

# Kernel Regressions and increasing OS noise

Christoph Lameter, Ph.D.  
*TAB, The Linux Foundation*





## Overview

- Intro
- What is a regression?
- Effects of regressions
- OS Noise characteristics
- Latency troubles
- Kernel bloat causing regressions
- Solutions (.....)
- Conclusion



## Regressions

- Functionality and/or OS performance in later version is worse than in earlier versions.
- We used to track Kernel regressions between versions.
- Always had consistently regressions in performance over time aside from areas where we did targeted performance optimizations.
- For some industries wrong focus: We need low latency not high performance.  
“Optimizations” make things worse.



## HPC / Gaming / Financials and Latency

- HPC apps need regular intervals to perform a rendezvous in order to reach high performance.
- Gaming: The low latency player wins.
- Financials: Latency determines who can make a trade.
- Situation is that old kernels are run instead of newer ones because of regressions (RHEL3 and RHEL4 popular). 64 bit: No thanks.
- Currently three to fourfold latencies(!!!) due to regressions in multiple subsystems.
- Only losers run upstream....



## OS Noise

- Application experiences random delays.
- On the application CPU the following may occur:
  - Scheduling of OS threads
  - Hardware interrupts
  - Faults (page faults?)
  - Timers trigger
  - Scheduler may run other tasks
- Disturbances increase with higher scheduling frequency.
- Lower scheduling frequency makes the delays longer that an application sees.

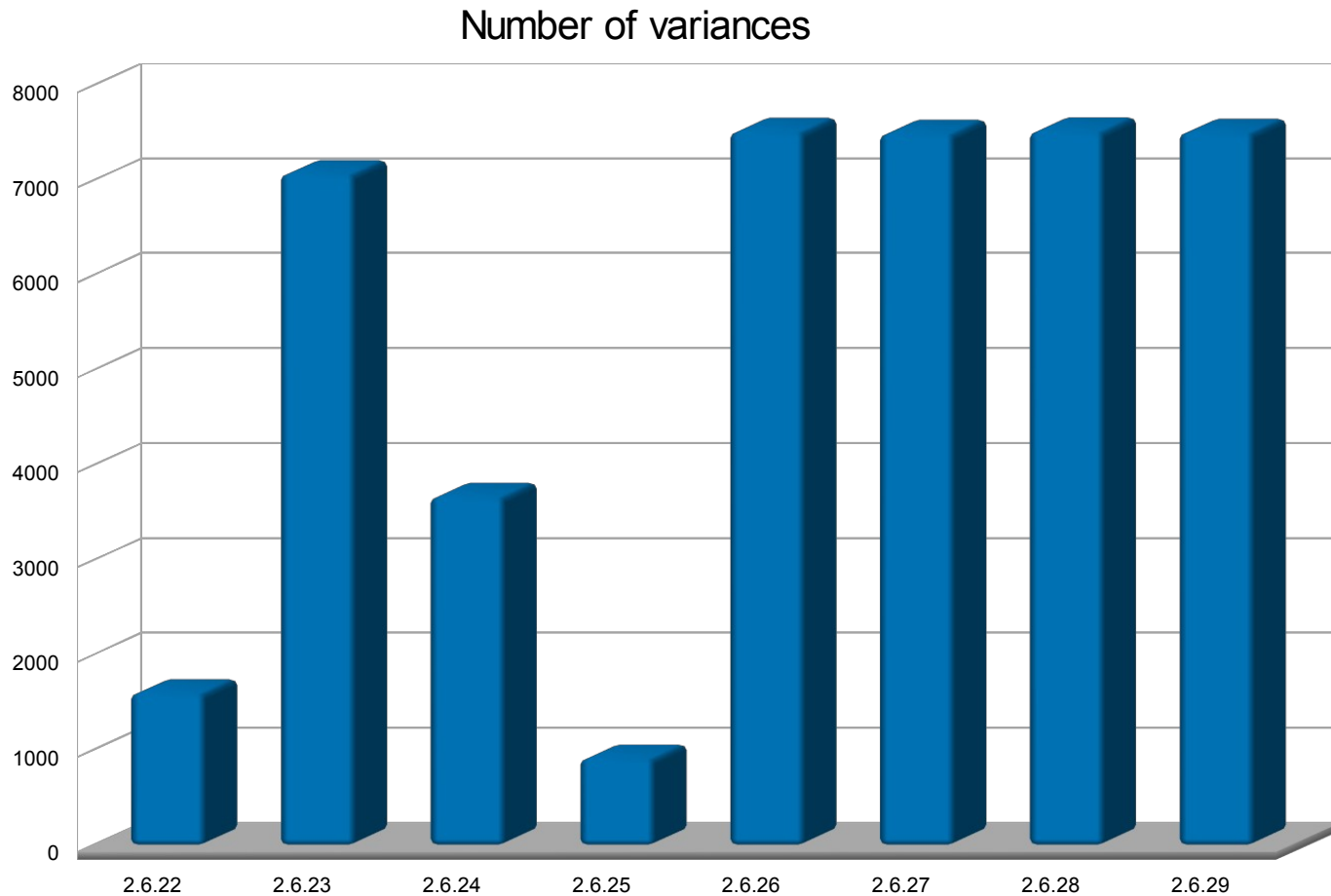


## Low Latency tools ([gentwo.org/II](http://gentwo.org/II))

- **latencytest**: An OS noise measurement tool
  - Number of OS reschedules
  - Number of Faults
  - Holdoffs and their frequency
- **udpping**: Measure minimum communication latencies.
- **mcast** : Multicast latency and stress testing.



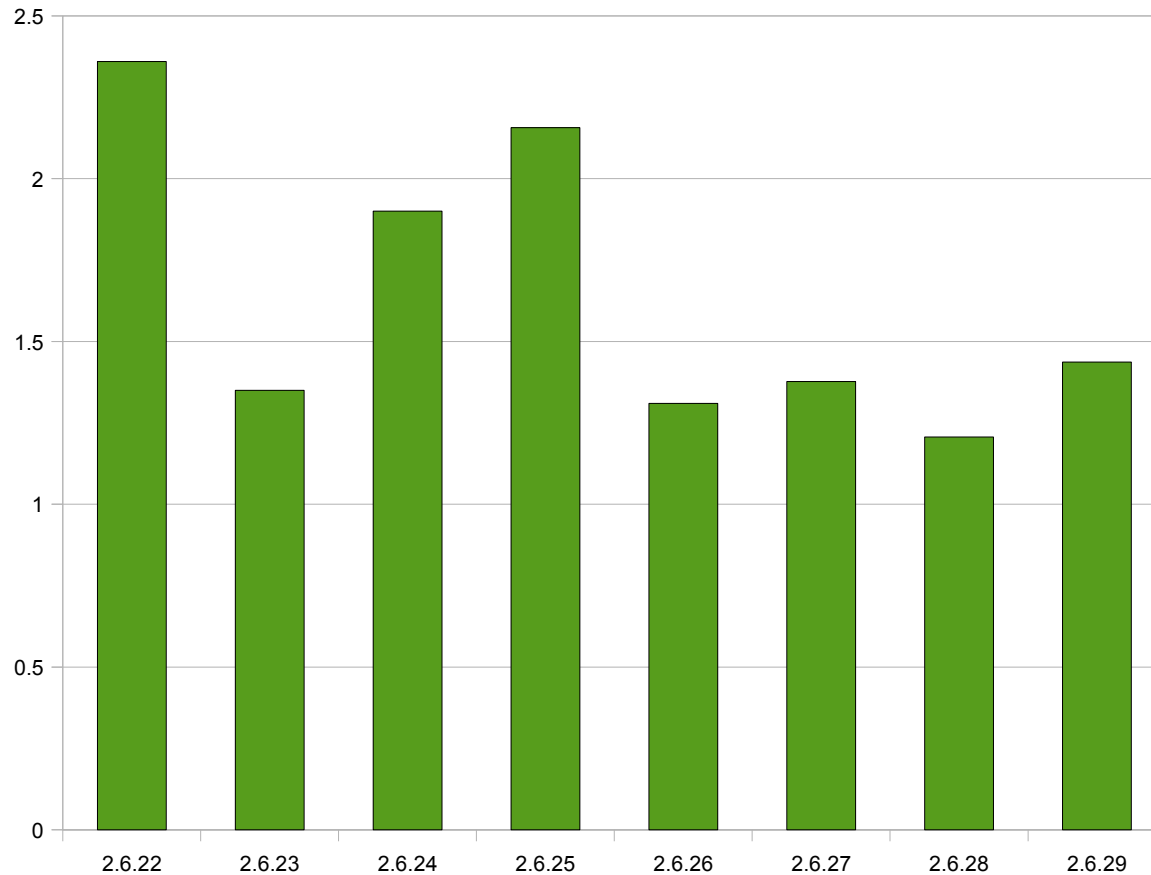
## Noise created by the Linux OS





## Length of Noise periods (microseconds)

Average length of interruption

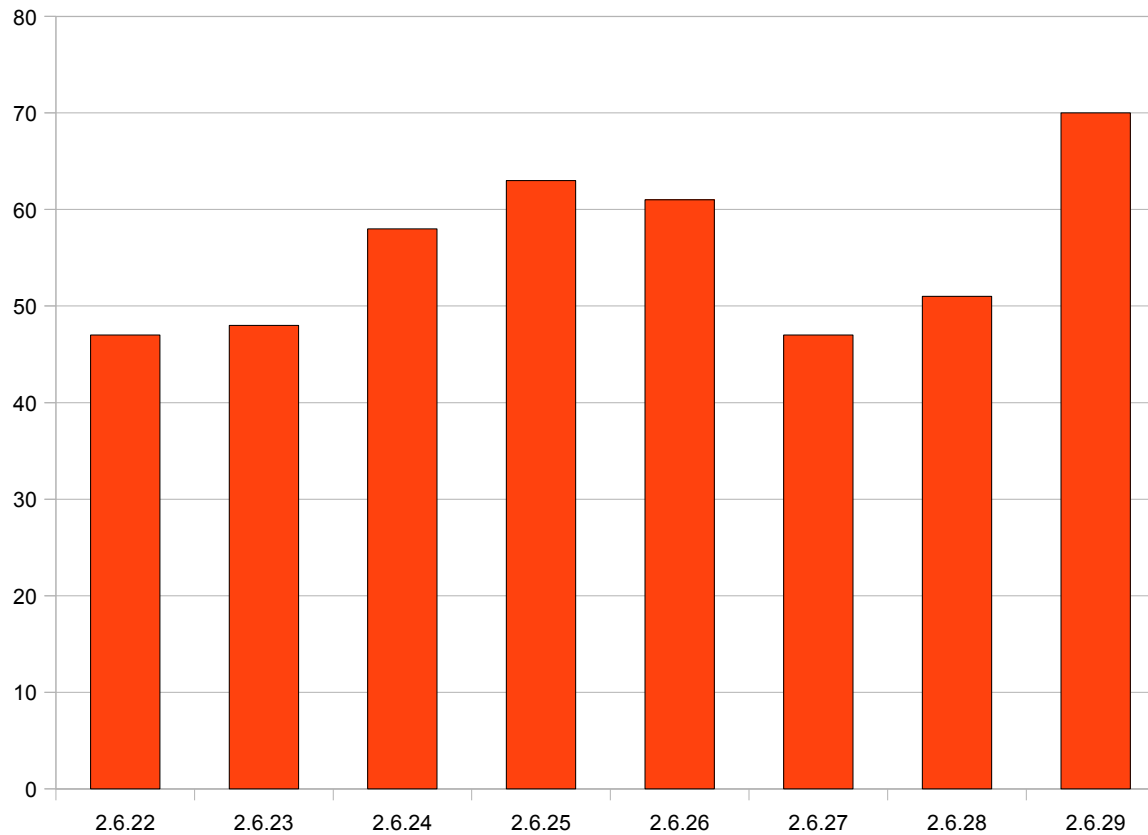






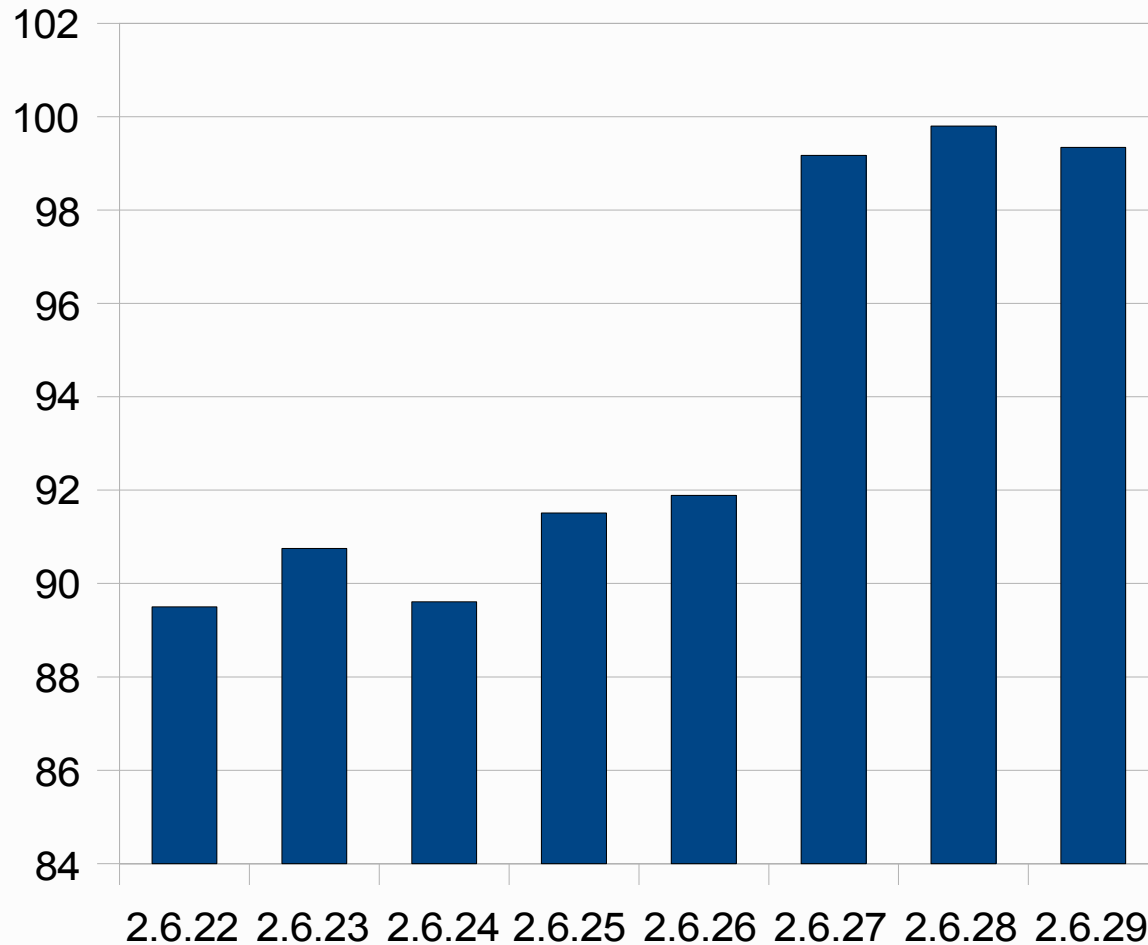
## Scheduler interventions

Number of scheduler context changes



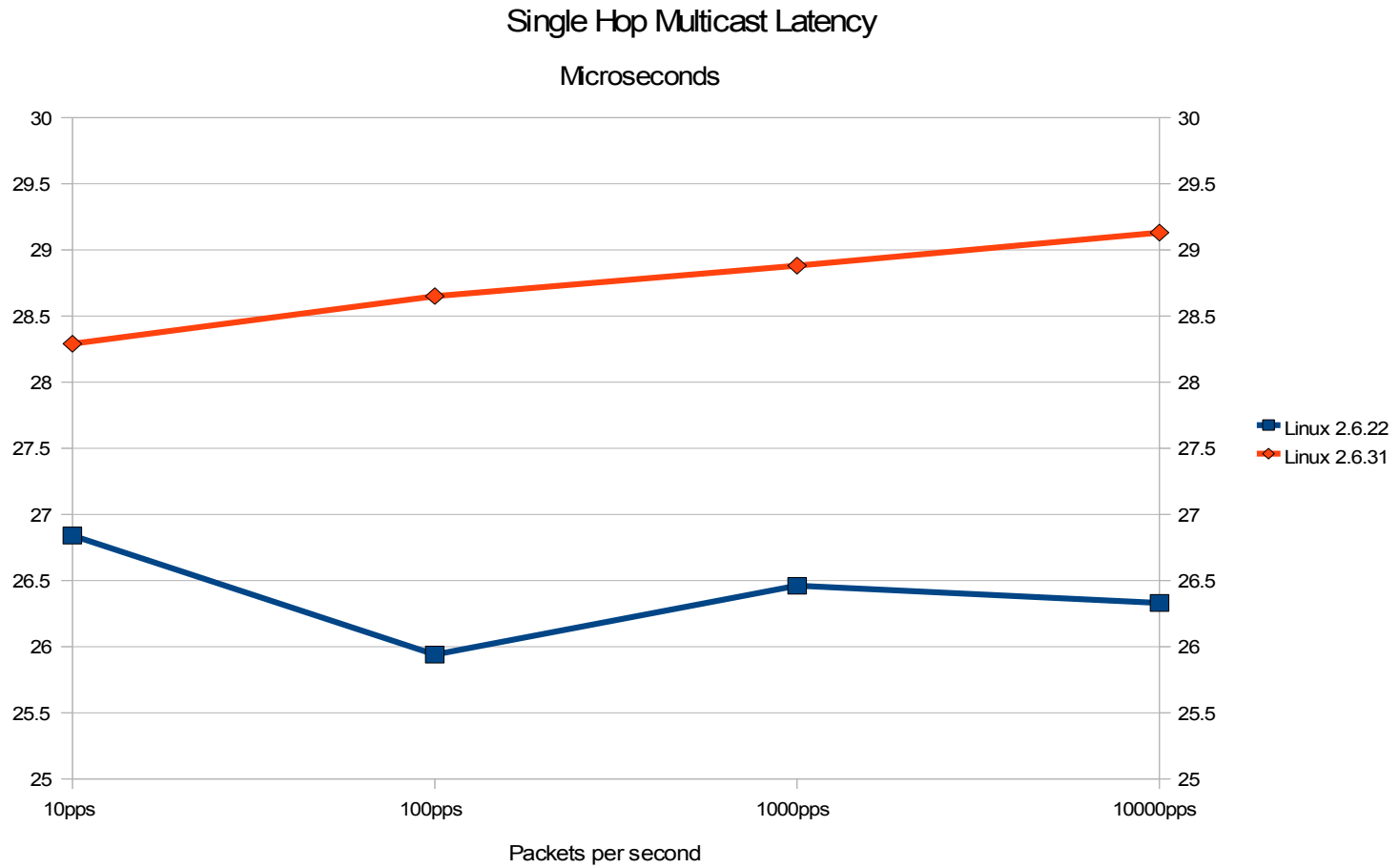


## UDP ping pong times (microseconds)





# Multicast latency regressions





## Latency regressions

- Loss of 2-3 microseconds in 10 kernel releases.
- 64 bit kernels show much higher variance. Caused by larger icache footprint is higher.
- Issues exist at low message frequencies due to more advanced power savings logic in newer kernels.



## Latency countermeasure

- Prefaulting
- Warming up caches
- Pinning
- Rt priorities
- Thread local variables
- Run old software (RH3, RH4?)
- Restrict OS scheduling to subset of CPUs.



## Measures to reduce OS noise

- Process pinning: taskset
- Realtime priorities: chrt
- Prefaulting pages
- Cache prepopulation
- OS features off
- Smaller cache footprint
- OS should not defer processing.



## OS bypass

- Atomic ops in user space
- Polling instead of sleeping
- Virtual NIC in user space
- Infiniband RDMA
- Packet MMAPed sockets
- User space RX and TX buffers
- Custom offload libraries



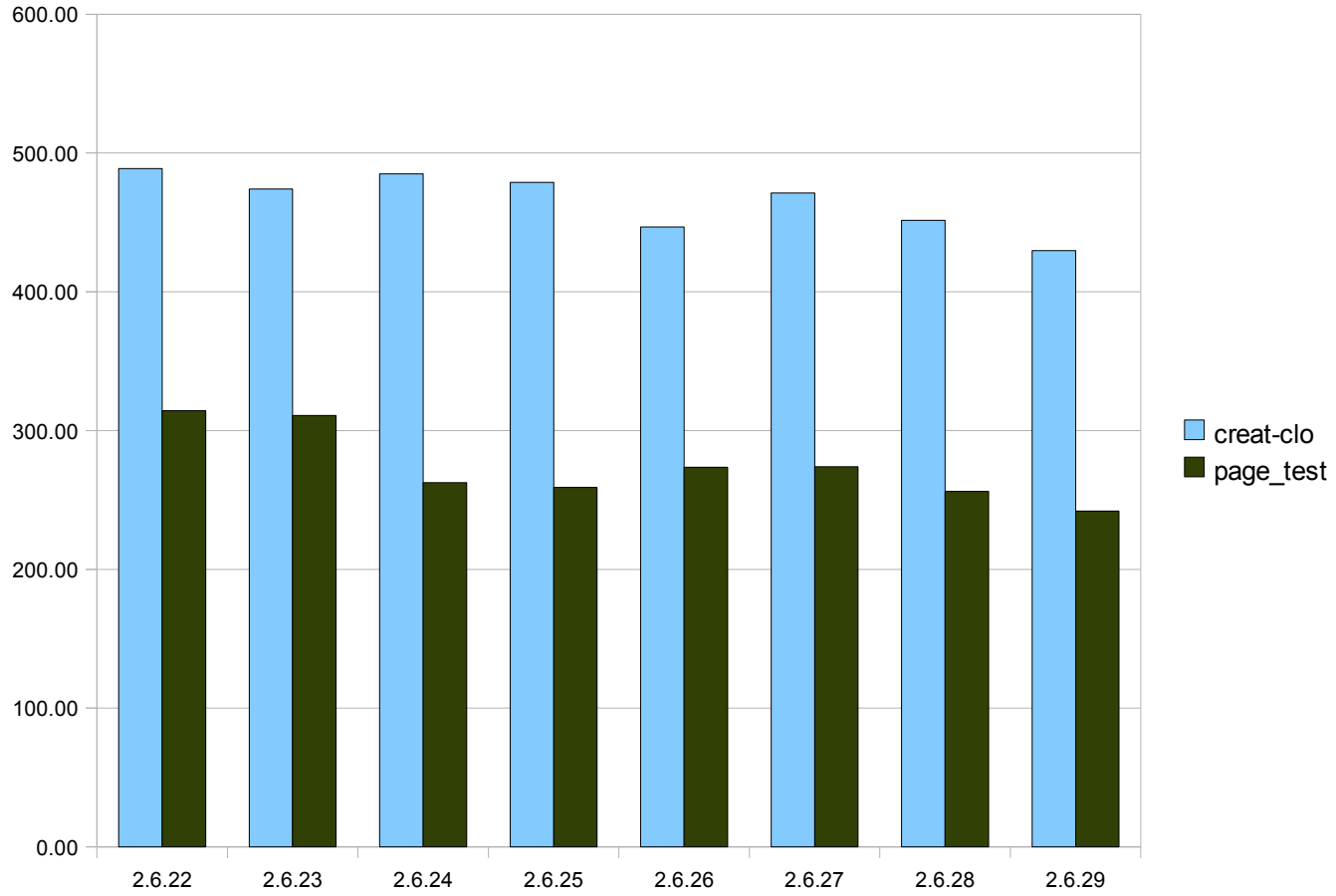
## Software bloat

- Featuritis: Over time software always adds more features to satisfy various needs.
- Caching schemes introduces to increase heavy callers of functions.
- Penalizing infrequent callers.
- “Performance” increases, software slower.
- Cache footprint grows bigger. Penalties if L1 cache overused.





## Kernel bloat/AIM9 regressions





## Plans to address regressions

- Advanced features to control affinity of queuing in network stack.
- Deadline scheduling algorithm?
- Enable NUMA options?
- Do not schedule on a subset of processors?
- Move noise to a single processor (0)?
- Add more features that require complexity?



## Things to do

- Track latencies and kernel performance over long time periods
- Establish better tools to measure OS noise.
- “Its in the noise” is really saying that a change may cause additional latencies.
- Feedback to OS developers re OS noise
- Establish latencies for critical OS paths and benchmark newly released kernels.



## End

- Questions
- Maybe answers?



## Time and Space considerations for Latencies

- Latencies are bound with distances due to relativistic speed issues. Nothing violates the speed of light.
- Latencies limit system design and processing speed.
- Signal propagation speeds limit system sizes and create NUMA latency issues.
- Only some latencies can be avoided.
- Bandwidth increases instead of Speed increases.



## 1 second

- Time needed for a signal to reach the moon.
- Upper bound on any reasonable network latency.
- VM statistics interval in the Linux kernel.
- High performance counters are only guaranteed to be upto date after one second.



## 100 milliseconds

- A signal can reach all of the earth's surface.
- High speed consumer link latency
- Half of TCP retry interval.
- Minimum human reaction speed.
- Frequently used timeout for devices.



## 10 milliseconds

- 2000 km distance. Signal can reach surrounding metropolitan area.
- Timer interrupt for systems with 100HZ.
- Major page fault (page read in from disk)
- Time interval for a process to receive another time slice if another process has to be run first.





## 1 milisecond

- 200km distance. Systems in your city.
- Sound travels 34 centimeters. Sound from the speakers reach your ear.
- Seek time of harddisks.
- Max camera shutter speed.



## 100 microseconds

- 20km. Signal confined to LAN or building.
- Maximum tolerable interrupt hold off.
- Ethernet ping pong times in a LAN via 1 Gb/s networking.



## 10 microseconds

- 2km. Signal confined to a LAN.
- Relativistic time distortion in GPS
- Minor page fault (Copy on write)
- Duration of timer interrupt
- Duration of hardware interrupt
- Typical IRQ holdoff.
- Duration of system call.
- Context switch.



## 1 microsecond

- 200m.
- Wire segment delay.
- Signal stays within a system.
- Resolution of `gettimeofday()` system call.
- PTE miss and reloading of TLB.
- Start of hardware interrupt processing.



## 100 nanoseconds

- 20m. Within the room.
- Cache miss. Time needed to fetch data from memory.
- TLB miss.



## Shot but not dead Or the miraculous resurrection...

- Video gaming across a LAN.
- Two gamers access the same game server.
- Game data propagates according to the distance.
- Gamer with long latency can shoot and the enemy will die on his screen since his system knows the position of the enemy.
- But at the time that the notification of this event reaches the server the other player has already made several other moves.
- So the game server reckons it was a miss and the enemy who just died a horrible death is miraculously resurrected and escapes.