Explanations for Xen-aware yielding patch

1 Preliminary

The patch is used to improve communication performance in VCPU overcommitted system with Xen. This document is used to explain the reasons behind the performance penalty on such systems, as well as how it improve mpich2-1.2.1 running on top of Xen.

Note: The patch applies only for the PV guests of Xen with overcommitted configuration, i.e., the total number of VCPUs is greater than the number of PCPUs (or cores) of the system.

2 Test-bed

A COTS PC server is used as the testbed. It has an Intel Core 2 Duo E6550 processor, which has two processing cores, running at 2.33GHz with 128KBytes L1 cache, and 4MBytes shared L2 cache. Moreover, it is configured with 2GBytes DDR2 memory, and 160GB SATA hard disk drive. For virtualization, we use Xen with version number 3.4.2. The guest domains (including dom0 and domUs) are installed Redhat Enterprise Linux x86_64 Operating System with version number 5.1. The Xen-Linux with kernel version 2.6.18.8 is used to boot the all the guest systems including dom0. The dom0 is the only privileged domain, which contains all drivers of the physical devices, and is configured with 512 MBytes memory. The domUs used in the experiments are Para-virtualized (i.e., PV for short) guests. Each of the domUs is configured with 256 MBytes memory, 4 GBytes virtual hard disk drive, and a virtual network interface card to communicate with world outside (via the bridge network installed in dom0). The number of VCPUs of the domUs can be easily varied by revising the configuration file.

3 Before patching

3.1 Communication Performance

The benchmark used to collect MPI communication performance data is b_eff (URL: https://fs.hlrs.de/projects/par/mpi//b_eff/).

We boot the testbed with a 2.6.28-rc2 Linux kernel that does not have SMP support to form a uni-processor (UP for short) environment, and obtain the MPI communication performance (we name it as **native_up**) between two benchmark processes. Since the UP environment is comparable with the scenario

that overcommits multiple VCPUs to one physical processor, we use native_up performance as a reference in all following performance evaluations.

By using MPI primitives, the processes running inside a VM may communicate via two possible channels: the loopback network of dom0 and shared memory. We measure the communication performance in both cases (i.e., inter-vm for the former and intra-vm for the latter).

There are two schedulers available in Xen hypervisor: Credit and SEDF. For all following experiments in this paper, we use the default scheduling parameters of both schedulers, i.e., for Credit scheduler, all domains has the same weight number of 256, and cap value 0, while for SEDF scheduler, the scheduling parameter of dom0 is (15ms, 20ms, 1) and that of domUs is (0ms, 100ms, 1). When testing the communication performance of overcommitted VCPUs, we consider the following four configurations:

- **inter_vm_credit_pin** The virtualized system uses Credit scheduler and is booted with two processing cores. Two domUs are created, each of which has only one VCPU, and all of their VCPUs are pinned to the physical core one. The MPI communication performance between these two domUs is measured.
- **inter_vm_sedf_pin** The same as inter_vm_credit_pin, except that Xen hypervisor uses SEDF scheduler.
- **intra_vm_credit_pin** The virtualized system uses Credit scheduler and is booted with two processing cores. One domU is created with two VCPUs, and all of its VCPUs are pinned to the physical core one. The MPI communication performance inside the domU is measured.
- **intra_vm_sedf_pin** The same as intra_vm_credit_pin, except that Xen hypervisor uses SEDF scheduler.

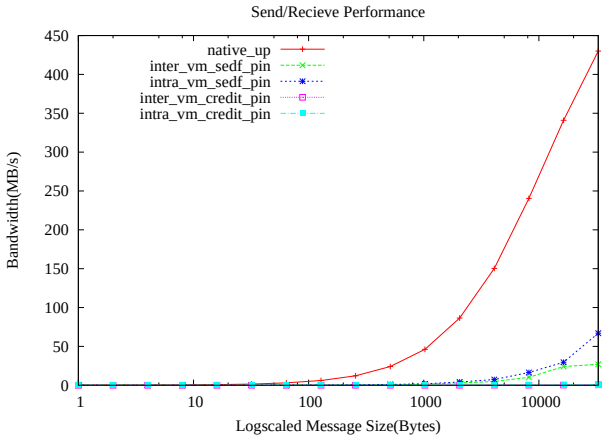

Figure 1: MPI communication performance in overcommitted VCPUs

From above figure, it can be observed that compared with the native_up case, the MPI communication performance of the overcommitted VCPUs drops

dramatically. For all four overcommitted cases, the performance on SEDF scheduler is always (in both inter-vm and intra-vm cases) better than that obtained on Credit scheduler. This is because the SEDF scheduler defines smaller time slices than Credit scheduler, and the frequent scheduling helps reducing the performance lose resulted by busy-polling mechanism, employed by MPI communication library.

3.2 Execution Time of NPB Programs

To further address the performance problem in overcommitted VCPUs, three benchmark programs are employed, i.e., is.A.2 (integer sorting, which relies on all-to-all communication to exchange intermediate results) lu.A.2 (solving five coupled parabolic/elliptic partial differential equations, which needs to communicate to exchange data during computation) and ep.A.2 (random-number generator, which is "embarrassingly" parallel in that no communication is required for the generation of the random numbers itself), from NPB 3.3.

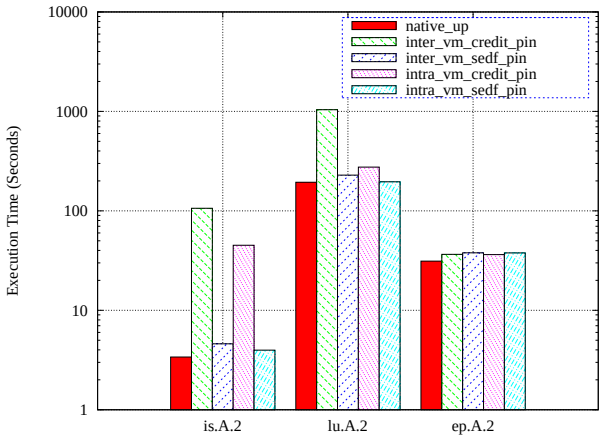

Figure 2: Impacts on applications of VCPU overcommitting

A close look on the source code of mpich2-1.2.1 reveals the fact that the performance penalty is resulted by the busy-polling mechanism employed by MPI library. Since the yielding (e.g, sched_yield in guest OS) after polling for regulated times (i.e., 1000 times defined by the library) simply does not work to yield the VCPU hosting the application to be re-scheduled at hypervisor layer.

3.3 Why need a patch?

There are two main reasons to improve MPI communication performance in the overcommitted VCPUs:

1. With current Xen architecture, the scheduling decisions are made locally. Each PCPU of a multi-processor system maintains its VCPU run-queue and schedules them to run without coordinations with other PCPUs of the system. Therefore, in overcommitted systems (i.e., the number of VCPUs is greater than the number of PCPUs), it is very possible that two VCPUs

that hosts processes communicating with each other via MPI primitives are not scheduled simultaneously. And this result in the same scenario as that discussed in the preceding experiments, although the VCPUs may reside in the run-queues of different PCPUs.

2. With the load-balancing strategy of Credit scheduler, which is the current default scheduler for Xen, PCPUs will "steals" VCPU from the run-queue of its neighboring PCPUs to execute when it found an empty local run-queue. And this scheme may make the VCPUs communicating with each other to run on the same PCPU concurrently, which forms the same scenario as that discussed in preceding experiments.

Therefore, improving the MPI communication performance in overcommitted VCPUs will help improving the overall performance of overcommitted systems with HPC workloads.

4 After patching

4.1 Communication Performance

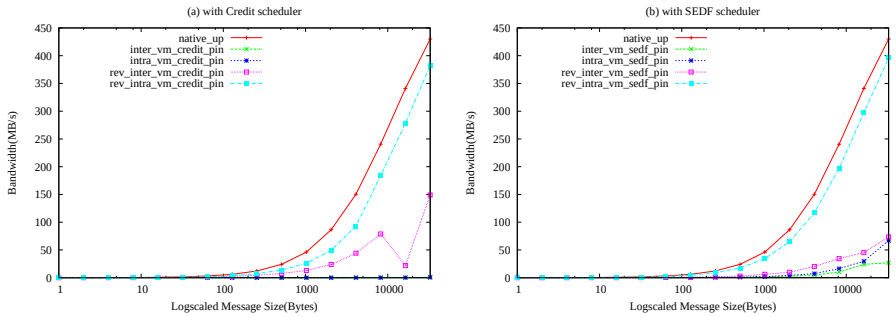

Figure 3: MPI communication performance in overcommitted VCPUs after improvement

Within above figure, the performance data marked with "**rev**_" prefix means that of b_eff linked with mpich2-1.2.1 that is revised by employing the patch. The communication performance is improved greatly: For Credit scheduler, about 300 times higher communication performance is achieved in inter-vm case, and 700 times higher for intra-vm case, while for SEDF scheduler, about 3 times higher for inter-vm case and about 4 times higher for intra-vm case.

Moreover, for intra-vm cases, the communication performance with revised MPI library becomes rather close to that of native UP case. This is because for intra-vm cases, most of the communications are conducted via shared memory, which is rather similar as the inter-process communication in native case. The difference is that the context switch happens between VCPUs is more expensive than the processes. For inter-vm cases, the communication data exchanged between domUs needs to be relayed by the loopback network of dom0, and this results worse performance than the intra-vm cases although it is still much higher than that of original MPI library implementation. Moreover, with the

load-balancing mechanism of Credit scheduler, the VCPUs of dom0 will automatically migrates to the "spared" core, and thus have enough processing resources to relay the messages, which results better communication performance than SEDF scheduler in inter-vm cases.

4.2 Execution Time of NPB Programs

4.2.1 On Defined Overcommitted Scenarios

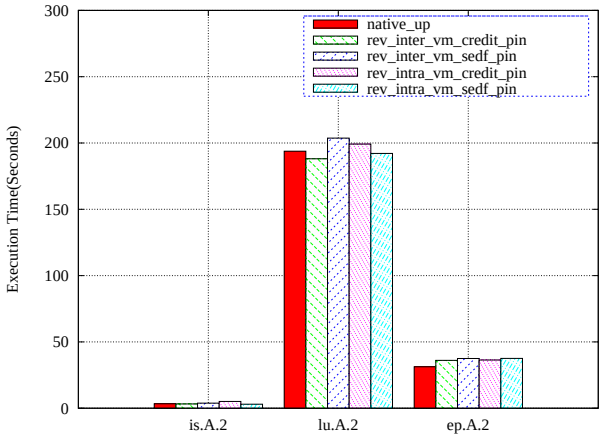

Figure 4: Performance of NPB programs after improvement

From above figure, it can be observed that after patching, the execution time of is.A.2 and lu.A.2 is rather close to that of running inside native UP environment.

4.2.2 On Overcommitted Systems

We use the domUs with two VCPUs to construct the overcommitted systems. By increasing the number of the domUs from one to four, we gradually increase the *overcommitting pressure* on the testbed. In each case, identical benchmark programs, i.e., is.A.2 or lu.A.2 from NPB, are invoked simultaneously to run in these domUs, and their execution time is recorded. Table 1 demonstrates the average (each average number is computed by three samples) execution time of these benchmark programs on Credit scheduler, and Table 2 demonstrates that on SEDF scheduler.

From Table1, it can be observed that when the overcommitting pressure is low (i.e., for the 1 domU and 2 domUs cases), the execution time of benchmark programs with revised MPI library is rather close to that with original MPI library. The performance of is.A.2 is even a little worse than the original one. This is because giving small number of VCPUs, the possibility that the VCPUs of the same domUs are not scheduled to run simultaneously is low, and this results in less wasting of CPU cycles as that in the intra_vm_credit_pin configuration. Moreover, more frequent VCPU context switches will resulted by our proposal, since there are VCPUs from dom0 even when there is only one domU

Table 1: Average execution time of is.A.2 and lu.A.2 with different overcommitting pressure on Credit scheduler

	is.A.2		lu.A.2	
	original (seconds)	revised (seconds)	original (seconds)	revised (seconds)
1 domU	1.841	1.852	105.047	104.658
2 domUs	4.711	5.220	265.270	248.948
3 domUs	9.555	5.986	389.371	344.684
4 domUs	13.980	7.906	522.217	448.37

hosting is.A.2 running on the testbed, which results in a little worse performance than that of is.A.2 with original version MPI library.

However, when the overcommitting pressure turns higher (i.e., for the 3 domUs and 4 domUs cases), the average execution time of the benchmark programs with revised MPI libraries are significantly shorter than that with the original MPI libraries. This is because with the more VCPUs running, the possibility of VCPUs of the same VM being not scheduled simultaneously is significantly higher than that with less VCPUs. And this inevitably results in the same kind of performance losing experienced in the intra_vm_credit_pin configuration for the programs employing original MPI library.

Table 2: Average execution time of is.A.2 and lu.A.2 with different overcommitting pressure on SEDF scheduler

	is.A.2		lu.A.2	
	original (seconds)	revised (seconds)	original (seconds)	revised (seconds)
1 domU	1.843	1.825	103.943	103.096
2 domUs	3.571	3.470	236.318	235.046
3 domUs	5.252	5.147	354.547	355.677
4 domUs	7.206	6.739	481.759	474.352

From Table 2, it can be observed that on SEDF scheduler, the performance improvements that can be achieved by replacing the MPI library is smaller than that on Credit scheduler. This is because SEDF schedules the VCPUs with smaller time slices than the Credit scheduler, which naturally results in less performance losing due to the busy-polling mechanism employed by MPI library. Moreover, the SEDF scheduler does not supports load-balancing by automatically migrating the VCPUs to idle PCPUs, which results less performance losing experienced in intra_vm_sedf_pin configuration. Compared the data in Table 2 and Table 1, it can be observed that the average execution time of lu.A.2 is substantially increased on SEDF. This is also because the load-balancing mechanism employed by Credit scheduler improved the throughput of the system.

Lastly, we employ a domU that is configured with one VCPU and hosting a dead-loop program as the *noise* workload to simulates the case that VMs hosting HPC workload compete resources with other VMs. The noise VM does nothing except honestly burning out each 30ms time slice that is allocated by

the Credit scheduler. In Table 3, we demonstrate the average execution time of is.A.2 running inside different number of domUs that run along with the noise VM on Credit scheduler. The table also records the averaged standard deviation of execution time that is obtained from the execution time of is.A.2 running inside the co-existing domUs in each case with different overcommitting pressure.

Table 3: Average execution time and standard deviation of is.A.2 with different overcommitting pressure and noise on Credit scheduler

	original		revised	
	Avg. Exec. Time (seconds)	Avg. Std. Deviation	Avg. Exec. Time (seconds)	Avg. Std. Deviation
1 domU	16.997	-	4.888	-
2 domUs	11.948	4.228	6.721	0.443
3 domUs	17.739	2.546	8.227	0.290
4 domUs	20.496	4.665	10.288	0.415

From Table 3, it can be observed that compared with the data from Table 1, after the introduction of noise workload, the execution time of is.A.2 with the original version of MPI library increased greatly, whereas that with the revised MPI library experienced a reasonable increasing. This is because if one of the VCPUs of a VM hosting is.A.2 linked with the original MPI library is scheduled simultaneously with the VCPU of the noise VM on the testbed, the busy-polling mechanism will force the VCPU to run 30ms, which results wasting of processing resources and the prolonged execution time as well. Moreover, this also results the confusion in statistically fair-share mechanism employed by Credit scheduler, which thus results the high standard deviation on execution time of the benchmark programs running simultaneously in the co-existing domUs. In this case, the revised MPI library by our proposal can greatly decrease the execution time as well as the standard deviation.

4.3 Problems remained

The patch assume the user application start MPI invocations with MPI_Init and stop that with MPI_Finalize, therefore, it does not deal with the threaded invocation of MPI routines.