vIC: Interrupt Coalescing for Virtual Machine Storage Device IO

Irfan Ahmad
Ajay Gulati, Ali Mashtizadeh

USENIX Annual Technical Conference
June 15, 2011

Outline

Interrupts

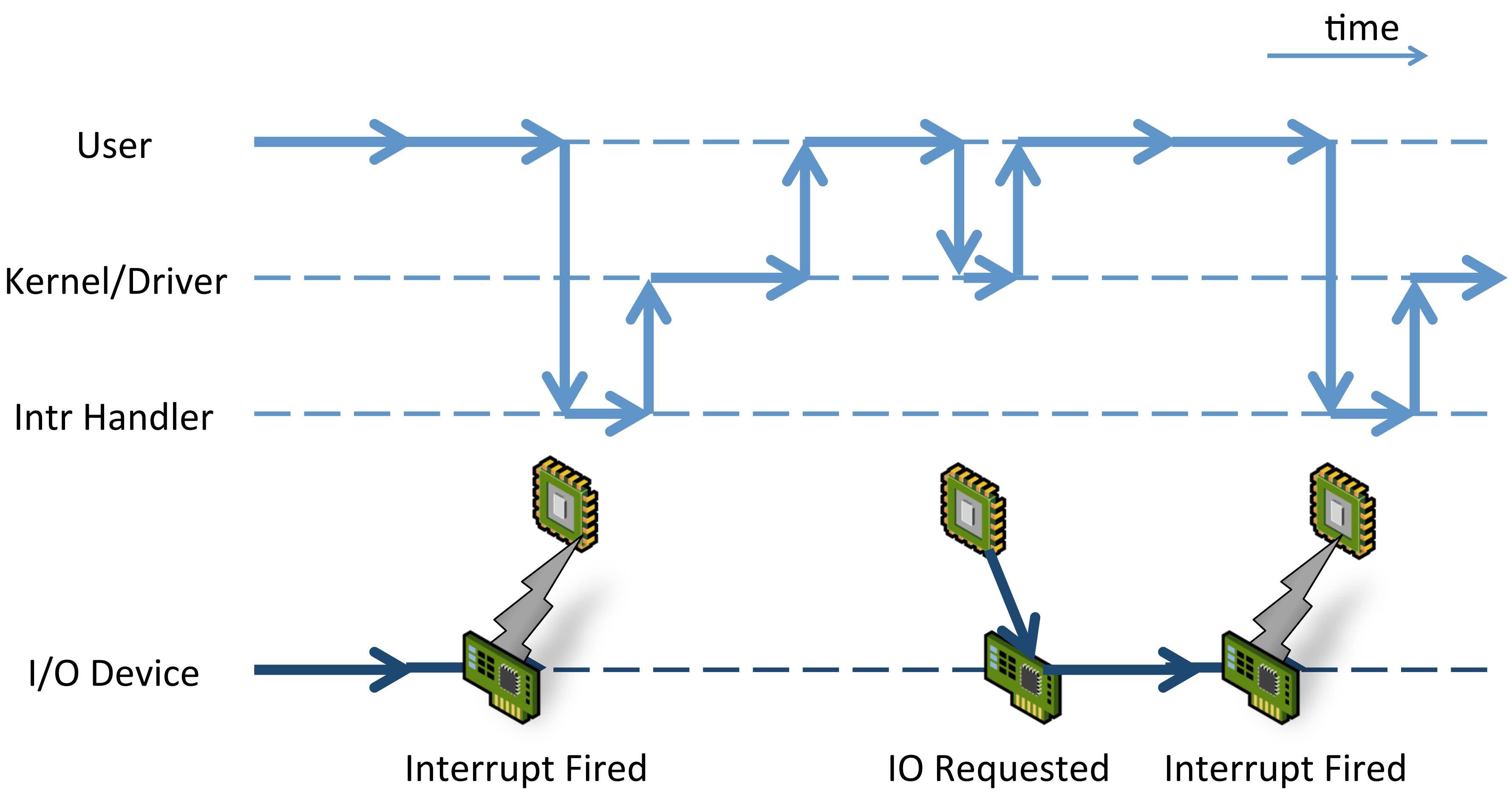
Interrupts Coalesced

Virtual Interrupts

Virtual Interrupts Coalesced

Inter-Processor Interrupts Coalesced

Interrupts


time
User
Kernel/Driver
Intr Handler
I/O Device
Interrupt Fired
IO Requested
Interrupt Fired

Interrupts

“It was a great invention,
but also a Box of Pandora.”
-- E.W. Dijkstra

Source: EWD 1303
http://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1303.html

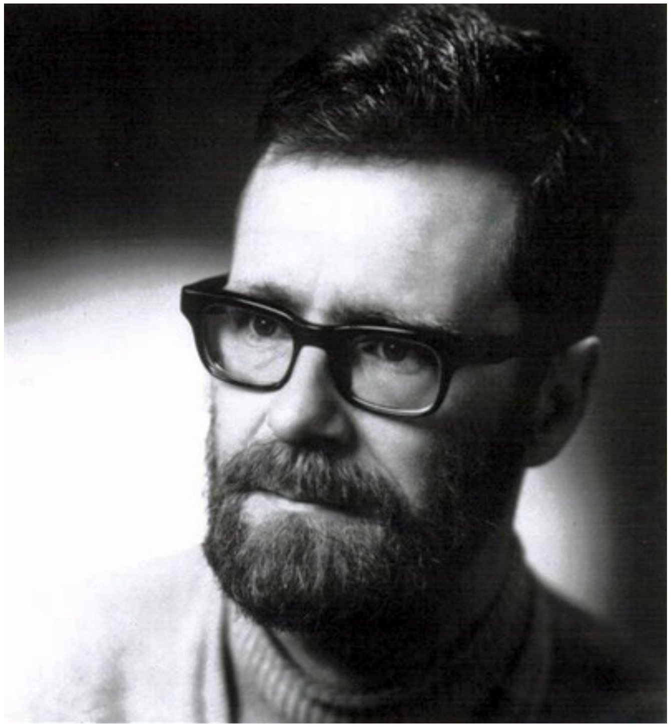
Electrologica X-1

Source: People Behind Informatics, An exhibition in memory of Dahl, Dijkstra, Nygaard
http://cs-exhibitions.uni-klu.ac.at/

Picture: http://cs-exhibitions.uni-klu.ac.at/fileadmin/template/pictures/Dijkstra_electrologica.jpg

E. W. Djikstra

“Halfway the functional design of the X1, I guess early 1957, Bram [J. Loopstra] and Carel [S. Scholten] confronted me with the idea of the interrupt, and I remember that I panicked, being used to machines with reproducible behaviour. How was I going to identify a bug if I had introduced one?”

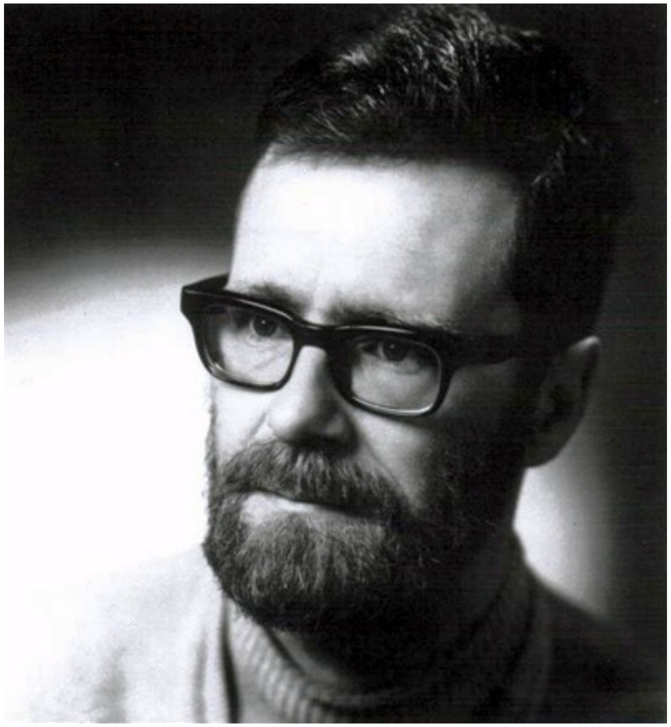


Picture: http://27.media.tumblr.com/tumblr_l4nhw73hGD1qz8lbio1_400.jpg

Source: EWD 1303 http://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1303.html
Dijkstra’s PhD dissertation (on X-1): http://www.cs.utexas.edu/users/EWD/PhDthesis/PhDthesis.PDF

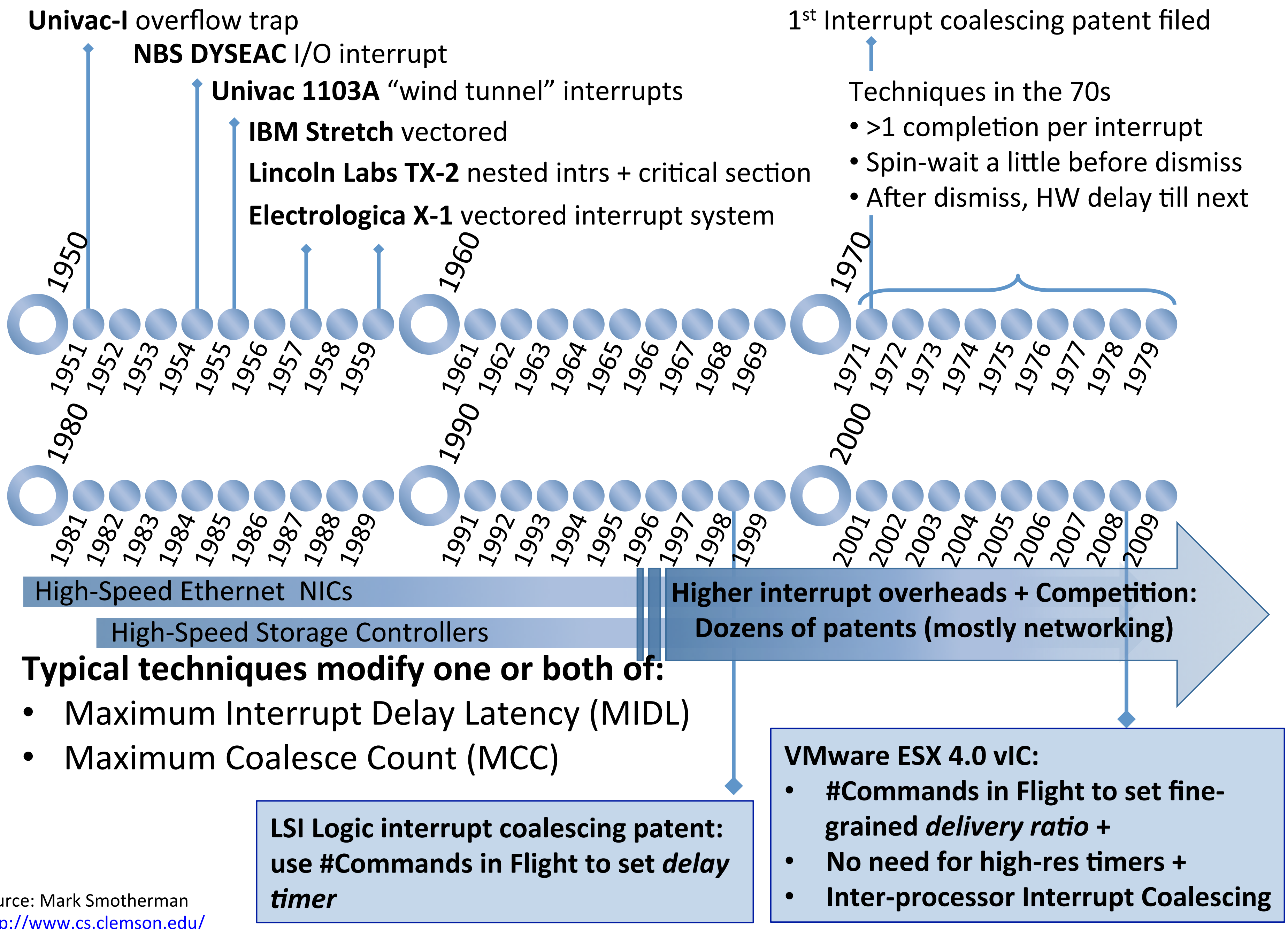
E. W. Djikstra

"After I had delayed the decision to include the interrupt for 3 months, Bram and Carel flattered me out of my resistance, it was decided that an interrupt would be included and I began to study the problem."



Picture: http://27.media.tumblr.com/tumblr_l4nhw73hGD1qz8lbio1_400.jpg

Source: EWD 1303 http://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1303.html
Dijkstra's PhD dissertation (on X-1): http://www.cs.utexas.edu/users/EWD/PhDthesis/PhDthesis.PDF



Univac-I overflow trap

NBS DYSEAC I/O interrupt

Univac 1103A “wind tunnel” interrupts

IBM Stretch vectored

Lincoln Labs TX-2 nested intrs + critical section

Electrologica X-1 vectored interrupt system

1st Interrupt coalescing patent filed

Techniques in the 70s

- >1 completion per interrupt
- Spin-wait a little before dismiss
- After dismiss, HW delay till next

High-Speed Ethernet NICs

High-Speed Storage Controllers

Higher interrupt overheads + Competition:
Dozens of patents (mostly networking)

Typical techniques modify one or both of:

- Maximum Interrupt Delay Latency (MIDL)
- Maximum Coalesce Count (MCC)

LSI Logic interrupt coalescing patent: use #Commands in Flight to set *delay timer*

VMware ESX 4.0 vIC:

- **#Commands in Flight to set fine-grained *delivery ratio* +**
- **No need for high-res timers +**
- **Inter-processor Interrupt Coalescing**

Source: Mark Smotherman
http://www.cs.clemson.edu/~mark/interrupts.html

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO

Virtual Interrupts are Different?

- Real HW I/O controllers are embedded systems
- Device emulation executes on general purpose, multi-user, time-shared architectures
- Can’t install timers for 100 microseconds intervals
  - Host would be overwhelmed by interrupt storm
  - Other VMs would be impacted
  - Shouldn’t solve interrupt coalescing for VMs by increasing interrupt rate on host*!*

First Intuition Behind vIC

- Let's pretend HW IO completions are "timers"
  - But, just can't program them to our desired rate
  - So, let's piggyback the ShouldDeliverInterrupt() logic on real HW completion handlers
- HW controllers: deliver when internal timers fire
- vIC: let's only deliver in line with HW completion
- Motivates using a *delivery ratio* instead of timer
  - Deliver a virtual interrupt for every nth completion

Delivery Ratio

- Naïve implementation: deliver an interrupt for 1 of every *n* HW completions
- Equivalent of the typical max coalesce count (MCC) parameter in HW controllers
- Problem in MCC: limits delivery ratio to be 1/n
  - E.g. 1/1, 1/2, 1/3, 1/4, etc.
  - Can't express, say, 80% delivery ratio
- Experiments suggest 1.0->0.5 jump too drastic

Delivery Ratio

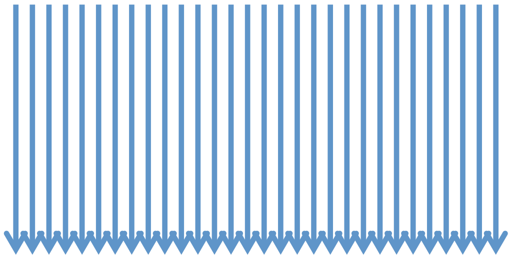
- Use two counting parameters (MCC has one)
  1. countUp
  2. skipUp
- Express arbitrary fractional delivery rate

☑☑☑☑☒

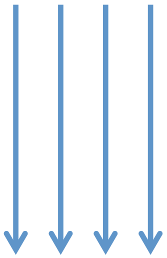
80% delivery: Deliver up to 4, Skip up to 5

Second Intuition Behind vIC

Commands in Flight: 32



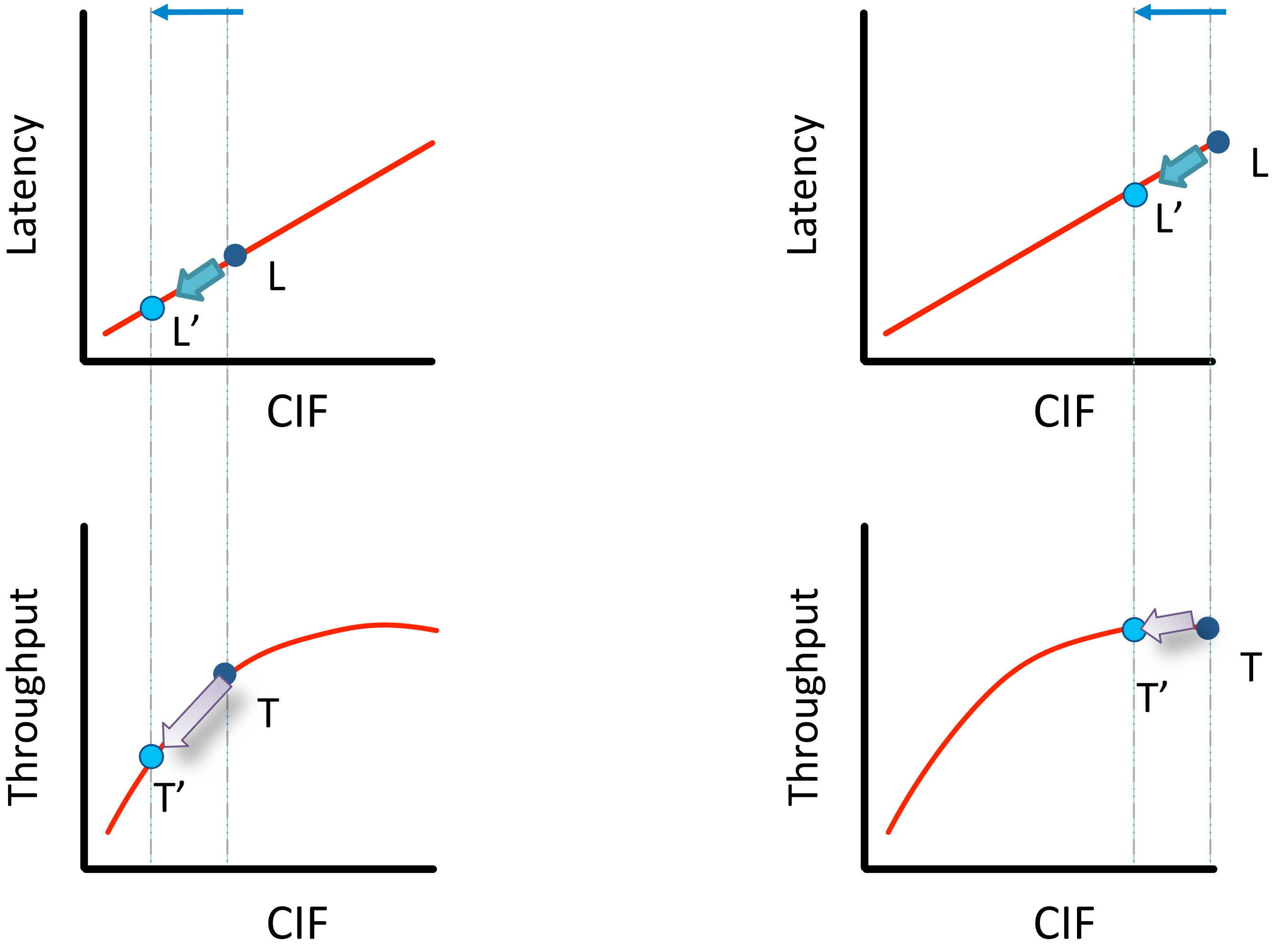
Commands in Flight: 4



- Suppose a scheme coalesces 2 completions
- With CIF of 32, pipeline remains mostly full
- With CIF of 4, pipeline is half empty!

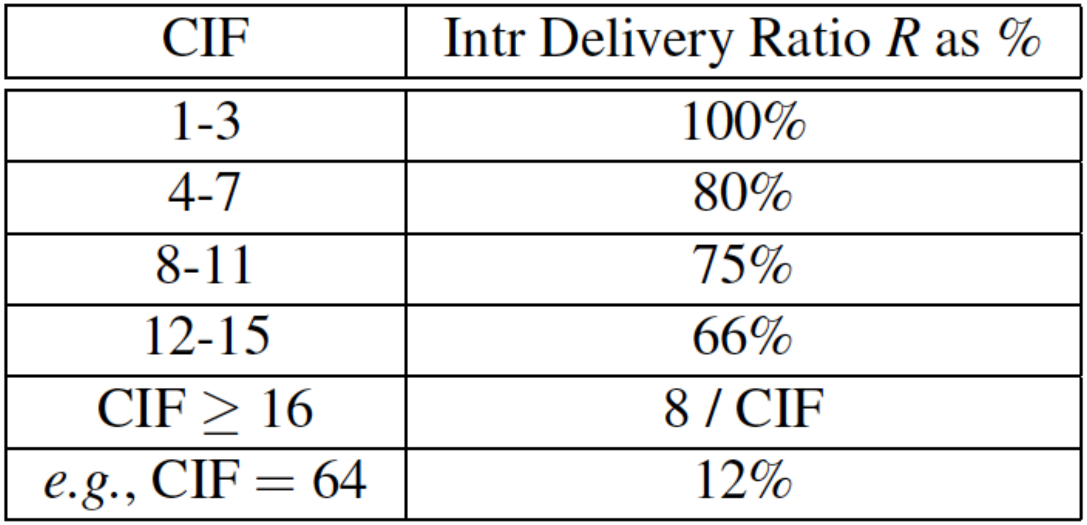
➔ make delivery ratio a function of CIF

Delivery Ratio: CIF Dependence



Delivery Ratio: CIF Dependence

- Measure dynamic Commands in Flight (CIF)
- Vary delivery ratio ***R*** inversely with CIF



CIF	Intr Delivery Ratio R as %
1-3	100%
4-7	80%
8-11	75%
12-15	66%
CIF $\geq$ 16	8 / CIF
e.g., CIF $=$ 64	12%

Interrupt delivery ratio (**R**) as a function of CIF.

Loose ends

- What if next HW completion never comes?
  - There is always a future I/O when CIF > 0 ☺
  - Still, short-circuit to deliver f/ low CIF situations
- What if the hardware completions are too far apart: could cause high latency?
  - Measure and automatically enable/disable vIC
- Trickle I/O
  - Measure and automatically enable/disable vIC

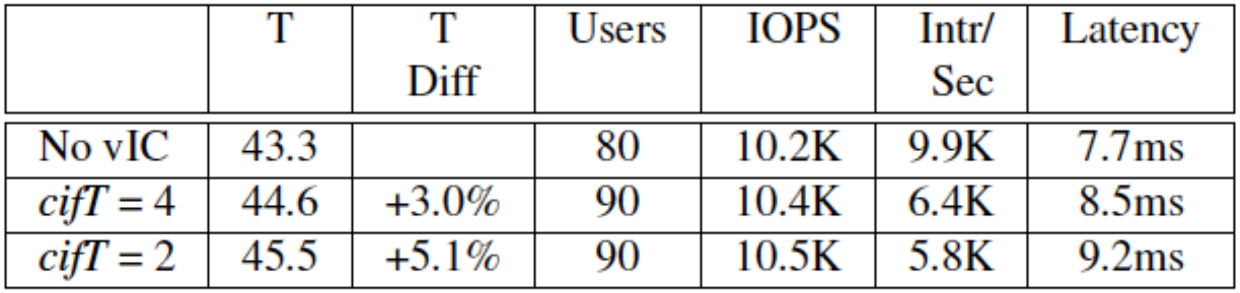
vIC Implementation

- Portable to other hypervisors on any CPU architecture. Also to firmware/hardware
- No floating point
- No int div or RDTSC in critical path
- Increase in the 64-bit VMM:
  .text: +400 bytes
  .data: +104 bytes.
- LSI Logic emulation in VMM: <120 new LoC
- IPI coalescing logic in the Vmkernel: 50 new LoC

Results

- Application benchmarks
  - GSBlaster and SQLIOSim
  - Throughput (IOPS) increase by up to 19%
  - Improve CPU efficiency up to 17%
- Let's look at TPC-C next
  - transcation rate increased by up to 5%

Internal TPC-C Testbed



	T	T Diff	Users	IOPS	Intr/ Sec	Latency
No vIC	43.3		80	10.2K	9.9K	7.7ms
cifT = 4	44.6	+3.0%	90	10.4K	6.4K	8.5ms
cifT = 2	45.5	+5.1%	90	10.5K	5.8K	9.2ms

Throughput Increased

IOPS Increased

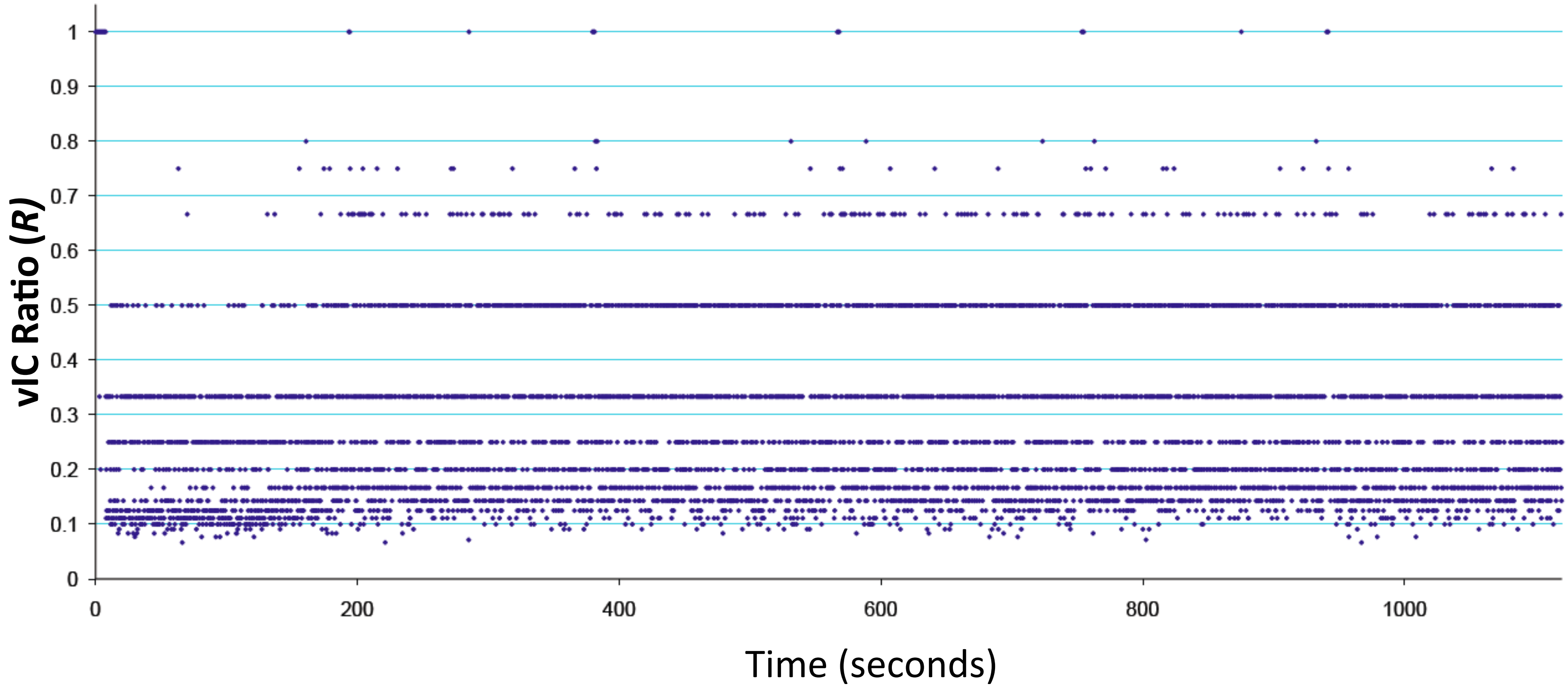
Proportional Latency increase

More Users

Interrupts Decreased

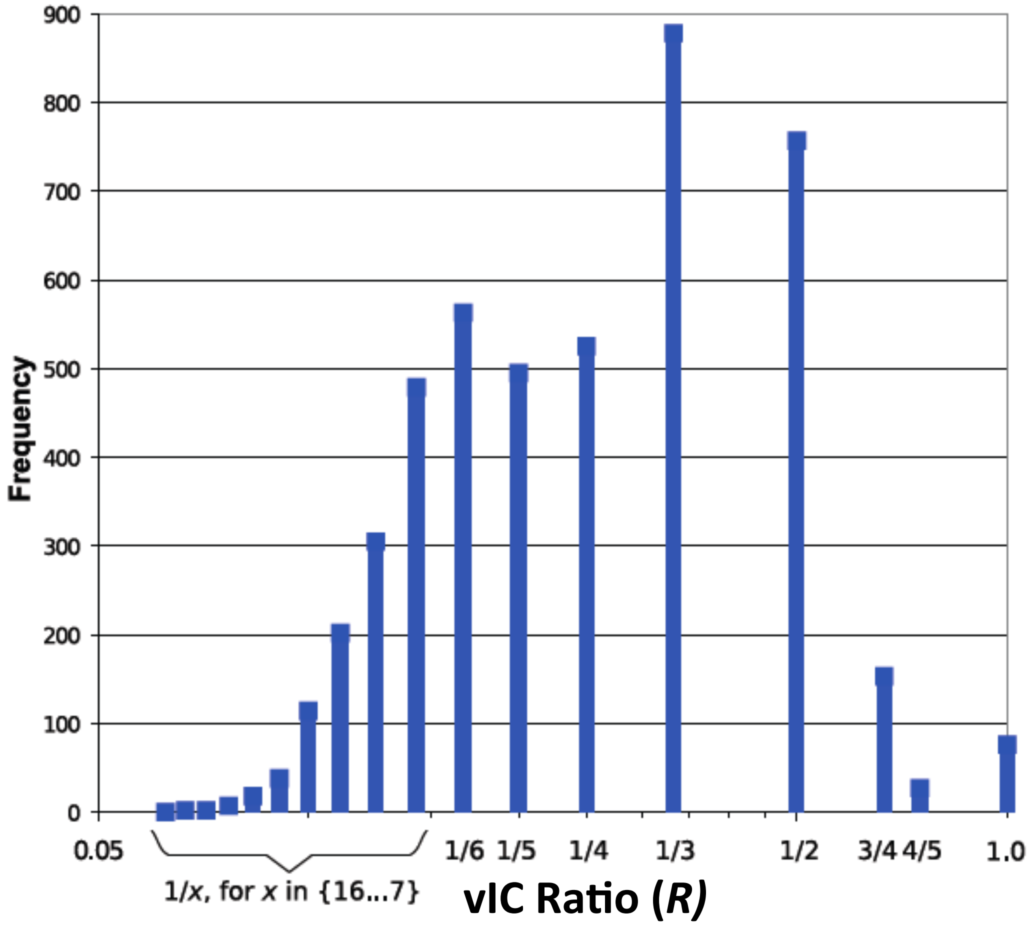
[1] **Non-comparable implementation; results not TPC-C^{TM} compliant; deviations include: batch benchmark, undersized database.**

Dynamic Adaptation (TPC-C)



Virtual interrupt coalescing rate, ***R***.
Online adaptation by vIC to burstiness in outstanding IOs of the workload

Dynamic Adaptation (TPC-C)



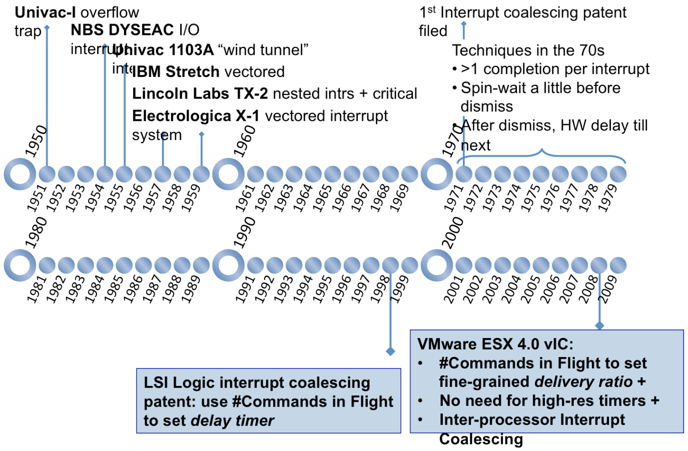
Virtual interrupt coalescing ratios, *R*, during our TPC-C run.
x-axis log-scale.

vIC Deployment Experience

- vIC default in VMware’s LSI Logic virtual adapter on ESX (since v. 4.0 released 2Q ’09)
- Till now, no performance bug reports

Key Takeaways

- 60-yr old problem revisited
- Encouraging results
  - TPC-C by 5%, other by 18%+
  - Take another look at *your* interrupt subsystem
  - IPI coalescing very beneficial



Univac-I overflow trap

NBS DYSEAC I/O interrupt

Univac 1103A "wind tunnel" inte

IBM Stretch vectored

Lincoln Labs TX-2 nested intrs + critical

Electrologica X-1 vectored interrupt system

1st Interrupt coalescing patent filed

Techniques in the 70s
- >1 completion per interrupt
- Spin-wait a little before dismiss
- After dismiss, HW delay till next

LSI Logic interrupt coalescing patent: use #Commands in Flight to set *delay timer*

VMware ESX 4.0 vIC:
- **#Commands in Flight to set fine-grained *delivery ratio* +**
- **No need for high-res timers +**
- **Inter-processor Interrupt Coalescing**

- More optimization opportunities exist in vIC
- Change the rules when they weigh you down
  - What about networking?
  - An equivalent of CIF there (TCP window size?)