vIC Notes

Background

- CPU cost is high if IO rate is high because of the Interrupts
- Solution is coalescing IO completion interrupts
- If you wait to deliver Interrupt, latency is increased. Other challenge is Synchronous IOs. Delaying the completion of prior IOs can delay the issue of future ones.
- If lots of IO request to choose, better scheduling algorithm can be used.
- What if the result of delivering the IO interrupt is known, it will be a big plus. But it is not possible.

MIDL, MCC

- MIDL (max interrupt delivery latency) is really important as it decides the maximum latency that can be tolerated
- MCC (max coalesce count) : no of accumulated completions before sending an interrupt
- These two parameters decides when to deliver interrupts (based on which one hits first)
- Use of High resolution timer is required in traditional techniques. This adds additional overheads of handling timer interrupts. But how to do synchronization without timer? First order effect is cost of processing interrupts for the timer. Second order effect is caching and TLB pollution.

VMkernel vs. the VMM

VMM

- Responsible for correct and efficient virtualization of the architecture
- emulation of high performance virtual devices
- It is a kind of a “process” to VMkernel

VMkernel

- Layer that controls access to physical resources from VMs.

New Approach

- Using delivery ratio (R belongs to (0,1]) . No timers.
- R is ratio of interrupts sent to guest by total IO completions. R is variable, depends on CIF. CIF is the number of outstanding IO. Increase R when CIF is low and decrease when CIF is high so as to balance the CPU efficiency gained by coalescing against additional

latency that may be added. Workload determines CIF values. Typical value is 32, or 64, not big.
- Usefulness of CIF: High CIF implies more options to coalesce. For workloads with high CIF, the extra delay works well since they are able to amortize the cost of the interrupt being delivered to process more than one IO.
- For Low IOPS and Low CIF, R =1 i.e no coalescing.
- Other ways instead of counter-based algorithms, such as random, table-based

IPI (Inter Processor Interrupt)

- IPI sent from the CPU that received the hardware interrupt to the remote CPU where the VM is running for notification purposes
- Useful for compute intensive tasks or cases where the guest is running without any exits to the VMM as the code inside VMM periodically checks for any incoming interrupts inside shared queues.
- Provides Latency Optimization. Not required for correctness.
- But IPIs are costly. Try to minimize their occurrence.

Calculation of R

- Avoid floating point calculations for R. Also, simple ratio of the form 1/x based on a counter x would result in drastic variations of R.
- Approach used : Deliver (countUp) out of every (skipUp) interrupts (R = countUp/skipUp)
- Algorithm 1 given in the paper shows the algorithm used for delivery ratio determination.
- Reevaluate the vIC rate at every period called “epoch period”. Default value : 200ms (selected based on the experiment). Use algorithm 1 provided in the paper. Allows the vIC to react to changes in the workload.

Interrupt Delivery

- Decide whether to post an interrupt to the guest or wait to coalesce it with a future one.
- Algorithm 2 in the paper provides the logic
- Calculate for each IO completion whether or not to deliver a virtual interrupt given the ratio R i.e countUp/skipUp.
- Algorithm 2 : Count up from 1 till countUp - 1 delivering an interrupt each time. Continue to count up till skipUp - 1 while skipping each time. When Counter reaches skipUp, it is reset back to 1 and an interrupt is delivered.
-

IPI optimization: Reducing the Expensive IPIs

- New time-stamp added to the shared area between the VMM and the VMkernel where completion queues are managed.
- It tracks when was the last time the VMM posted an IO completion virtual interrupt to the guest
- Before firing an IPI, check the current time against what the VMM has posted to the shared area. Post IPI If the time difference is greater than a configurable threshold.

Evaluation

- Also increases from 4/5 to 1/6
- CPU cycle per IO drops, meaning efficiency increases
- (column 5 in table 2 – column 8 in table 2) / column 8 in table 2
- Figure 3 and 4 shows R distribution depends on workload that has IO burst patterns, independent of vIC value
- Again # of interrupts decreases, but latency goes up: at the median delivery rate 1/3, 10k mean IOPS, the latency increase would be 200us
- Setting a good IPI send threshold is important but nontrivial and even workload dependent.

Related work

- Several patents are cited