests running on emulated network drivers will see significantly higher CPU usage and lower throughput.

Bond mode. LACP (802.3ad) provides higher aggregate throughput than active-backup for workloads with multiple TCP flows, but does not improve single-flow throughput. For environments dominated by a few high-throughput flows, NIC speed matters more than bond mode.

For benchmarking, the patterns are: throughput between two VMs on the same host (measures the OVS path), throughput between two VMs on different hosts (measures the OVS path plus the inter-host network), and throughput from a VM to an external destination on the workload network (measures the path your applications will actually use).

`iperf3` is the standard tool for all three. Run for several minutes to let throughput stabilise.

Putting it together

A useful framing for performance work on Netframe is to spend time on the dimensions that have the largest impact for your workload, and to validate empirically rather than relying on published benchmarks. The order of impact for most environments is:

1. Storage path (network, appliance, mount options).
2. Memory headroom (avoid swap, measure tail latency under load).
3. CPU placement (NUMA awareness for latency-sensitive workloads).
4. Network (NIC speed, VirtIO drivers in Windows guests).
5. Hypervisor overhead (rarely the bottleneck).

If a workload is performing poorly on Netframe, the right diagnostic order is usually the reverse: rule out storage first, then memory, then CPU placement. Most performance problems we are asked to investigate turn out to be one of the first three.

A benchmarking template

For new deployments, the recommendation is to run a standardised benchmark suite against the environment during the build phase, before any production workloads arrive. The numbers become the baseline against which all future performance investigations are compared.

The minimum-credible suite:

- `fio` storage benchmarks: sequential 1MB, random 4KB, random 8KB; read, write, and 70/30 mix; measured at 99th-percentile latency.
- `iperf3` network benchmarks: VM-to-VM same host, VM-to-VM different host, VM-to-external; measured throughput over 60 seconds.
- A representative application-level benchmark for the dominant workload class (HammerDB, pgbench, or equivalent).

Capture the results, the date, the hardware configuration, and the Netframe version. Re-run after major platform upgrades and significant hardware changes. A baseline that you can compare against is worth more than a benchmark number in absolute terms.

Next steps

If you are sizing or evaluating performance on a new Netframe deployment, the practical next steps are:

1. Decide the storage tier mix that matches your workload categories. Validate by `fio` on the actual appliance.
2. Decide the network topology and confirm `iperf3` throughput meets the design target.
3. Capture an initial baseline using the suite above. Document it alongside the build documentation.

4. Contact Neon Dynamics for a performance review if the initial baseline does not match expectations. There are usually one or two configuration changes that move the numbers materially.

Performance on Netframe is not magic; it is the result of sensible choices on each of the dimensions above, validated empirically against the workloads you will actually run.