Implementing Ultra Low Latency Data Center Services with Programmable Logic

John W. Lockwood, CEO: Algo-Logic Systems, Inc.
Why the Move to Programmable Logic?

“There are large challenges in scaling the performance of software now. The question is: ‘What’s next?’ We took a bet on programmable hardware.”
- Doug Burger, Microsoft

• Driving Metrics in the Data Center
  – Latency:
    • Reduce delay
    • Avoid jitter
  – Throughput
    • Processing packets at line rate
    • Handle 10G, 25G, 40G, and 100G
  – Power:
    • Driving cost of OpEx

• Field Programmable Gate Array (FPGA) logic moves into the CPU
• Microsoft accelerates BING search with FPGA
• Intel acquires Altera
Servers Accelerated with FPGA Gateware

• FPGA Augments Existing Servers
  – Can run on an expansion card (same size as a GPU)
  – Or may be integrated into the CPU socket
• GDN Applications run on FPGA
  – Implements low-latency, low-power, high-throughput data processing
Example of Low Latency Service: Key/Value Store

- **Key/Value Store (KVS)**
  - Simplifies implementation of large-scale distributed computation algorithms
  - Data Center Servers exchanges data over standard Ethernet

- **Challenges**
  - Operating System delays packets and limits throughput
  - Per-core processing inefficient at high-speed packet processing

- **Solutions**
  - Bypass kernel bypass with DPDK
  - Offload of packet processing with FPGA

<table>
<thead>
<tr>
<th>Examples:</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>Algo-Logic</td>
<td></td>
</tr>
<tr>
<td>Phone #</td>
<td>(408) 707-3740</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Directory</th>
<th>IP Address</th>
<th>Interface : MAC Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>204.2.34.5</td>
<td>Eth6 : 02:33:29:F2:AB:CC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Forwarding Tables</th>
<th>Content Hash</th>
<th>Storage Block ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ</td>
<td>948830038411</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data De-duplication</th>
<th>Order ID</th>
<th>Symbol, Side, Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATY11217911101</td>
<td>AAPL, B, 126.75</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stock Trading</th>
<th>Virtex</th>
<th>Edge List</th>
</tr>
</thead>
<tbody>
<tr>
<td>v140</td>
<td>v201, v206, v225</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graph Search</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v140</td>
<td></td>
</tr>
</tbody>
</table>
Mobile Application Servers Need Fast Key/Value Stores

- **Scalable backend services to share data**
  - Sensors { location, bio, movement, .. }
  - Social { status, dating, updates, multi-player games .. }
  - Media { video/security, audio/music, .. }
  - Communication { network status, handoff, short messages .. }
  - Database { users, providers, payments, travel, authentication, .. }

- **Must be able to scale**
  - as the number of users grows
  - **Scale up** to provide the best latency, throughput, and power
  - **Scale out** to increase storage capacity, throughput, and redundancy

- **Example:**
  - Mobile location sharing
Case Study: Implementing Uber with KVS

• Uber in 2014
  – 162,037 drivers in the US completed 4 or more trips
  – New drivers doubled every 6 months for past 2 years
  – Number of Uber Users = 8M
  – Number of Cities = 290
  – Total trips = 140M
  – Daily Trips = 1M

• Analysis and Assumptions
  – Assuming 25% of the drivers are active
  – 25% of 160k drivers = 40k active cars (<48k)
  – Drivers update position once per second = 40k IOW

• Implementation
  – Uber with on an Algo-Logic KVS card

Algo-Logic’s KVS solution for Mobile Applications

Scale UP for FASTER access to shared data
Algo-Logic's KVS solution for Mobile Applications

And SCALE-OUT quickly to increase storage capacity and throughput.
Provisioning and Measurements with GDN-Switch
Linux Software Socket KVS

OCSM Packet 10g Ethernet Intel 10G NIC Kernel Driver Message Process

LEGEND
Data Transfer =

Algo-Logic software on Intel 82598 10GE NIC and Core i7-4770k CPU
Algo-Logic KVS with DPDK: Bypass the Kernel

Intel 82598 DPDK Supported NIC

Receive Queue

Message Buffer

Transmit Queue

Message Process

Response Generation

Note: Message read once into CPU Cache

Legend:
- Control Handoff
- Data Transfer

Algo-Logic software on Intel 82598 10GE NIC and Core i7-4770k CPU
Algo-Logic GDN-Search: KVS in FPGA

Algo-Logic gateware on Nallatech P385 with Altera Stratix V A7 FPGA
Trends for Adding Storage around FPGAs

• **Six banks of memory controllers**
  - QDR SRAM
  - RLDRAM
  - DDR3, DDR4

• **64 lanes of SERDES**
  - SATA disk and Flash
  - Serial memories
    - Hybrid Memory Cube (HMC)
    - Mosys Bandwidth Engine (BE2, BE3)

• **New Memories**
  - 3D Xpoint with DDR4 Interface
  - Potential for Terabytes of Memory on each card
Implementation of KVS with Socket I/O, DPDK, and FPGA

- Benchmark same application
  - Key/Value Store (KVS)
- Running on the same PC
  - Intel i7-4770k CPU, 82598 NIC, and Altera Stratix V A7 FPGA
- With three different implementations
  - Socket I/O, DPDK, FPGA
KVS Hardware in Data Center Rack

- KVS in Software
- KVS in DPDK
- KVS in FPGA
- Rack of Search Servers
- Additional KVS Servers
- Provision Controller
- UPS Power

GDN-Classify
Associative Rule Match CAM

PHYs
MACs
Packets
Frames
ACLs
Target Queues

10G
40G

© 2015 Algo-Logic Systems Inc., All rights reserved.
Load Testing off the KVS Implementations

**GEN: Traffic Generator**
- Mellanox 10G NIC
- Intel 82599 10G NIC
- Mellanox 10G NIC

**KVS Implementations**
- Intel 82598 10G NIC
- Intel 82598 10G NIC
- Intel DPDK EAL
- Kernel Driver
- Process Message
- Process Message
- Process Message
- Nallatech P385 10G
Full-Rate Packet Processing (100% of TX Load)

Stratix V FPGA with EMSE-2 Drops no Packets

DPDK Drops Packets

Software Drops Packets
Sockets Power Consumption Profile
(10M Packets with 40 CSM Messages/Packet)

Voltage: 26V
Marginal Energy Consumption: 3.53 μJoul/CSM
Total Energy Consumption: 11.07 μJoul/CSM
DPDK Power Consumption Profile
(10M Packets with 40 CSM Messages/Packet)

Voltage: 28V
Marginal Energy Consumption: 2.83 μJoul/CSM
Total Energy Consumption: 6.59 μJoul/CSM
Latency Measurement with GDN-Classify

- Round trip latency of GDN Switch is deterministic (constant)
- Round trip latency of KVS Sever = Total Round trip Time – Round trip time with 10G loopback on GDN Switch
KVS Latency in FPGA, DPDK, and Sockets

Latency Comparison 100k packets, 1 OCSM per packet, 1k pps

- **KVS in FPGA**: Best Latency, No Jitter
- **KVS in DPDK**: Lowers Latency, Some Jitter
- **Sockets**: Average: 41.40μs
- **DPDK**: Average: 6.29μs
- **Altera Stratix V RTL**: Average: 0.467μs

Tighter Spread = Less Jitter

Lowest Latency = Faster Response
Measured Latency, Throughput, and Power Results

All Datapaths Summary

<table>
<thead>
<tr>
<th></th>
<th>Latency (µseconds)</th>
<th>Tested Throughput (CSMs/sec)</th>
<th>Power (µJoules/CSM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sockets</td>
<td>41.54</td>
<td>4.0</td>
<td>11</td>
</tr>
<tr>
<td>DPDK</td>
<td>6.434</td>
<td>16</td>
<td>6.6</td>
</tr>
<tr>
<td>RTL</td>
<td>0.467</td>
<td>15</td>
<td>0.52</td>
</tr>
</tbody>
</table>

GDN vs. Sockets 88x less 13x 21x less
GDN vs. DPDK 14x less 3.2x 13x less
Conclusions: Programmable Hardware in the Data Center

- **Lowers Latency**
  - 88x faster than Linux networking sockets
  - 14x faster than optimized DPDK (kernel bypass)

- **Increases Throughput (IOPs)**
  - 3x to 13x improvement in throughput
  - Lowers Capital Expenditures (CapEx)

- **Reduces Power**
  - 13x to 21x reduction in power
  - Reduces Operating Expenditures (OpEx)