LOW LATENCY 101
• Introduction to low latency
• Technology Challenges
• Structured approach to latency improvement
• Pushing the envelope – latest approaches and technologies being deployed to achieve low latency

Slides available from my website at http://www.informatix-sol.com
Latency value proposition

- It may be necessary to lower latency just to retain a Market making position
- It may be desirable to improve latency to improve ones ranking or to enter a new market
- It may be desirable to improve latency to stop getting picked off by competitors
- Traditionally limited to Equities but now becoming critical for FX and other OTC’s
- The value of lower latency is intrinsically understood but extremely difficult to quantify
- Lower latency systems cost more to build and deploy
  - Challenge is to find the right balance between investment in lowering latency and obtaining a return on this investment
Typical Asset class latencies

Source: Donavan Ransome, Citihub, 2009
Performance challenge

• Unconstrained demand
  • Worsened as network pipes have got fatter

• All markets are growing both in volume and # of trades
  • High frequency and Algo trading now accounts for around 80% of total trading\(^1\) and growing
  • Average Equity trade value has decreased as bigger trades are automatically segmented to reduce market impact
  • # of cancellations increased as Traders/Algo’s seek to discover the full book
  • Increasing sophistication of algorithms makes it more difficult to predict behaviour

• Everybody bursts at the same time
  • Produces biggest trading opportunities – need to stay in the market
  • Greatest risk of loss if fail to keep up – may get picked off

• Developer productivity improvements (e.g. Java and .NET) has made the control and visibility of the underlying infrastructure more difficult

• Distributed computer architectures have more complex performance and reliability characteristics

• Need to increase throughput and lower latency simultaneously

• Can no longer rely on Moore’s law to save us – only delivering more cores at the same speed, not getting any faster

\(^{1}\) High frequency trading firms account for 73% of all US equity trading volume, AdvancedTrading.com, 2009.

FX has been reported as having about 25% of orders automatically traded in 2006.
Burst capacity and computer meltdown

• We need massive headroom free to allow for bursts
  • typical guide is 10:1
• But computers degrade in a non linear fashion
• Can be triggered by a shortage of any one of it’s resources:
  • CPU cycles
  • Memory
  • I/O channel capacity
  • Disk transfers
  • Network transfers
• Shortage of one can trigger another e.g. shortage of memory causing CPU and I/O thrashing
• Distributed deployments add an order of magnitude more complexity

Typical system response time against throughput
Suffering consequences of Amdahl’s Law

• New CPU’s are just giving us more cores at a similar speed
  • Stopped getting faster at 60nm geometries

• Dependant on multi-threading capability of the application to utilize additional cores

• Sequential code streams increasingly dominate latency

• Resulting in additional cores providing diminishing benefit

• Sacrificing cores for speed may improve latency in some circumstances e.g. Intel Turboboost but power saving measures introduce unpredictability
Network serialization cost

The latency cost of sending a packet

e.g for 500byte packet
10Mb/s = 566µS
100Mb/s = 56µS
1Gb/s = 5.7µS
10Gb/s = 0.57µS

• Serialization latency is added at every server and most network ports due to store and forward of most network packets
• Cut-through only extends end to end in very simple network topologies
Cut Through Switching Constraints

• Traditionally Ethernet switches used a store and forward approach which received and verified the entire packet before starting to send
  • Introduces latency proportional to packet size and line speed
  • Ensures only good packets are forwarded

• Cut-through technology allows the packet to start transmitting before it is completely received
  • In practice there is still a minimum transmission unit, theoretically 6 bytes but typically in the 64-1024 bytes range that has to be received before this transmission can start
  • Vendor quoted latency numbers for cut-through switches are based on FIFO measurement i.e. time delay to first bit to pass through the switch, this is independent of packet size. Minimum time is quoted.

• Cut-through constraints
  • Only between ports of equal line speed
    • Most uplinks are a different line speed
  • Only when egress port is idle
  • Assumes forwarding table has been set up on the port, i.e. Flow established
  • Propagates invalid packets, e.g. CRC errors, runts etc.
Propagation cost of distance

Predicted round trip latency cost of routing over a long line for an existing deployment

e.g. for 1000 bytes at 1 Gbps (including access network)

1000 km = 10.1 ms
100 km = 1.2 ms
10 km = 303 μs
1 km = 213 μs
0 km = 203 μs

Modelled using distance, packet size, bandwidth, serialization, router delay, server delay. Router and serialization latency dominate as distance decreases.
Low latency Networking

• Optimal carrier selection to optimize propagation latency
  • Demand SLA’s based on latency
• Select high line speed to reduce serialization
• Collapse network to reduce number of network devices in path
  • Increase L2 size and span, reduce # of L3 hops
• Optimize selection of network devices
  • Cut through switches
  • Low latency routers with good Non Drop Rates (NDR)
  • Test with Multicast packets since many switches will drop these early to leave space in buffers for Unicast
  • Only use SPAN ports where delay is predictable and impact to source being monitored in minimal
• Plan to handle microbursts e.g. Plenty of spare bandwidth
• Managing drops and discards to reduce jitter
Ethernet Packet Drops and discards

Server 1

- User
- Kernel context switch
- Socket buffers
- TCP/IP
- NIC Buffers

Server 2

- User
- Kernel context switch
- Socket buffers
- TCP/IP
- NIC Buffers

Queued using flow control

Discarded when full

UDP (including Multicast) discarded when space limited

Error Detection and Correction By TCP

TOR Switch
1:2.4 contention

Core Routers
1:10 contention

Informatix Solutions
Network microbursts

Microbursts can lead to packet loss, message retries and jitter.

Courtesy of Cisco
Latency and Jitter

• Whilst there is a lot of attention to latency, the consequences of jitter can be more damaging e.g.
  • A single lost message can wipe out all profits from hours of successful trading

• Need to reduce both latency and jitter

• Requires latency distribution analysis and goals around median not minimum latencies attained

Reducing latency shifts the mean to the left
Reducing jitter tightens the distribution and reduces the range of possible values
Component breakdown of latency

- Latency can be broken down into the following components
  - \( L = P + N + S + I + AP \)

- \( P \) = Propagation time - sending the bits along the wire, speed of light constrained

- \( N \) = Network packet processing – routing, switching and protection

- \( S \) = Serialization time - pulling the bits on/off the wire

- \( I \) = Interrupt handling time – receiving the packet on a server

- \( AP \) = Application Processing time

- In most situations it involves a data message exchange between a minimum of 2 systems i.e.
  - \( L = AP^1 + I^1 + S^1 + N^1 + P + N^2 + S^2 + I^2 + AP^2 \)

- This can be expanded to add more hops as required

- We can break this down into three main focus areas for optimization:
  - Application = \( \text{sum}(AP^1:Ap^n) \)
  - Infrastructure Latency = \( \text{sum}(I^1:I^n) + \text{sum}(N^1:N^n) + \text{sum}(S^1:S^n) \)
  - Propagation Latency = \( P \)
Low Latency is relative

Goal is for Infrastructure to account for 10-20% of Application Latency

<table>
<thead>
<tr>
<th></th>
<th>Latency tolerant</th>
<th>Low Latency</th>
<th>Ultra Low latency</th>
<th>eXtreme Low latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>&gt; 1 mS</td>
<td>&lt; 10 mS</td>
<td>&lt; 500 µS</td>
<td>&lt; 50 µS</td>
</tr>
<tr>
<td>(end-to-end including</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>middleware and run</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time environments)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td>&gt; 1 mS</td>
<td>&lt; 1 mS</td>
<td>&lt; 50 µS</td>
<td>&lt; 5 µS</td>
</tr>
<tr>
<td>(LAN, Server, storage, OS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Propagation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location (CoLo),</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carrier and path</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Packet capture with time synchronization

- SPAN ports on those switches where it is safe to use
- Grandmaster clock to synchronize all timestamp systems
- IEEE 1588 Precision Time Protocol
- Pipe into CEP for real time alerting and feedback
- Aggregate into DW for reporting
Time Synchronization Protocols

Network Time Protocol (NTP)
• Widespread deployment both internally and on the Internet
• Sync to Internet or GPS
• Included with Linux, UNIX and Windows OS and most Cisco routers
• Achieves synchronization in the 1-20mS range on a LAN

PPS
• 1 pulse per second, electrical signal carried on co-ax used for local hw clock synchronization. Often output from GPS synchronized Grand master clock and used to distribute to stratum 2 servers

Precision Time Protocol (PTP) IEEE 1588
• Developed for Real time automation systems. Typically uses dedicated Ethernet LAN. Uses a Servo based feedback design which allows configurable settle and accuracy.
• Accuracy is dependent on implementation but can typically achieves synchronization in the 5-20 µS range.
• Stratum 0 clock at each location synced to GPS
Technology examples for each latency category

Increasing levels of sophistication to reduce latency to the minimum
Need to match this investment to the resulting business benefit

Increasing cost and complexity

Latency Tolerant
- 100Mb Ethernet
- Java
- In-line RDBMS

Low Latency
- 1Ge Tuned TCP/IP
- Low latency Java
- In memory DB

Ultra low Latency
- 10G Ethernet
- OpenOnload TCP
- Pinned memory
- CPU binding
- Low latency kernel – Soft RT
- FP Scheduling,
- C or C++

eXtreme low Latency
- 40G InfiniBand
- TCP bypass
- FPGA
- Hard Real Time
- RT Scheduling,
- RT Exec
- RDMA memory replication
- Assembler
General Server tips

- Do not run under any hypervisor
- Where a choice is possible select the lowest latency Operating System platform
  - Linux, Solaris, Windows
- Use fastest cores available
  - i.e. Currently x86 rather than SPARC
  - Consider Turbo boost or sacrificing cores e.g. Intel Everest
  - Decide where to trade between speed and reliability
    - E.g. Water cooled X5680 running at 4.4Ghz
- Use fastest network connection to reduce serialization delays and reduce likelihood of dropped packets - 1Gbp/s minimum
  - Consider InfiniBand as alternative to Ethernet between critical servers
- Minimize network contention between critical servers to reduce likelihood of dropped packets and flatten networks by reducing hops
- OS’s are tuned by default for throughput, re-tune for low latency
- Tune the TCP/IP network stack or bypass it altogether
- Optimize program runtime
  - Prioritize critical applications and minimize running system by disabling unwanted services
  - Interrupt masking and allocation
  - Pinning to Cores and locking memory
Disk I/O

- Avoid it, or at least move it out of critical code paths
  - Achieve durability by replicating critical data across the network, 100’s of times quicker than a disk I/O
- In Memory caches
  - Apache Memcached – read only, for subsequent reads
  - Cacheonix, Oracle Coherence, GemStone, GigaSpaces, RNA networks, TIBCO ActiveSpaces, Terracotta
- Tune Storage subsystem
  - Heavily cached storage for random I/O and write mostly
    - Linux, disable elevator re-order algorithm
      - Boot time parameter - `elevator == noop`
  - FLASH drives primarily for random I/O
    - PCIe card bases solutions provide better b/w (e.g. Fusion-IO, ioExtreme)
- Tune File System
  - Select the most appropriate FS type
    - EXT3 is a good all-rounder
    - XFS for large files
    - Consider specialist file systems such as:
      - OCFS32, HSF2, BTRFS
Linux minimum kernel level

- Newer tends to be better for low latency
  - Preemptible kernel first appeared in 2.5 but took a long while to appear in mainstream
    - See if CONFIG_PREEMPT enabled
      - `grep /boot/config-`uname` -r` CONFIG_PREEMPT
  - Major distro’s leave disabled. Can enable and rebuild kernel but need to verify has not exposed device driver issues, although these are now very rare
- OFED RDMA – mainstreamed in kernel 2.6.14
- CFS scheduler – mainstreamed in kernel 2.6.24
- High resolution timer (nanoclock) in kernel 2.6.16 and high resolution events in 2.6.21
- Soft Real time support is available as a patch for 2.6.24 and later, or pre-packaged as RedHat MRG or SuSE SLERT
  - CONFIG_PREEMPT_RT
- RH MRG and Novell SLERT both include RT patch and OFED (although not latest versions)
- Personally saw big latency wins when I moved from RH 5.3 using 2.6.18 to a 2.6.27 kernel
- OFED is pushed back into kernel but vendors prefer you to use their “value added” stack which replaces this.
  - Latest OFED build tends to have best latency and has more bug fixes, particularly for Ethernet RDMA and iWARP
Tuning Network Stack on Servers

• Be selective about NIC’s
  • Big differences on latency and drops, caused by both hardware and quality of driver
  • 10G NICS either optimized for FCoE or Latency

• Consider your network stack

  ![Network Stack Diagram]

  - **Kernel TCP/IP**
    - 1G Linux
      - 48 µS RTT
    - 10G Linux
      - 17 µS RTT
  - **User Space TCP/IP**
    - OpenOnload
      - > 8 µS RTT
  - **Bypass TCP/IP**
    - OFED
      - > 2 µS RTT

• Tune NIC driver
• Tune TCP/IP stack
• Note the tradeoffs – lower latency increases CPU overhead, power and complexity
latency Tuning tips – Application Level

- Pre-fault (i.e. touch) all memory and then pin all application virtual memory into physical pages with `mlockall(3c)`.
- Disable Nagle using `TCP_NODELAY` option on socket. Good practice to do this even if Nagle is disabled at the system level for code portability.
- Zero copy of files using `sendfile(2)`
- Bypass TCP if possible by using OFED
- Linux specific
  - CPU groups and interrupt binding with `cpuset`
  - Stop CPU slow down, designed to save power – `cpuspeed`
  - Move to Fixed Scheduling priorities using `sched_setscheduler()` with CFS
  - Detect issues and tune with `mpstat`, `memprof`, `strace`, `ltrace`, `blktrace`, `valgrind`, `latencytop`, `tcptract`,
    - Kernel trapping with `systemtap`, similar to `dtrace`
- Solaris specific
  - Early bind or link statically all libraries to prevent delays loading dynamic libraries during runtime – `setenv LD_BIND_NOW =1`
  - Run in a processor group which is masked from interrupts and pin memory using `psrset(1m)`
  - Set to a fixed priority 60 using FX scheduling class using `priocntl(1m)`. This will keep it above all TS, IA, and FSS processes.
  - Zones can be used for additional isolation since they add no detectable latency
  - Avoid application lock contention issues – identify with `mpstat(1m)` , `prstat(1m)` , `plockstat(1m)` and `dtrace`
Example Linux Server Tuning

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>average</th>
<th>median</th>
<th>max</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>untuned Broadcom 1G</td>
<td>64.81</td>
<td>67.00</td>
<td>66.95</td>
<td>279</td>
<td>1.14</td>
</tr>
<tr>
<td>multicast</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>untuned Broadcom 1G</td>
<td>35.66</td>
<td>39.22</td>
<td>37.99</td>
<td>652</td>
<td>5.76</td>
</tr>
<tr>
<td>TCP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tuned Broadcom 1G</td>
<td>33.93</td>
<td>38.04</td>
<td>37.89</td>
<td>269</td>
<td>1.26</td>
</tr>
<tr>
<td>TCP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tuned Broadcom 1G</td>
<td>27.93</td>
<td>32.02</td>
<td>31.91</td>
<td>253</td>
<td>1.33</td>
</tr>
<tr>
<td>multicast</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

66% improvement of multicast median RTT latency

Results of initial Linux OS measurement and tuning
eXtreme Low Latency

- 40G InfiniBand
  - Available since 2009, creates large L2, low latency, high throughput networks with RDMA, bypassing TCP stacks
  - Compatible with existing TCP/IP socket programs with IPoIB
  - Extensive and well proven management capabilities
- TCP bypass
  - InfiniBand Verbs/RDMA, RoCEE or iWARP are main contenders and share common API’s
- Hard Real Time
  - Difficult to reduce latency and jitter for applications on conventional OS
  - Small RT Execs are the preferred to remove the I/O to Application boundary but are consequently limited to “simple” applications
- RDMA memory replication
  - Used primarily to achieve fast durability of data
- Assembler
  - Used for small critical code parts, mainly in conjunction with RT Execs
- FPGA - see over
FPGA

• Fuse Programmable Gate Array – programmable hardware
• Faster than software but slower than dedicated hardware e.g. ASIC or CPU
  • Cycle time only around 500Mhz but able to support high levels of parallelism
• Programming language for “hardware designs” described in VHDL (Verilog)
• Common “core” components available from FPGA vendor or licensable
  • Memory controllers, Ethernet NICs, PCI-E interfaces, ARM or FreeScale CPU core, TCP/IP stacks
    • Claim to be able to process a TCP/IP packet in 1µS
    • Moves algorithms closer to the wire
• Translation libraries of ‘C’ to FPGA inefficient due to the poor fit between the serialization inherent in procedural languages and the parallelism needed to exploit FPGA’s. Dataflow languages such as HyperStreams and Handel-C provide better abstractions.
• Primarily deployed in Appliances today due to the programming challenge and complexity of production operation
• Delay/bottleneck is still the FPGA to host messaging due to interrupt notification, memory copy and host OS application wakeup and reschedule
• Not a “silver bullet” but could be best solution in a small number of Use cases
TCP Bypass with OFED

- Initially just InfiniBand but now RDMA Ethernet with RoCEE and iWARP
- InfiniBand - just treat it like a TCP/IP socket
  - Running IP over InfiniBand (IPoIB)
  - Full support of all IP protocols and multicast
- Pre-load SDP to bypass TCP/IP stack for TCP sessions
- For optimal performance need to change API’s to use zcopy and RDMA transfer direct between Application memory spaces. Several API's of varying degrees of complexity and performance provide this
  - Reliable Datagram Sockets (RDS) – Used by Oracle for RAC
  - Direct Access Programming Library (DAPL) – used by IBM for DB2
  - Libibverbs – fastest access, near native access to HW VERBS – used by Websphere LLM, TIBCO FTL, Millennium IT, IBM GFS
    - Able to support both ping-pong and pub-sub Use cases
High Level Latency Improvement Program

**Measure**
Capture trade flows and measure trading time

**Existing Infrastructure Tuning**
Tune existing servers for low latency

**Identify**
Break down the end-to-end flows into constituent components

**Redesign**
Optimize location 80/20 rule for application infrastructure latency
Model Latency

**Research and Discovery**
Seek out and evaluate new technologies and products

**Deploy**
Capture trade flows and measure actual trading time
Refine Model

improve
Research and Discovery

- Continuous research into new technologies
- Structured investigation and testing

<table>
<thead>
<tr>
<th>References available</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Tested</td>
<td></td>
</tr>
<tr>
<td>Performance Tested</td>
<td></td>
</tr>
<tr>
<td>Functionality Tested</td>
<td></td>
</tr>
<tr>
<td>Constraints clearly documented</td>
<td></td>
</tr>
<tr>
<td>Can be shown to work</td>
<td></td>
</tr>
<tr>
<td>Prototypes Available</td>
<td></td>
</tr>
<tr>
<td>Under development</td>
<td></td>
</tr>
<tr>
<td>Plausible</td>
<td></td>
</tr>
</tbody>
</table>

Technologies:
- 10ge
- InfiniBand
- 40ge
- 100ge
- Openonload
- RoCEE
- FPGA
- GPU
- Optical switching
- Quantum
- Photonics
- Probability chips
About me

• Low latency design and Consultancy for FS clients
• Latency modelling and prediction
• Vendor independent and able to advise on technology and product selection
• Consultancy, design, performance reviews and education

www.informatix-sol.com                           richard@informatix-sol.com

Workshop Modules available including:

- Low latency tuning for Solaris and Linux
- Time Synchronization
- Low latency monitoring and reporting
- Programming with OFED
- Ethernet v. InfiniBand

- InfiniBand Overview
- Long Distance InfiniBand
- InfiniBand Management and Observability
- InfiniBand diagnostics and troubleshooting
- InfiniBand Product Selection