A Final Report for the SAP-KVM Project

For internal use only

Matt Fleming
SUSE Performance Team

1. Executive Summary

The SAP-KVM project was started to investigate whether the SUSE KVM stack could be tuned so that the performance of SAP HANA on KVM was at least 88% of bare metal. At the start of the project, it was understood that hitting the target was required for achieving SAP HANA certification for KVM on SLE12-SP2.

Performance was measured using the SAP SOLTP testsuite which provides latency results for a variety of SQL transactions. The target set by SAP was that 80% of the SOLTP tests be no worse than -12% of the bare metal latency when running on KVM.

A cross-functional, virtual team of nine staff was created to carry out the performance investigation. The project was completed after 6 months, and a list of KVM and SLE tunings were documented that proved the performance target was achievable.

This report covers everything that occurred in the project, from the scope of the problem relayed by SAP, to the creation of the virtual team, all the performance investigations, and the final list of recommended tunings. Reflections on the project as a whole, and what SUSE might do differently next time, are presented in Chapter 7. Conclusions.

2. Introduction

SAP HANA is an in-memory relational database which uses network I/O to communicate between client and server. SAP HANA can be characterised as a message-passing workload, which means that latency is the primary measurement of performance.

The SAP-KVM project was a performance investigation, started by Lee Martin and Matt Fleming at SUSE, to understand why the out-of-the-box performance of SAP HANA on KVM was worse than bare metal, and whether KVM could be tuned to improve it.

Bsc#975155 was the report that kick-started the project. In it, SAP demonstrated how poorly their message-passing workloads were running on KVM. In some cases, their microbenchmark, *numaperf*, took 90x longer to run on KVM than on bare metal.

SAP's rationale for improving SAP HANA performance on KVM was that it was required for SAP HANA to be certified on SLES KVM. Given that KVM is a major technology in SUSE's cloud products, the SUSE SAP Alliance team explained that achieving HANA certification from SAP would drive adoption of SUSE's OpenStack product.

Additional motivation was provided in that Red Hat were also working with SAP to certify Red Hat Enterprise Linux (RHEL), and that VMWare had already achieved certification; the project was a race against SUSE's competitors.

SAP provided a testsuite known as the SOLTP testsuite for measuring performance. The SOLTP testsuite contains over 500 tests that measure the latency of various SQL transactions. SAP HANA certification for KVM was conditional on 80% of the tests in the testsuite hitting a target latency of -12% of the bare metal results. In other words, 80% of the tests needed to be no more than 12% worse than bare metal.

From the beginning, the project was only ever intended to be investigatory and not a development project. The goal was to deliver a list of tunings for the host and guest that would improve performance to meet the target set by SAP.

The specification of the project was kept narrow, and included only a single virtual guest running on a host. Additionally, the host was assumed to only be used for running the virtual guest, which meant that the vast majority of the host's hardware resources (processor, memory, etc) could be used by the guest.

Key results of research

- It is possible to tune an out-of-the-box SLE host and guest to be within -12% of bare metal performance for SAP HANA on KVM workloads.
- Providing the virtual guest with accurate CPU and NUMA topology information is critical for achieving top performance.
- The project revealed a number of testing gaps when combining virtualization and performance, and only some have been plugged.
- The SAP-KVM team is proof that SUSE can tackle challenging virtualization/performance issues in the future.

3. Project Narrative

SAP had been attempting to improve the performance of SAP HANA on KVM for about 12 months before SUSE became involved. SAP had initially tried running the SOLTP testsuite on a 6TB 8-socket machine but quickly encountered trouble diagnosing performance issues on such a large and complex machine.

The investigation started for SUSE in earnest when a bug report was opened, "Bug 975155 - SAPonKVM: Micro jobs running slow in kvm virtualized machine". The SAP Alliance, Virtualization, and Performance teams all individually gave input on the bug.

SUSE had tried but failed to reproduce the problem on hardware that was available inside of SUSE data centers, so access to the SAP data center and the SOLTP testsuite was requested.

3.1 SAP Data Center Access

The typical size of machines used by SAP's customers are much larger than most available inside of SUSE; the smallest configuration is a 120-CPU, 4-NUMA node machine with 512GB of memory. Fortunately, two machines had been allocated to the SUSE Alliance team for testing inside of the SAP data center, and these were used during the initial part of the performance investigation.

SAP C-User accounts (with RSA SecurID hardware tokens) are required to remotely login to SAP's desktop terminal and from there to ssh into their machines in the data center. These C-User accounts had to be applied for, the paperwork completed, and the account created. This process took 14 days to complete.

The size of the SAP machines -- and in particular the memory, which has to be initialized every boot -- meant that reboot times were around 30 minutes. This became an obstacle and eventually smaller machines (Intel Xeon 80 CPU, 2-NUMA node with 512GB of memory) within the SUSE performance team were found and used as a stopgap solution. The plan was to return to using the SAP data center once performance had been improved for the smaller (and hence less complex) SUSE machines.

3.2 SAP/SUSE Weekly Calls

Weekly calls were setup between SAP and SUSE to discuss the progress of the investigation, hardware procurement issues, and for SAP to explain the inner workings of SAP HANA. The latter talks led to the discovery of diagnostic SQL information provided by HANA (recorded on the internal SUSE wiki), which showed that mutex contention inside of HANA was causing some of the tests from SOLTP testsuite to experience very high latencies on KVM.

The overview that SAP gave of the SAP HANA thread architecture was crucial for coming up with a debugging plan. The number of threads spawned by SAP HANA is based both on the topology of the machine as well as the latency of work as it moves through the internal work queues of SAP HANA -- additional worker threads may be spawned if jobs placed on the internal work queues take too long to complete.

The complexity of the thread API meant that traditional kernel tracing methods had to be abandoned in favour of SAP HANA's JobEx(ecutor) tracing infrastructure and a Windows-only graphical trace viewer. Using SAP's tools, the issue was eventually tracked down to the virtual guest's vCPU tasks being starved of CPU time.

Being able to demonstrate this issue to SAP using their own tools focussed their attention, and they collaborated with SUSE to investigate the issue.

3.3 SAP-KVM Virtual Team

Up until December 2016 the investigation had largely been driven by Lee Martin on the SAP Alliance team and Matt Fleming from the SUSE performance team. By December, only 40% of the tests in the SOLTP testsuite were achieving the -12% latency goal.

It was clear that the remaining performance issues were too complex, and the project scope too large, for Lee and Matt to solve by themselves; dedicated support from other members of SUSE Labs was sorely needed.

So the SAP-KVM team virtual team was created with the charter of finding tunings for KVM that would meet the KPI given by SAP. The usual virtual team-creation rules were bypassed because SUSE were chasing certification and wanted to beat their competitors in getting there.

Team members were chosen for their domain knowledge of specific areas of the Linux kernel, and since it was unclear which subsystems were responsible for the remaining performance problems, a range of talent was needed. The members were,

1. Lee Martin (SAP Alliance)
2. Matt Fleming (Performance)
3. Mel Gorman (Performance)
4. Mike Latimer (Virtualization)
5. James Fehlig (Virtualization)
6. Alexander Graf (Virtualization)
7. Michal Hocko (Kernel)
8. Vlastimil Babka (Kernel)
9. Joerg Roedel (Kernel)

Weekly SUSE-internal meetings were established and proved to be incredibly useful for discovering gaps in the team's collective knowledge. For example, it turned out that no one knew that enabling 1GB hugepages for the guest required the guest's memory to be a multiple of 1GB or that assigning real-time priority to any KVM I/O threads (IO Threads) requires disabling real-time task throttling on the host.

These blind spots were at the intersection of the individual teams' knowledge. Only by coming together in a virtual team did the gaps become apparent.

A new mailing list, sap-kvm@suse.de, was created so that the team could discuss the performance investigations, record new ideas, and coordinate their activities. Minutes from the weekly meetings were also posted. Due to the size of the team, very few meetings had all members present, and the minutes on the mailing list were invaluable for keeping everyone in sync.

3.4 Changes to the Certification Requirements

In mid-February SAP relaxed the certification requirements for running SAP HANA on KVM because both SUSE and Red Hat were struggling to meet the original KPIs. The new KPIs required 100% of tests pass within -50% of bare metal performance. By this point, the team had anywhere from 7 to 54 tests that were not meeting the new target.

While this was a welcome relaxation of the KPIs, it was also the first inkling that the certification process was not as fixed as the team had initially believed, and that they were not following a well-tested procedure.

3.5 Test Automation and Hitting the KPI Target

The team repeatedly hit issues with different members seeing different SOLTP test results despite the test machine having supposedly identical configurations. These issues were assumed to be down to unintentional differences in the way the host, guest OS, and SAP HANA instances were configured, and the way the SOLTP tests were run.

This was addressed by adding automation to the Performance team's existing infrastructure to deploy KVM guests, install SAP HANA, run the SOLTP testsuite, and generate results. This was a substantial amount of work from Mel Gorman, but once it was in place it became straightforward to try out a list of tuning options to see which improved performance.

While other teams within SUSE Labs, such as the Virtualization team, already had some method of automatically creating and configuring virtual guests, that code was not easily integrated into the Performance team's existing code base and it didn't provide all the required functionality.

The following new functions were added to the Marvin and mmtests projects which power the Performance team's testing infrastructure,

- Tune the of the KVM I/O writeback policy
- Offline any guest CPUs that have IO Threads pinned on the host
- Use separate partition for running tests within KVM guest if available
- Allow extra boot parameters to be specified for KVM guest
- Use host topology info to optionally remove one core per NUMA node from guest XML configuration
- Pass `x2apic` option to Qemu `-cpu` parameter if available
- Add option to pin IO Threads to CPUs based on NUMA topology

Since everything was deployed automatically, it was possible to "lock-in" known good tunings so that they weren't lost with further changes. Things quickly progressed from this point and the team achieved the original KPIs within a couple of weeks.

Once the complete list of tunings were applied to the machines inside of the SAP data center, and the KPI target confirmed, the process of disbanding the SAP-KVM team began.

3.6 Additional Help for the BW4 Benchmark

Though it was outside of the original scope, the team assisted the SAP Alliance group with configuring and running the SAP BW4 benchmark because it was understood that this was a required step for achieving certification of SAP HANA on SLES KVM.

This assistance included,

- Finding a suitable machine with 1TB+ of RAM
- Connecting fast storage to the machine

Results for the BW4 benchmark were always required by SAP for certification, but no hardware inside of SUSE had been allocated prior to the start of the project. When SUSE learned that Red Hat would likely be announcing that they passed certification at the SAP Sapphire conference in May, the priority of SUSE getting BW4 results escalated dramatically.

A Fujitsu PRIMEQUEST 2800E2 machine was offered by SUSE L3 QA for running the BW4 benchmark after the request for hardware was escalated up the management chain. Because the L3 division were also using it to conduct their testing for the upcoming SLE12 SP3 release, its use had to be coordinated.

When access was eventually granted, additional time was spent finding suitably fast storage for the L3 machine because none was installed when the machine was purchased. A NetApp storage array was located within SUSE Labs and attached for running the BW4 benchmark.

3.7 Disbanding the Team

After tunings were discovered to reach the SOLTP KPIs, the Fujitsu machine had been procured, and most of the project's bugzilla entries had been resolved, the original objectives (and some additional ones) had been achieved. The SAP-KVM team began disbanding.

Because of the team's success, this was met with some resistance. Stakeholders encouraged the team to expand the scope of the initial investigation and continue helping to run bigger benchmarks on larger machines. But there was no reason to believe that any future work items would use the unique skills of the SUSE Labs team members, namely kernel and virtualization analysis and tuning. Instead, it was mainly a job of running benchmarks far more complicated than the SOTLP testsuite.

With the ever-increasing backlog of work items for the pending SLE12 SP3 release, it was eventually agreed that the SAP-KVM team should disband. A few SUSE engineers agreed to

continue working on some items such as the SAP-KVM SAP Note which outlines the tunings the team put together to achieve the KPIs, and with updating the SLE Virtualization documentation to incorporate the discoveries made throughout the project.

4. Performance Investigations

This chapter covers the performance investigations that led to the recommended list of tunings in Chapter 5 and Chapter 6. Every investigation began with performance analysis.

Some of the setbacks in this investigation can be attributed to Amdahl's Law. Tunings that improved performance at the start of the project, later turned out to hurt it as more and more bottlenecks were removed.

4.1 Reschedule IPIs

The Linux kernel scheduler has two methods of waking up a task on a CPU when it has been queued on the run-queue. The simplest, and optimal way is for the waking CPU (the one that enqueued the task) to preempt the currently running task directly. Since this involves accessing the remote CPU's data structures, it can be prohibitively expensive if waking up tasks on CPUs that do not share any caches, such as CPUs on different NUMA nodes that do not share a last-level cache (LLC).

There is a second method which is better for the case of waking a task when the CPUs do not share an LLC: interrupt the remote CPU with an inter-processor interrupt (IPI) and ask the remote CPU itself to handle preempting the currently running task. This is known as a *reschedule IPI*.

Looking at the number of reschedule IPIs during the investigation of bsc#975155 showed a 1000x difference between host and guest. This was because the host topology was not correctly exposed to the guest; the guest thought it had 48 CPUs, with 1 CPU per socket, but the host had 48 CPUS, with 24 CPUs per socket. CPUs that did in fact share an LLC on the host were not described that way on the guest.

This meant that the guest never used the quicker, optimised wakeup method.

Apart from the fact that the reschedule IPI method is the slowest of the two methods, emulated IPIs on the guest take roughly twice as long to handle as on bare metal. Hence, any reduction in reschedule IPIs improves performance.

Alexander Graf provided a more detailed analysis:

“IPIs are basically the worst case for VMs. You need to trap an exit, emulate the LAPIC event, fire a host IPI to wake up the other thread (+schedule) or kick it out of the VM as well (#vmexit) and then inject that IPI event into the other CPU.”

Updating the guest’s libvirt XML configuration file with an accurate `topology` section, and binding the vCPU tasks to host CPUs, presented the correct topology information to the guest. But the guest still did not contain *any* LLC details, and was still issuing 1000x reschedule IPIs when compared to the host.

Alexander Graf found an upstream Qemu patch, written by an engineer at Huawei, that exposed a virtual LLC (with a static size) to the guest (bsc#1007769). The size of the LLC is less important than the topology information that is expressed, i.e. which CPUs are connected to the LLC, and hence which CPUs can use the quick wakeup method.

A second source of reschedule IPIs was that the VM required an IPI to be sent whenever the vCPU task went to sleep. Using the `idle=poll` kernel boot parameter forces the vCPU task to poll for more work instead of sleeping and requiring an IPI. This option reduced latency at the expense of increasing the power consumption. Fortunately, SAP were happy to make that trade-off.

The `idle=poll` option isn’t necessary for bare metal because it uses the `MWAIT` instruction which consumes minimal power, and doesn’t require an IPI to be woken up. This instruction isn’t exposed via KVM and so isn’t available to guests.

4.2 Issues Discovered with idle=poll

While using the `idle=poll` kernel boot parameter helped to improve performance at the start of the project, as more tunings were applied and more bottlenecks in the KVM stack removed, it was found to actually hurt performance.

The problem is that when it’s enabled, vCPU tasks always appear to be busy and never voluntarily give up the host’s CPU. This leads to latency issues if the host CPUs have hyperthreading (HT) enabled.

HT technology relies on idle cycles in the CPU pipeline to schedule siblings -- since HT siblings share some CPU pipeline resources, they can’t always execute in parallel. The problem with tasks that are constantly busy, such as vCPU tasks when running with `idle=poll` on the guest, is that there are rarely ever any idle cycles. It becomes much more difficult for both HT siblings to get equal share of the CPU resources. And this translates to observed latency on the guest.

The solution was to use the `intel_idle.max_cstate=1` kernel boot parameter instead, which reduced the latency seen on the guest.

The existing SLE SAP-HANA profile for the `tuned(8)` package was recommended by the SAP installation guides, but it also turned on the equivalent of `idle=poll` by writing a 0 to `/dev/cpu_dma_latency`. Modifying the `force_latency=1` parameter of the profile was necessary for `intel_idle.max_cstate=1` to take effect.

4.3 Reducing KVM MMU Lock Contention with Hugepages

While investigating SOLTP testsuite latency, the KVM MMU page table code in the kernel was showing significant lock contention for one of the locks taken while servicing page faults caused by the guest. The `perf record` command was used to discover the lock contention and its cause.

Joerg Roedel suggested enabling hugepages on the guest libvirt XML configuration, which would decrease the number of page faults (since each fault maps 1GB instead of 4KB), and should therefore reduce lock contention.

During a phone call with Red Hat where tuning ideas were discussed, they confirmed that enabling 1GB hugepages had resulted in a 10% improvement across the board.

But enabling hugepages wasn't straightforward. The libvirt deamon on SLE doesn't mount the 1GB hugetlbfs file system by default, which means that Qemu cannot use 1GB for guests without manual intervention.

Further problems were hit when libvirt failed to automatically round the size of the guest's memory to be a multiple of 1GB. Non-aligned memory sizes caused libvirt to exit with a cryptic error message. This was a tracked to a bug in the upstream project, and Jim Fehlig worked with the libvirt community get it resolved.

There was also a timeout issue when booting VMs with lots of hugepage-backed memory. Again, this required Jim to discuss the fix with the upstream community and backport the patch into the SLE libvirt packages (bsc#1013113).

When things were finally working, 1GB hugepages increased the number of tests hitting the target KPI by 20 percentage points.

4.4 Reducing Scheduler Run-queue Contention with kvm_intel.ple_gap

The `kvm_intel` kernel module has a parameter that controls how long a vCPU task spins executing a guest `HLT` instruction (to wait for more work) before going to sleep. While no noticeable performance improvement was observed by disabling this feature (causing the vCPU task to spin forever), enabling it does avoid the vCPU task taking the host scheduler's run-queue locks, which can be highly contended for message-passing workloads.

The Pause-Loop-Exit (PLE) code assumes that by giving up the current host CPU using the `sched_yield(2)` system call, and causing the current vCPU task to sleep, some other vCPU task will be able to execute on the current host CPU.

But since all vCPU tasks were pinned to unique host CPUs (as discussed in Host CPU Reservation), yielding the host CPU to another vCPU task is useless and just causes unnecessary run-queue lock contention.

4.5 Improved I/O Latency with Qemu IO Threads

Qemu provides a couple of methods for performing guest I/O. The default for Linux is *native*, which uses the Linux asynchronous I/O system calls, `aio(7)`. Being asynchronous, the I/O code must signal when the I/O completes by interrupting the vCPU task. This causes noticeable latency issues on the guest.

Switching to the *iothreads* option improved performance of the SOLTP testsuite on KVM. Since the *iothreads* option creates a separate task purely for handling I/O, the context-switching required for the asynchronous I/O method is avoided. Plus, when the IO Thread is pinned to a dedicated CPU (discussed next), I/O handling no longer needs to preempt any of the vCPU tasks.

4.6 Host CPU Reservation

After the vCPU tasks were pinned to host CPUs the team noticed that the vCPU tasks would occasionally experience extreme scheduling latency. The `perf sched` tool was used to record scheduler latency and the vCPU tasks reported intermittent, but extreme, latencies.

The scheduler latency was observed on the guest as SAP HANA tasks holding internal database mutexes for very long times (see SAP/SUSE Weekly Calls), which lead to increased SOLTP latencies.

Assuming a 144 CPU host, reducing the size of the VM from 144 vCPUs to 136 vCPUs (one host CPU core free per socket) and pinning Qemu IO Threads to the free cores helped to reduce the SOLTP latencies by ensuring that no tasks on the host were being starved of CPU time.

Changing the number of CPUs on the guest did require results to be regenerated for the bare-metal baseline, so that a fair comparison of results was maintained, e.g. 136 vCPU guest against a 136 CPU host.

This highlights a fundamental limitation of running latency-sensitive workloads on KVM: **it is not possible to give a guest access to all host CPUs without causing scheduler latency.**

4.7 irqbalance Causes Latency

When searching for sources of latency on the guest, the kernel function tracer (ftrace) showed that the `/proc/interrupts` file was being held open for reading, sometimes for hundreds of milliseconds. The cause was the irqbalance daemon which reads this file to understand which devices are generating interrupts. irqbalance uses that information to update interrupt affinities and ensure that the system interrupt load is spread evenly across all CPUs.

Reading the `/proc/interrupts` file acquires a spin-lock inside the kernel, preventing the irqbalance task from being preempted, which directly caused latency on the guest when running the SOLTP testsuite. Since irqbalance was always running on the host too it caused scheduler latency for the KVM vCPU tasks, which indirectly led to increased latency on the guest.

The solution was to disable the irqbalance daemon on both the guest and host. Of course, disabling the daemon is not without issue: it means that interrupts can no longer be routed dynamically to CPUs based on the load of the system. Instead, careful manual pinning of interrupts is necessary if any system devices are likely to have a high interrupt rate.

4.8 Enabling MWAIT/MONITOR Idle Wakeup in the Guest

Once the performance impact of having to wake idle vCPU tasks was analysed (see Reschedule IPIs), a set of patches for KVM and Qemu were written by Alexander Graf to expose the `MWAIT` and `MONITOR` instructions to the guest.

The patches, when combined with the other tunings listed, surpassed the KPI target and 97% of the SOLTP tests were within -12% of the bare metal latency.

In the end, these patches were not required to achieve the KPI target and were not applied for SLE12-SP2. However, since they provide a decent performance improvement, they were applied for SLE12-SP3.

5. Host Tunings

Below are the recommended host OS tunings for running SAP HANA on KVM. Since the rationales for some of the tunings were discussed in the previous chapter, they are not repeated here or in the following chapter on Guest Tunings. Instead, references to the previous chapter are included for each tuning where appropriate.

It's assumed that the reader has an understanding of how to install and configure a SUSE Linux Enterprise release and a virtual machine.

5.1 kvm_intel.ple_gap=0

The *kvm_intel* kernel module *ple_gap* parameter should be set to 0 either by adding

```
kvm_intel.ple_gap=0
```

to the kernel command-line, or by using `modprobe(8)` to specify the parameter value at module-load time,

```
modprobe kvm_intel ple_gap=0
```

See 4.4 - Reducing Scheduler Run-queue Contention with kvm_intel.ple_gap

5.2 CPU frequency governor and performance bias

The CPU frequency governor should be set to *performance* and the performance bias value set to 0 (maximum performance),

```
cpupower-frequency-set -g performance
cpupower-set -b 0
```

5.3 Tuned Profile

The *virtual-host* or *sap-netweaver* `tuned(8)` profiles should be installed and enabled.

```
zypper in tuned
systemctl enable tuned
systemctl start tuned
tuned-adm profile virtual-host
```

See 4.2 - Issues Discovered with idle=poll

5.4 Disable irqbalance Daemon

The irqbalance service, which is enabled by default, should be disabled on the host because it can cause latency issues on the guest.

```
systemctl stop irqbalance.service
systemctl disable irqbalance.service
```

See 4.7 - irqbalance Causes Latency

5.5 Hugepages

1GB hugepages should be used on the host because they reduce the cost of mapping memory for use by the guest. Hugepages should be allocated on the kernel command-line using the `hugepages=` option. Enough hugepages need to be allocated to cover the amount of memory specified for the guest, e.g. for a guest with 128GB of memory, hugepages can be allocated with,

```
hugepages=128 hugepagesz=1GB default_hugepagesz=1GB
```

See 4.3 - Reducing KVM MMU Lock Contention with Hugepages

5.6 Automatic NUMA Balancing

Enabling automatic NUMA balancing on the host can cause latency on the guest. It should be disabled, either by adding the following parameter to the kernel command-line,

```
numa_balancing=disable
```

Or by writing 0 to the `numa_balancing` file in procfs,

```
echo 0 > /proc/sys/kernel/numa_balancing
```

5.7 Transparent Hugepages

Because 1G pages are used for the virtual machine, there is no additional benefit from having THP enabled. Disabling it will avoid `khugepaged` kernel task interfering with the virtual machine while it scans for pages to promote to hugepages. Transparent hugepages can be disabled on the host by adding the following kernel command-line option,

```
transparent_hugepage=never
```

Or by writing to the following sysfs file,

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

See 4.3 - Reducing KVM MMU Lock Contention with Hugepages

5.8 I/O Scheduler

The deadline I/O scheduler should be used for the host, which can be done for all block devices by specifying the following kernel command-line option,

```
elevator=deadline
```

Alternatively, the I/O scheduler can configured at runtime by writing to the sysfs `scheduler` file for a specific block device, such as `/dev/sda`,

```
echo deadline > /sys/block/sda/queue/scheduler
```

5.9 intel_idle.max_cstate=1

To ensure that the host processor does not try to conserve power at the expense of performance, the maximum processor C-state should be specified on the kernel command-line,

```
intel_idle.max_cstate=1 processor.max_cstate=1
```

See 4.1 - Reschedule IPIs

6. Guest Tunings

It is assumed that the guest was created using the `virt-install` command as described in the official SUSE Virtualization documentation. The XML description file for the guest should be edited with the `virsh edit` command.

6.1 Virtual Machine QEMU command-line options

There are a few options required in the QEMU command-line XML element, "`host,migratable=off,+invtsc,l3-cache=on`". Using these options requires changing the domain XML element that is present in the default virtual machine description.

Here's the relevant portions of XML for enabling the `-cpu` parameters.

```
<domain type='kvm'
xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
...
  <cpu mode='host-passthrough'>
  ...
```

```
  </cpu>
  ...
  <qemu:commandline>
     <qemu:arg value='-cpu'/>
     <qemu:arg value='host,migratable=off,+invtsc,l3-cache=on'/>
  </qemu:commandline>
</domain>
```

6.2 Qemu IO Threads

The default guest configuration created by `virt-install` uses the Linux native I/O driver. Switching to the “threads” virtio driver improves performance for the SAP HANA workload. IO Threads should also be pinned to physical CPUs that are not used by the guest, and a unique IO Thread should be assigned for each block device.

```
<domain type=...>
  ...
  <iothreads>1</iothreads>
  ...
  <cputune>
    ...
    <iothreadpin iothread=’1’ cpuset=’0’/>
  </cputune>
  ...
  <devices>
    ...
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='threads'
iothread='1'/>
      <source dev='<source device path>'/>
      <target dev='vda' bus='virtio'/>
   </disk>
  </devices>
</domain>
```

See 4.5 - Improved I/O Latency with Qemu IO Threads

6.3 Matching Host and Guest Topology

The guest’s CPU and NUMA topology should mirror the host’s so that all platform information used by the guest’s kernel is accurate; **over-allocating physical resources will lead to poor performance.**

It’s important to note that QEMU hyperthread sibling pairs are always consecutive CPUs, starting from CPU 0, e.g. CPU 0 and CPU 1 are the first hyperthread siblings, followed by CPU 2 and CPU 3, etc. This is unlikely to be the same enumeration scheme used by the host.

One physical processor core should be unused by the guest for each NUMA node. This allows the host to schedule tasks on the unused core, along with any IO Threads used by the guest’s block devices.

Here’s an example guest XML topology snippet for a 2-NUMA node, 10-core guest with 1 block device. The host configuration is:

- 2-NUMA nodes
- 12 cores
- Hyperthreading enabled
- The first core of each NUMA node unused by the guest

```
<domain type=...>
  <vcpu placement=’static’>20</vcpu>
  <iothreads>1</iothreads>
  <cputune>
    <vcpupin vcpu='0' cpuset='1'/>
    <vcpupin vcpu='1' cpuset='12/>
    <vcpupin vcpu='2' cpuset='2'/>
    <vcpupin vcpu='3' cpuset='13'/>
    <vcpupin vcpu='4' cpuset='3'/>
    <vcpupin vcpu='5' cpuset='14'/>
    <vcpupin vcpu='6' cpuset='4’/>
    <vcpupin vcpu='7' cpuset='15'/>
    <vcpupin vcpu='8' cpuset='5’/>
    <vcpupin vcpu='9' cpuset='16'/>
    <vcpupin vcpu='10' cpuset='7'/>
    <vcpupin vcpu='11' cpuset='18'/>
    <vcpupin vcpu='12' cpuset='8'/>
    <vcpupin vcpu='13' cpuset='19'/>
    <vcpupin vcpu='14' cpuset='9'/>
    <vcpupin vcpu='15' cpuset='20'/>
    <vcpupin vcpu='16' cpuset='10'/>
    <vcpupin vcpu='17' cpuset='21'/>
    <vcpupin vcpu='18' cpuset='11'/>
    <vcpupin vcpu='19' cpuset='22'/>

    <iothreadpin iothread='1' cpuset='0'/>
  </cputune>
  <numatune>
    <memory mode='strict' nodeset='0-1'/>
    <memnode cellid='0' mode='strict' nodeset='0'/>
```

```
    <memnode cellid='1' mode='strict' nodeset='1'/>
  </numatune>
  …
  <cpu mode='host-passthrough'>
    <topology sockets='2' cores='5' threads='2'/>
    <numa>
      <cell id='0' cpus='0-9' memory='<Memory per NUMA node>'
unit='KiB'/>
      <cell id='1' cpus='10-19' memory='<Memory per NUMA node>'
unit='KiB'/>
    </numa>
  </cpu>
</domain>
```

See 4.6 - Host CPU Reservation

6.5 Hugepages Memory Backing

Hugepages should be enabled in the `memoryBacking` XML element and the guest's memory size must be a multiple of 1GB.

```
<domain type=...>
  ...
  <memoryBacking>
    <hugepages>
      <page size='1048576' unit='KiB'/>
    </hugepages>
    <nosharepages/>
  </memoryBacking>
</domain>
```

See 4.3 - Reducing KVM MMU Lock Contention with Hugepages

6.5 virtio Random Number Generator

The host's Random Number Generator (RNG) device should be exposed to the guest with the `rng` XML element.

```
<domain type=...>
  <devices>
    ...
    <rng model='virtio'>
      <backend model='random'>/dev/random</backend>
```

```
      <alias name='rng0'/>
    </rng>
  </devices>
</domain>
```

6.6 Disable irqbalance Daemon

The irqbalance daemon can cause latency issues when running SAP HANA and should be disabled the same way as on the host (5.4 - Disable irqbalance Daemon).

```
systemctl stop irqbalance.service
systemctl disable irqbalance.service
```

See 4.7 - irqbalance Causes Latency

7. Conclusions

Overall, the SAP-KVM project was a success. The performance target for the SOLTP testsuite put forward by SAP was achieved. The project was extremely experimental in nature, and much larger in scope than originally anticipated -- what started as an investigation with two people ballooned into a team of nine.

However, changes were made to the project as the problem became clearer. Ultimately it was the team's adaptability that led to the project's success, and there were a number of tactical decisions made throughout that resulted in huge steps forward.

When the initial investigation involving bsc#975155 proved to be too difficult for two team members to handle, the SAP-KVM virtual team was created with domain experts from the SUSE Labs division. The virtual team structure worked extremely well, and every single member contributed to the project in some way. Having experts from different parts of SUSE on a single team also made it clear where gaps existed in the team's collective knowledge.

The nature of the project means SUSE Labs is well positioned to tackle virtualization-performance issues in the future, such as for OpenStack.

Much of the project work covered new ground, and the team hit multiple configuration issues when running a virtualized workload on machines of comparable size to SAP's customers. Enabling 1GB hugepages (see 4.3 - Reducing KVM MMU Lock Contention with Hugepages) triggered both user interface bugs with rounding of guest memory sizes (bsc#1029738) and a timeout issue (bsc#1013113) when the guest had hundreds of gigabytes of memory. All of these were analysed, fixed, and patched in the libvirt package before any known customers hit them.

The new automation added to the Performance team's infrastructure was a turning point for the project. It proved to be critical for speeding up the rate of testing suggested tunings. Any future virtualization-performance work should leverage this investment.

There are also lessons to be learned from the things that did not go so well. Turnaround times for new hardware continue to be a problem where the request wasn't planned in advance. One engineer had to wait over 2 weeks for a hard disk to be delivered so he could run tests locally on his machine. There was also the problem with acquiring an SAP-scale machine inside of the SUSE data center for running the BW4 benchmark. A potential solution to these problems is better forecasting of required hardware, for example, compiling a list of necessary hardware at the start of the project.

Once the KPI target was achieved, the suggestion to disband the team was met with resistance. Some stakeholders wanted to keep the momentum of the SAP-KVM team going. This was despite the fact that the tasks that made the best use of the team's skills had already been completed, and that the team members were required on other projects. Any future teams should be aware of these potential pressures.

While ultimately the certification of SAP HANA on SLES KVM was not achieved due to factors outside of the team's control, the SAP-KVM project was a huge success. It proved that SUSE is capable of staffing a task force with domain experts to achieve an ambitious target with uncertain scope. The virtual team concept and lessons learned throughout this project provide a useful model that can be reused for any future projects that occur at the intersection of SUSE's technologies.

Appendix A - Terminology

IPI - Inter-processor Interrupt
LLC - Last-level cache
MMU - Memory management unit
NUMA - Non-uniform memory architecture
vCPU -- Virtual CPU on the guest

Appendix B - SAP-KVM Bugzilla Entries

bsc#975155 - SAPonKVM: Micro jobs running slow in kvm virtualized machine

bsc#1006795 - SAPonKVM: tuned attempts to set cpugovernor in VM (KVM)

bsc#1007769 - SAPonKVM: Upstream Qemu virtual L3 patch to improve performance of message-passing workloads

bsc#1017017 - SAPonKVM: Improve qemu's L3 cache support

bsc#1031142 - SAPonKVM: SAP HANA Performance: MONITOR/MWAIT handling missing

bsc#1016687 - SAPonKVM: Enabling 2MB hugetlb pages on host causes severe performance regression for HANA

bsc#1011640 - SAPonKVM: hyperthread siblings not correctly mapped into VM

bsc#1011899 - SAPonKVM: Linux scheduler very low sched_migrate_task in Guest vs. physical

bsc#1029738 - Ensure hugepage memory settings are properly aligned

bsc#1013113 - SAPonKVM: Add support for changing timeout value to open unix monitor socket / hugepages limited

bsc#1006997 - SAPonKVM: how to enable invtsc (and thus nonstop_tsc and constat_tsc) in libvirt