Latency Analysis in the Trading Community

MARK E. DAWSON, JR.
PERFORMANCE ANALYST

CHRISTOPH LAMETER

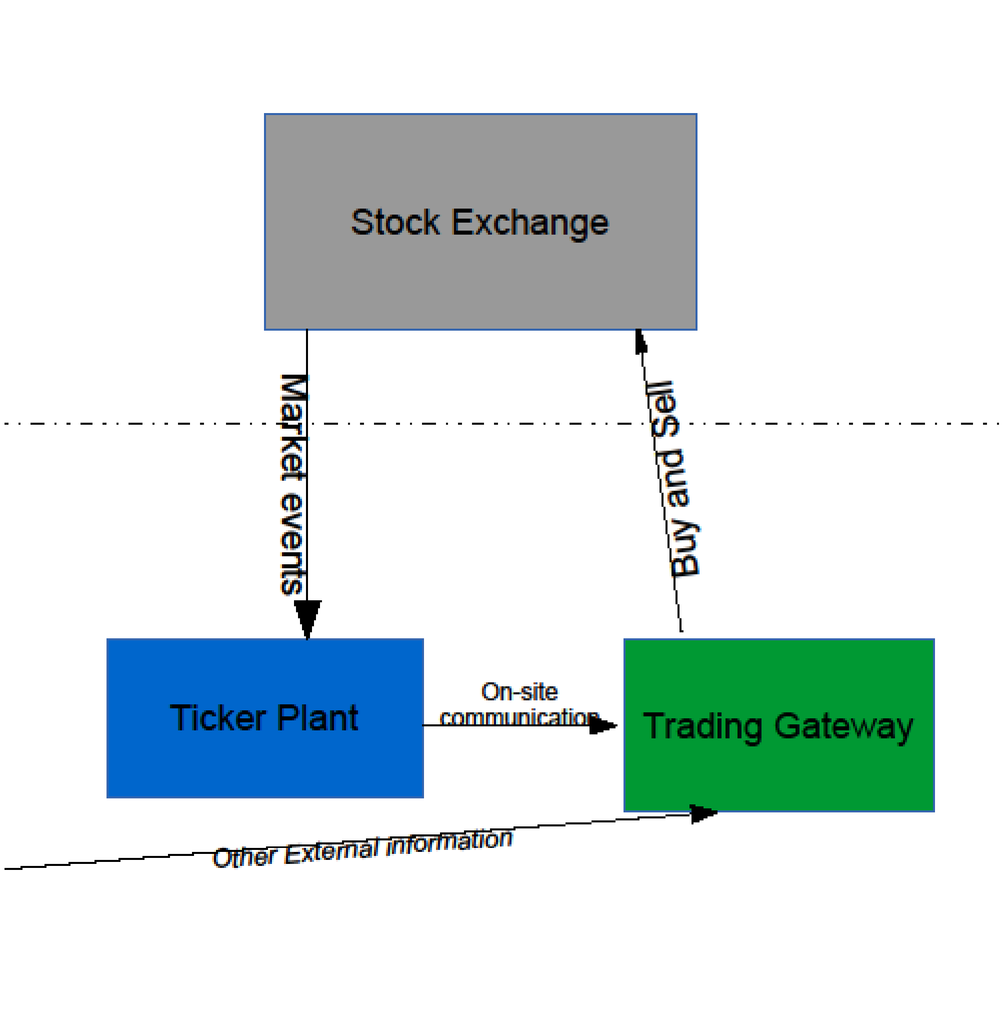
TRACING SUMMIT 2013
EDINBURGH

High Frequency Trading

- Competition based on speed of reaction to Market conditions
- Need for fast propagation of event information
- Latency analysis at multiple location
  - Intra-machine
  - Inter-machine (within & between datacenters)
  - Network ingress & egress points
- Timestamps on the system as well as in network packets
  - PTP
  - Network stack timestamps
  - Hardware timestamps
- Stock Exchange interaction challenges

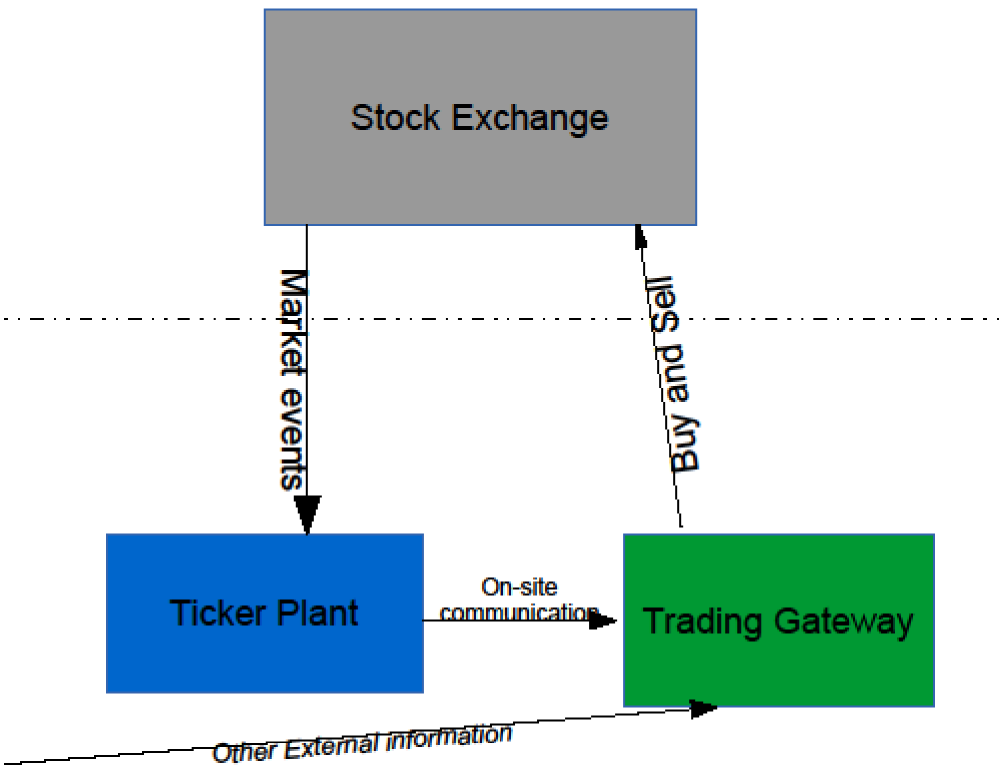
A Simplified Trading Environment

- Exchange
- Servers
- Local Network
- Remote Network
- Correlation
  - Within a server
  - Between servers
    - Same datacenter
    - Remote datacenters



Timing Issue Illustrated

1. SPY ETF vs. S&P 500 (index arbitrage)
2. How old is my "Market Event" data?
3. How quickly can I now "Buy or Sell"?
4. If new info arrives, how quickly can I cancel my Buy or Sell?
5. My strategy failed. Where along the chain was I too slow?



FTrace/Perf

- Local system time-stamping only
- Mostly kernel based
- No correlation capability

LTTng

- User space events
- Correlation of timestamps at the kernel level with user space functionality and driver timestamps

What is missing?

- Importing timestamps and events contained within a network packet
- Correlation with NIC HW timestamps
- Correlation of events between multiple systems
- Commercial solutions available in that space
  - Correlix
  - TSA
  - Corvil

Conclusion

- Questions
- Comments
- Opinions
- Ideas on how to solve this?

Latency Analysis in the Trading Community

APPENDIX

Need for Sub-microsecond Precision

- NTP inadequate
  - Millisecond accuracy in real world
- *gettimeofday* inadequate
  - Microsecond precision
- *rdtsc* too platform dependent
- Nanosecond precision *printk*

GPS issues

- GPS Antenna cabling varies widely in length
  - Range from 10ft to >1,500ft
- Rule-of-thumb in signal delay
  - ~1ns per foot of cable
- Greatly influences inter-datacenter time skew

New Timestamp Call

- Simpler return value
  - e.g., 64-bit value representing nanoseconds since epoch
  - Requires no division/multiplication like *clock_gettime*
- Low overhead
  - *gettimeofday* notoriously high overhead
  - *clock_gettime* still takes 20 - 30ns
  - *rdtsc* has low overhead but platform-specific

PTP Challenges

- Linux ptpd inadequate
  - Subject to OS noise
  - Does not account for PCIe read latency of HW timestamp
  - Does not offer lightweight time call intercept like *TimeKeeper*
  - Competitive servo algorithm?

- Accurate sync between system time and NIC time-stamping clocks
  - a la *hardpps* (NTP_PPS kconfig option)
  - Must work with tickless kernel

Packet Capture Devices

- Only nascent nanosecond support
- Require port spanning or tapping
  - Can impact timing or degrade signal
- On-the-wire timing only
  - Does not include application decision-point timing
- Do not scale
  - Most only offer 2 ports (costly additions)
  - Aggregation induces packet serialization
  - Only 1GbE offerings
- PCIe offering tradeoffs
  - Require OS mgmt.
  - No PPS or PTP reference clock support

Q&A

Questions?