Oracle SPARC – Software In Silicon

Stephen Phillips
Senior Director - Oracle Systems
SPARC @ Oracle
7 Processors in 6 Years

Including Software-in-Silicon
https://swisdev.oracle.com/

Security in Silicon:
- Silicon Secured Memory
- Cryptography Accelerators

SQL in Silicon:
- Data Analytics Accelerators
- In-line Decompression Engines

More to Come!

T3
16 x 2nd Gen Cores
6MB L2 Cache
1.7 GHz

T4
8 x 3rd Gen Cores
4MB L3 Cache
3.0 GHz

T5
16 x 3rd Gen Cores
8MB L3 Cache
3.6 GHz

M5
6 x 3rd Gen Cores
48MB L3 Cache
3.6 GHz

M6
12 x 3rd Gen Cores
48MB L3 Cache
3.6 GHz

M7
32 x 4th Gen Cores
64MB L3 Cache
4.1 GHz
DAX1

S7
8 x 4th Gen Cores
16MB L3 cache
4.2 GHz
DAX1
SPARC M7 Processor

- 32 SPARC Cores
  - Dynamically Threaded, 1 to 8 Threads Per Core
  - 4 SPARC S4 Cores per Core Cluster

- New Cache Organizations
  - Dedicated Level 1 Caches Per Core
  - Shared Level 2 Data and Instruction Caches
  - 64MB Partitioned and Shared L3 Cache

- 4 Memory Controller Units (MCU)
  - 16 DDR4-2133 Memory Channels
  - Up to 2 TB Physical Memory Per Processor
  - 170GB/s Aggregate Memory Bandwidth per Processor

- 32 Data Analytics Accelerator (DAX) Pipelines

- SMP Scalability from 1 to 16 Processors
  - Direct Connect and Switched Topologies
  - Up to 444GB/s Coherence Bandwidth per Processor

- PCIe 3.0 Support via I/O Controller ASIC
  - 4 x 8 Lane I/O Links @ 18.1 Gbps/Lane
  - 127GB/s Aggregate PCIe Bandwidth per Processor

- TSMC 20nm, 13 ML
SPARC Transformational Performance and Scale

SPARC M7 is the World’s Fastest Conventional Microprocessor

#1 SPECjEnterprise2010
1-chip
25,093.06 EjOPs

#1 SPECfp_rate2006
1-chip
832 peak

#1 SPECint_rate2006
1-chip
1,200 peak

#1 SAP-SD
2 processor
30,800 SAPs

And more...

Software in Silicon Adds
Revolutionary HW/SW Co-Engineering

Software in Silicon: Software Functions on Chip

(See Disclosure Slide)
SPARC M7: Software In Silicon Features

Security in Silicon:
- Silicon Secured Memory
- Cryptography Acceleration

Query Acceleration:
- Database Analytics Accelerator Engines

Capacity in Silicon:
- Decompression Engines
SPARC M7: Broadest Set Of Ciphers For All Your Apps

32 Crypto Accelerators per Processor

Clear Data In

Encrypted Data Out

32 Crypto Accelerators per Processor

Camellia

AES

SHA-512

SHA-384

SHA-256

SHA-244

SHA-236

SHA-1

MD5

ECC

DSA

DH

DES

3DES

CRC32c

RSA

SHA-512
## SPARC Leads in On-Chip Encryption Acceleration

<table>
<thead>
<tr>
<th>On-Chip Accelerators</th>
<th>SPARC M7</th>
<th>IBM Power7</th>
<th>IBM Power8</th>
<th>Intel Westmere / SandyBridge</th>
<th>Intel Haswell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric / Public Key Encryption</td>
<td>RSA, DH, DSA, ECC</td>
<td>None</td>
<td>RSA, ECC</td>
<td>RSA, ECC</td>
<td>RSA, ECC</td>
</tr>
<tr>
<td>Symmetric Key / Bulk Encryption</td>
<td>AES, DES, 3DES, Camellia</td>
<td>None</td>
<td>AES** (4 modes)</td>
<td>AES</td>
<td>AES</td>
</tr>
<tr>
<td>Message Digest / Hash Functions</td>
<td>CRC32c, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512</td>
<td>None</td>
<td>MD5, SHA-1, SHA-256, SHA-512</td>
<td>None</td>
<td>CRC32c</td>
</tr>
<tr>
<td>Random Number Generation</td>
<td>Supported</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

** IBM lacks AES hardware support for AES-CFB required by Oracle DB
Much Faster End-To-End Encryption

M7 Advantage Increases on Highest Security Ciphers

**AES 128-CBC:** Popular for Cloud, DB

- **Oracle M7 32 cores:** 83 GB/s
- **Intel X86 E5 v3 18 cores:** 22 GB/s
- **IBM Power8 6 cores:** 8 GB/s

**11X Faster vs. IBM Power**

**SHA 512-1024:** Important for Banking Operations

- **Oracle M7 32 cores:** 84 GB/s
- **Intel X86 E5 v3 18 cores:** 4.7 GB/s
- **IBM Power7+ 8 cores:** 1.4 GB/s

**18X Faster vs. X86**

**60X Faster vs. IBM Power**
Security Kernel SPARC Performance

AES: SPARC core is 2.2x – 2.3x faster than x86 E5 v4 core (AES-NI)
SHA: SPARC core is 7.25x – 7.5x faster than x86 E5 v4 core

AES-CFB per core
SPARC M7, S7 E5 v4 Broadwell

<table>
<thead>
<tr>
<th>CPU Type</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARC S7 4.27 Ghz</td>
<td>2.05 GB/s</td>
</tr>
<tr>
<td>SPARC M7 4.13 Ghz</td>
<td>1.98 GB/s</td>
</tr>
<tr>
<td>x86 E5 v4 2.2 GHz</td>
<td>0.89 GB/s</td>
</tr>
<tr>
<td>x86 E5 v3 2.3 GHz</td>
<td>0.89 GB/s</td>
</tr>
</tbody>
</table>

SHA512 per core
SPARC M7, S7 E5 v4 Broadwell

<table>
<thead>
<tr>
<th>CPU Type</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARC S7 4.27 GHZ</td>
<td>2.99 GB/s</td>
</tr>
<tr>
<td>SPARC M7 4.13 GHZ</td>
<td>2.90 GB/s</td>
</tr>
<tr>
<td>E5 v4 2.2 GHz AVX2</td>
<td>0.40 GB/s</td>
</tr>
<tr>
<td>E5 v3 2.3 GHz AVX2</td>
<td>0.30 GB/s</td>
</tr>
</tbody>
</table>

AES-256-CFB
Data at rest
DB, Cloud,..

SHA512-1024
Secure Checksum
Banking, UK online money...

"per core = (server performance)/(server core count)"
End-to-End Strong Encryption

Standard Java Application & Database Application Benchmark (SPECjEnterprise2010)

BENCHMARKED PERFORMANCE

**STRONG ENCRYPTION**

14,121.46 EjOps

**NO ENCRYPTION**

14,389.83 EjOps

Typical Application Performance Impact of Activating Strong Encryption

- Application DB Tablespace: Oracle Advanced Security Transparent Data Encryption (TDE) with 128-bit AES Cipher
- App & DB Server Network Connection: Oracle Network Data Encryption with JDBC Driver and RC4-128 Cipher
SPARC M7: Software In Silicon Features

**Security in Silicon:**
- Silicon Secured Memory
- Cryptography Acceleration

**Query Acceleration:**
- Database Analytics Accelerator Engines

**Capacity in Silicon:**
- Decompression Engines
In-Memory Query Acceleration

• Dedicated Analytics Accelerators built on chip
  – Independently process streams of compressed data placed in system memory
  – Like adding 32 additional specialized cores to chip
  – Up to 170 Billion rows per second

• Frees processor cores to run other applications, such as OLTP

• Decompresses data simultaneously to processing SQL functions
  – Like adding 64 additional specialized cores
M7 Query Accelerator Engine

- 32 In-Silicon Offload Engines
- Cores/Threads Operate Synchronous or Asynchronous to Offload Engines
- User Level Synchronization Through Shared Memory
- High Performance at Low Power
- 3x more Memory Bandwidth than x86
In-Memory Columnar Two-level Compression

- Efficient In Memory Columnar Table Scan
  - Contiguous storage per column

- Dictionary encoding huge compression
  - 50 US only need 6 bits (<1 byte) \textit{vs.} “South Dakota” needs 12 characters or 192 unicode bits = 32x smaller
  - Ozip/RLE compression is then applied on top of dictionary encoding

- Innovations:
  - \textit{Directly scan dictionary encoded data}
  - Ozip decompression without writing to memory
  - Save Min & Max for scan elimination
  - Can use dictionary for “featurization” of data for ML
Query Acceleration: Behind The Scenes
Equivalent of 32 extra cores plus 64 extra decompress cores
## Query Acceleration

### Data Analytics Accelerator (DAX) Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Use Cases</th>
</tr>
</thead>
</table>
| **SCAN** | Compare Operation  
Single Value & Range Comparisons | Array Search, Vector Search for Ranges/Values, Key Value Pairs, JSON/XML Processing |
| **SELECT** | Filtering Function  
Input: Input Vector and Bit Vector of Same Size  
Output: Elements which Bit Vector = 1 | Lookup Based on Given Bit Vector, Combine with Scan, Mapping, Spatial, Pattern Matching Algorithms |
| **TRANSLATE** | Lookup Function  
Input: Vector of Indices, Bit Table  
Output: Bit Table Value for Index | Complex Lookup |
| **EXTRACT** | Decompress Run-length Encoded, OZIP | Decompression – Compressed Memory, Java Heaps, Java Classes |
SQL & the Many Flavors of DSL – All potentials for DAX

SQL can be written in Domain Specific Languages (aka Language Integrated Queries)

- **SQL:**
  - `SELECT count(*) from person WHERE citizen.age > 18`

- **Apache Spark SQL (DSL):**
  - `val nvoters : Int = citizen.filter("citizen.age" > 18).count()`

- **Java Streams (DSL):**
  - `int nvoters = arrayList.parallelStream().filter(citizen-> citizen.olderThan(18)).count();`

- **Goldman Sachs Collections – Apache Eclipse (DSL):**
  - `int nvoters = fastList.count(citizen-> citizen.olderThan(18));`
Accessing DAX by S/W

- User Applications
  - High level API
    - libvector - Analytic / SQL API
    - libvector – Java / Python API
  - libdax
- DAX H/W
libdax: Low Level C Interface System Library

• libdax
  – Provides access to DAX units to accelerate in-memory data analysis operations
  – Hides hardware details and handles limitations transparently
  – Supports basic functions: scan, select, extract, translate, compress, logical operations
  – Thread safe library with probes to support trace and debug functionality
  – Supports multiple independent clients posting DAX requests in the same thread
  – Limitations: Supports only 64-bit applications
libvector

• It is a package that introduces a class called Vectors. Methods of Vector library use DAX under the hood to accelerate analytic applications

• Java Vectors are ordered collection of numbers or Strings
  – They are high level abstraction of arrays processed by DAX
  – All elements of Vectors are of same type and Vectors are immutable
  – They are similar to Lists in the sense that they can be accessed by their index
  – They are also similar to Java streams in the sense that they support many aggregate operations for example max, min, filtering, searching
  – There is a special type of Vector, which are called Index Vectors and contains integers representing index of another vector. They are typically generated as result of searching a vector for a specified value or a range of values.
DAX vs. Running on Core only
10x to 22x improvements depending on dictionary encoding size

DAX Enhancement vs. Core only
15M Lines, 12 Scan Cols

- 8 bits
- 16 bits
- 32 bits
DAX Integration with Java Streams

- Offload of IntStream filter, allMatch, anyMatch, nonMatch, map(ternary), count and toArray functions to DAX
- Can be leveraged with very minimal change in app (extra import stmt)
- [https://community.oracle.com/docs/DOC-1006352](https://community.oracle.com/docs/DOC-1006352)
- Available as standalone Library at: [https://swisdev.oracle.com/DAX](https://swisdev.oracle.com/DAX)

**Speedup with DAX**

10 Million Rows

<table>
<thead>
<tr>
<th>Function</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlier</td>
<td>3.6x</td>
</tr>
<tr>
<td>%tile</td>
<td>4.0x</td>
</tr>
<tr>
<td>Top-N</td>
<td>4.1x</td>
</tr>
<tr>
<td>Filter</td>
<td>10.7x</td>
</tr>
<tr>
<td>allMatch</td>
<td>21.8x</td>
</tr>
</tbody>
</table>
The Power of DAX for Analytics and Machine Learning
Community Driven Innovation

<table>
<thead>
<tr>
<th>Area</th>
<th>Analytics</th>
<th>DAX Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning</td>
<td>K-Nearest Neighbor</td>
<td>4x to 12x faster</td>
</tr>
<tr>
<td>Data Scanning</td>
<td>Top N In Memory</td>
<td>4x to 7x faster</td>
</tr>
<tr>
<td>Data Scanning</td>
<td>SQL on JSON data</td>
<td>4x to 5x faster</td>
</tr>
<tr>
<td>Streaming</td>
<td>Tweet Analysis</td>
<td>5x to 9x faster</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>K-Means</td>
<td>Up to 3x faster</td>
</tr>
</tbody>
</table>

• SPARC DAX Offload Acceleration with Open DAX API
  – Easily used in Java, Scala, Python, C, C++, ...
  – Applicable to a wide variety of algorithms
  – University research finding new creative uses of Open DAX API
Oracle Software In Silicon Developer Cloud
Free DAX Access Portal for Universities, Researchers, Customers and Partners

- Access Solaris Zones with Software in Silicon and DAX technology
- Open APIs, libraries, man pages, headers
- Example Apache Spark integration
- Use cases and code examples
- Prebuilt templates to extend and customize
- 30GB storage to upload your test data and applications
- Simple Online Click-thru license agreement

Available now at http://SWiSdev.Oracle.com/DAX
Resources

• Software in Silicon Community: https://community.oracle.com/community/softwareinsilicon/overview

• Software in Silicon Developer Cloud: http://SWiSdev.Oracle.com/DAX
Integrated Cloud
Applications & Platform Services
What is Java Streams?

- Java 8 API for SQL style analytics processing on in-memory source data residing as collections
- Simplicity of expression
  - Functional programming style, including lambdas
  - More efficient expression of pipelined data processing stages
- Automatic parallelization leveraging the data parallelism transparently
- Any java application with a large collection with millions of elements that needs to be queried is a candidate for using Streams
  - Real time analytics on existing in-memory heap data without moving it
  - Get this done almost for free without using core cycles when offloaded to DAX
  - Extreme consolidation opportunity when leveraging offload to DAX