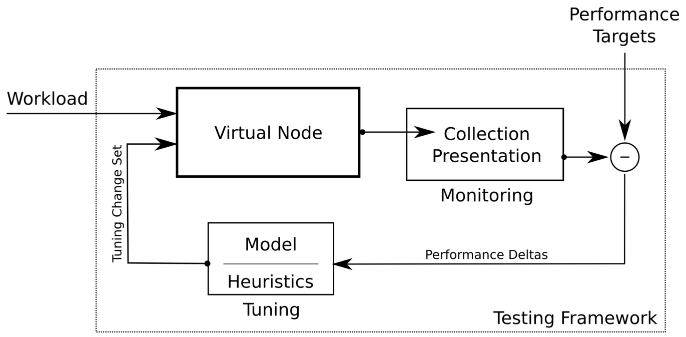
Performance Tuning Overview

Understanding tuning and its workflow will help you create production-ready systems that meet prescribed performance expectations.

Tuning is the process of adjusting system configuration and operational parameters to eliminate performance bottlenecks. It works together with a monitoring process to accomplish a set of performance goals associated with a working virtual node. The overall workflow is illustrated in the following diagram.

Monitoring and Tuning



Tuning is performed on a virtual node under a specified workload with the goal of ensuring that the host and all guest images operate as expected. Achieving a desired set of performance targets usually involves tuning the host and each of the guests. In each case the same workflow applies, just replacing the virtual node in the figure with the specific sub-target in consideration.

In this workflow:

- A monitoring process mines target data. Mining includes collecting data from the virtual node and organizing it to be presented in a meaningful way, including plots, tables, and lists.
- The data is compared with the performance targets set as expectations from the virtual node, and the difference is used as an input to the tuning process.
- The tuning process uses a performance model and possibly a set of heuristics to suggest a tuning change set. The tuning change set is then applied to the target, monitoring resumes, and the cycle continues until the desired set of goals for the virtual node is achieved or is within an expected range.

This cycle is executed within the context of a testing framework that includes hardware, software, and networking components as necessary.

The tuned virtual node is eventually moved into a production environment with the expectation that it will behave as it did while under test. But as it often happens, further monitoring and tuning will likely be required to make final adjustments and have the node(s) meet the original performance target expectations. Furthermore, even if everything works as tested, ongoing monitoring and tuning capabilities will always be required in a production environment to understand what is happening inside the node, to make improvements to meet new system demands, and to dynamically modify operational details when necessary.

There are limitations to the monitoring and tuning processes while the virtual node is in production. For example, you won't be able to include hypervisor kernel configuration changes in a tuning change set unless you take the virtual node off line. Source debugging will also be very limited, or even completely unavailable, as it will likely make production applications unavailable. Nevertheless, having monitoring and tuning capabilities, together with a convenient set of management tools, is a key component in the daily life of a virtual node while in production.

Performance tuning is highly dependent on the nature of the target and on the performance goals. There is no *one size fits all* option for the monitoring and tuning processes.

For example, when tuning a particular VM or container, one of the first questions likely to arise is whether a user application is I/O- or CPU-bound. If it is CPU-bound, then the monitoring process will emphasize tools that collect CPU statistics, and the presentation layer will highlight data such as CPU usage and interrupt latencies. The tuning process will focus on elements such as number of virtual CPUs, a CPU to socket mapping, affinity and pining options, and memory and cache variables. It will likely exclude parameters related to storage and virtual networking. The heuristics that may be applied will likely relate to elements such cache behavior and interrupt frequency. The tuning change set will also place emphasis on those configuration and operational elements that are more relevant to this particular case. In the end, all parameters will contribute to some degree to the final behavior of the virtual node, but it will be in the details of how they are used that the requirements for each use case will be addressed.

Performance Development, Tuning, and Platform Virtualization

Wind River Linux platform virtualization provides an extensive set of tools to help with the implementation of the monitoring and tuning processes. For monitoring, the following debugging, profiling, and tracing tools are available for the host and guest systems.

- Debugging:
    - Host and guest kernel debugging with KGDB and support for kprobes.
    - Host and guest application debugging with GDB.
    - *vmitools* for VM introspection (monitoring the run time state of a VM from within the hypervisor or another VM).
    - Support for native Linux monitoring utilities such as the **systats** family of commands (**iostat**, **vmstats**, **sar**).
- Profiling:

    - **perf** for host and guests, and for user space applications.
- Tracing:
    - **lttng**, **lttng2** and **lttng2-ust** for kernel and user space tracing.
    - **ftrace**, the internal kernel tracing tools and associated user space helpers.
    - **virtio-trace**, a tool to collect VM kernel tracing data that builds on **ftrace** and an enhanced **virtio-serial** interface.

For the tuning process, Wind River provides guidelines and tuning options at different system levels that you can use to build a tuning change set. In general, you first build the tuning model based upon the hardware, software, and networking resources in the virtual node. You then use the guidelines, general Linux knowledge, and your own expertise to select a tuning change set. In addition, you can also use tuning profiles to specify performance options.

Tuning Profiles

A tuning profile is an engineered set of configuration and operational parameters that can be applied to a host or guest image to tune it for a particular behavior.

The default profile

A tuning profile where all kernel images and root files systems have been configured to operate in a virtualized environment. The host kernel is configured to perform as a hypervisor with access to all additional components of the virtualized platform, such as fast packet paths for optimal networking. Guest kernels are paravirtualized appropriately for close integration with the hypervisor. Root file systems come preloaded with all the necessary management tools. You can think of the default tuning profile as the result of applying a large tuning change set to the stock Wind River Linux platform project image.

The real-time profile

A profile that further tunes the host kernel for an overall balanced real-time response. When coupled with extra steps to isolate a set of dedicated real time cores, this profile optimizes the delivery of a given external interrupt to a KVM guest. It disables kernel support for subsystems likely to introduce high scheduling variability; in particular, note that it disables the USB subsystem entirely, which prevents you from using any USB-attached devices such as keyboards and mice. In addition, this profile enables common functionality like high-resolution timers and kernel same page merging (KSM).

If you enable this feature, the build system will display a number of warnings. This is expected, and occurs as a result of the kernel configuration fragments changed to ensure balanced real-time from the kernel.

There is a tuning profile available as a template for both host and guest systems. You can add this template to your platform project configuration by using the **setup.sh** tool **--templates feature/host-rt-tune** or **--templates feature/guest-rt-tune** option, depending on the platform project, when configuring your host or guest platform projects. Optionally, you can update the build directory **conf/local.conf** file to include the template, for example:

```
WRTEMPLATE += "feature/host-rt-tune"
```

or

```
WRTEMPLATE += "feature/guest-rt-tune"
```

These profiles can be used as examples for the types of kernel features and tuning required to optimize the system for a particular purpose. You can add your own profiles to address particular use cases of interest.

Detailed information on tuning is available in Tuning the Workload Overview. For a procedure with an example on tuning for specific hardware, see Transferring a Network Card to a VM and Tuning Performance.