DATA-CENTRIC COMPUTER ARCHITECTURE

Pankaj Mehra, Founder and CEO

July 3, 2017
Data-Centric Computer Architecture

1. Elements of Infrastructure: Bits, Cores, and Fabrics
2. Data Sources, Data Varieties, and Data Growth
3. Data Lifecycle and Business Value of Information
4. Toward a Memory-Centric Architecture
5. iMemory Prototype
Bits, Cores & Fabrics: the elements of infrastructure
Data Center Infrastructure in context

Key Themes
• Data centricity
• New memories
Bits, Cores & Fabrics
The foundation of infrastructure

Universal System Concepts

<table>
<thead>
<tr>
<th>Services</th>
<th>Information</th>
<th>Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>mServices</td>
<td>Objects</td>
<td>Protocols</td>
</tr>
<tr>
<td>APIs</td>
<td>Metadata</td>
<td>Topologies</td>
</tr>
<tr>
<td>Software</td>
<td>Data</td>
<td>Routes</td>
</tr>
<tr>
<td>OS</td>
<td>Metabits</td>
<td>End points</td>
</tr>
<tr>
<td>Cores</td>
<td>Bits</td>
<td>Fabrics</td>
</tr>
<tr>
<td>ALUs</td>
<td>Caches</td>
<td>I/O Ports</td>
</tr>
<tr>
<td>Firmware</td>
<td>Data Paths</td>
<td>Switches</td>
</tr>
<tr>
<td>control</td>
<td>state</td>
<td>flow</td>
</tr>
</tbody>
</table>

Universal Hardware Concepts
**Systems, Services, Devices**

Bit primacy historically at device level only

![Diagram showing levels of abstraction with Compute, Network, and Data primacies]

- **Core Levels of Abstraction**
  - Compute primacy
  - Network primacy
  - Data primacy

- **System**
  - CORES
  - FABRICS
  - BITS

- **Service**
  - FABRICS
  - CORES
  - BITS

- **Device**
  - BITS
  - FABRICS
  - CORES
The quest for data primacy
Follow the bits

Graphic courtesy: ARM

We used to call them Computer Centers!
Data at the Center: Why?
Sources, Varieties, Growth
Typical One-Stop Online Portfolio
The perfect user data trap
The Cloud: What User Bits Vanish Into

The Cloud: Where bits gather context
THE RIGHT INFORMATION AT THE RIGHT TIME
... IN THE RIGHT CONTEXT

Business value of information

Raw, new information

Contextualized information

“right time” benefit

Net gain from contextualization, standing queries

Net gain from batch ML, ad hoc queries

Time since inception

Expectedness

SLA
Typical Storage Abstraction Cake

Often a shared utility owned by an Infrastructure & Ops team for internal properties + 1000s of ecosystem partners + IaaS customers?

Not uncommon to find multiple EBs across 100Ks servers
User Data

• Generally,
  – Never-say-no attitude!
  – “Free & Unlimited” BYOC
  – 40+% growth in photo and video tier
    • Machine learning based information extraction
  – Users revealing each other’s context in social graphs and CCOs
    • Advertising gold!
Logging, and not just transactions
The root of all data collection

<table>
<thead>
<tr>
<th>TLOG</th>
<th>ALOG</th>
<th>ELOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTION LOGGING</td>
<td>APPLICATION LOGGING</td>
<td>LOG EVERYTHING</td>
</tr>
<tr>
<td><strong>Business Critical Tx in Operational Data Stores</strong></td>
<td><strong>SIEM (ArcSight), Kissmetrics (SaaS) and Google Analytics, spur a wave of app logging</strong></td>
<td><strong>The user is the product</strong></td>
</tr>
<tr>
<td><strong>Paid transactions ($0.10/tx)</strong> → <strong>Free Transactions</strong> ($0)**</td>
<td><strong>5 EB in MSFT Cosmos!</strong></td>
<td><strong>Every read becomes a write</strong></td>
</tr>
<tr>
<td><strong>Blockchain (FSI, pharma, ...) for Distributed Ledger</strong></td>
<td></td>
<td><strong>PBs/day pour in from phones, fixed cameras, cars (GM), travelers, ...</strong></td>
</tr>
</tbody>
</table>
Lifecycles and Business Value of Information
Information Lifecycle Management

*Driven more by protection and retention than by cost*

- **Operational**
  - frequently updated during 72 hours after creation

- **Transitional**
  - infrequently updated
  - converted to business record format

- **Archival**
  - static (rarely accessed)
  - subject to long-term records management
Copy Data Management

- Production databases
- Historical archive (long-term retention)
- Extract, transform, load
- Test and development
- Archive (or delete)
- Individual data marts (decision support)
- Information Quality
- Extract Correlate Contextualize
Toward MCA
Memory-centric
Computer Architecture
Shipping computation to the data

**Power**
Reduction in data movement count and distance

**Performance**
Parallelism, Bandwidth, and Latency

**Cost**
Low gate count embedded cores with future open ISA and tools

Works best when simple expressions computed against large number of data records
iMemory: Bits meet Cores
Beyond Devices: Data Primacy as the ticket to systems

Domain Specific Language optimizers are key

Level of Abstraction

Database optimizer

File system

Block layer

DEVICE

Opaque blocks
I/O expansion fabric
Embedded cores for soft logic

SYSTEM

Coherent accelerator fabric
Application format bits
Application acceleration cores
a new tier in the Data Center
where
Data can be Big \textit{and} Fast
# Market Segments and Currently Architected Tiers

<table>
<thead>
<tr>
<th>Compute Tier</th>
<th>Memory-storage convergence in full swing.</th>
<th>All about highest capacity at the lowest cost.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC</td>
<td>Several monumental shifts driven by the need to query petabytes in real time</td>
<td>Revolutionary shifts driven by the need to retain data for 20-100 years</td>
</tr>
</tbody>
</table>
| Hyperscale Server | 1. Hana, a database without an I/O stack  
2. Spark and ML, placing analytics in focus  
3. Petabytes held in DRAM by memcached and redis  
4. Kafka, a pub-sub system without any storage I/O  
5. pmemobj, ext4-DAX maturing | 1. Unified scale-out filesystems for block-file-object  
2. Spark and ML in Compute Tier highlight the need for bandwidth over latency in archive tier  
3. Encryption, Access Control, Global deployment and wide-area optimization of data synch are key |
| Enterprise Server | |  |
| Enterprise Storage, Converged | | Revolutionary shifts driven by the need to retain data for 20-100 years |

1. Sustained investment in optical and DNA storage to create an alternative to tape below HDD tier
**Confluence of forces driving a memory-centric tier**

<table>
<thead>
<tr>
<th>Compute Tier</th>
<th>HPC</th>
<th>Hyperscale Server</th>
<th>Enterprise Server</th>
<th>Enterprise Storage, Converged</th>
</tr>
</thead>
</table>
| **Memory Tier** | Memory and storage converge  
Accessing big data using I/O memory semantic | | | |
| | Memory disaggregates across fabric  
Provision working memory for peak median usage | | | |
| | Memory-centric addressing  
Bulk of processing happens near the CPU memory | | | |
| **Archive Tier** | | | | |


Query execution dominated by scan bandwidth

TPCH Query 12
87% of total cost at the leaf of the query plan

Scan and seek cost relative to total query cost

<table>
<thead>
<tr>
<th>Number of TPCH queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;20%</td>
</tr>
<tr>
<td>20%-40%</td>
</tr>
<tr>
<td>40%-60%</td>
</tr>
<tr>
<td>60%-80%</td>
</tr>
<tr>
<td>80%-100%</td>
</tr>
</tbody>
</table>

Most queries dominated by scan and seek cost

Source: http://www.qdpma.com/tpch/TPCH100_Query_plans.html
The Bandwidth Mismatch

Moving data to computation

128 NAND Dies

16 packages

BW 128GB/s

BW 64GB/s

Storage Interface

eSSD Controller

BW 8GB/s

BW 7GB/s

BW 6.4GB/s

1 FM

1 FE

I/O Interface

Data trickles out to host
Possible Placements of Compute Cores in iMemory

Benefit
- $/perf and W/ perf
- Greater bandwidth and lower latency between a computation and its data

Challenge
- Lack of ECC and possibly FTL functionality

Conventional placement of compute cores
Core integrated with controller
Core integrated in die or package
Challenges of Core Placement in SSDs

Exploiting memory bandwidth requires rethinking memory management.

Lowest Bandwidth
Physical addressing only
Uncorrected memory
No direct access to DRAM
Cost, power and fab friendliness

Highest Bandwidth

I/O Interface

SSD Controller

Front End (FE)
Inter chip Interface

Flash Manager (FM)

NVM Interface

Interposer

NVM Package(s)

Compute Cores

DRAM

Inter-chip Interface

Lowest Bandwidth

Physical addressing only
Uncorrected memory
No direct access to DRAM
Cost, power and fab friendliness

Highest Bandwidth
**Cores near memory**

**How many cores?**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan bandwidth</td>
<td>130 GB/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average record size</td>
<td>1000 B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record scan bandwidth</td>
<td>130 M records/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computation (Instr/Record)</td>
<td>10</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>Total processing power required (MIPS)</td>
<td>1300</td>
<td>13000</td>
<td>130000</td>
</tr>
<tr>
<td>Processing power per core</td>
<td>800 MIPS (say)</td>
<td></td>
<td></td>
</tr>
<tr>
<td># of cores</td>
<td>1.6</td>
<td>16.2</td>
<td>162.5</td>
</tr>
</tbody>
</table>

**Another metric**

MIPS/Scan bandwidth -> Processing power required per unit of available scan bandwidth

For example, in the case above, the system requires 10, 100 or 1000 MIPS per GB/s

**Need low gate count, cache-less cores tuned for data-intensive workloads**
iMemory Architecture
Achieving 100GB/s processing rate

• Fast Read Path:
  – Judicious core placements enable iMemory to exploit internal read bandwidth and provide order of magnitude processing bandwidth.
  – iMemory exposes cores, translations, and data placement via APIs to database optimizers.

• Auto targeting and Just-In-Time (JIT) enabled data-layer optimizers
  – Generated (not handwritten) code efficiently targets 10s-100s of DPU cores in iMemory.
  – JIT compilation improves system efficiency with optimal targeting of iMemory.

• Application aware ECC to enable high throughput decoding
  – ECC engine aware of logical and physical database schemata (record size, column count and sizes, row or column order).
  – Decoder informed on a query-by-query basis about table fields used, projected or ignored.
Scan Bandwidth
The road to 32.5 GB/s per TB

Key Technology Enablers: Controller enhancement, Packaging, Die Enhancement
Aligning with Industry and Academic Initiatives
Analytics Infrastructure Scaling Trends

*If it does not scale, it will fail*

**UP**
- Scaling
- Driven by limited scale-out of Oracle RAC and even Hana for handling transactions and mixed workloads

**Scaling OUT**
- From analytics under *add as you grow* Hadoop instances and *more iron at Spark*, now transactions in Spanner, Cockroach

**Scaling IN**
- Scaling out without sprawling out. e.g.,
  - RackOut (SoCC’16) uses RDMA to disaggregated data shared within rack.
  - HPC and ML, memcache, etc. benefit 36-88% from using fewer nodes with MORE MEMORY
Scaling Down
an attractive alternative

• Makes sense for lightweight compute and moderate to high bandwidths
  – Key-Value Stores, for instance!

• Delivers best cost when integrated with semiconductor memories such as flash and perhaps SCM

• Integrated with SCM, it could give GPUs, FPGAs, and von Neumann configurations with big memory a run for the money
  – HANA and IMDBs, for instance

• REQUIRES
  – Investment in optimizers
  – Low power, low cost interconnects
  – Silicon integration of cores with memory
Anthropomorphic Workloads

Hard: Logic and memory on same die
Hard: Cores routable using 3-4 metal layers
Lack of killer apps and optimization ecosystem

PIM cores
Optimizers, JIT compilers, x-compilers

Compute to Memory

Memory to Compute

Communication Energy

- 64-bit DP 20 pJ
- 256-bit buses 25 pJ
- 256-bit access 8 KB SRAM 50 pJ
- DRAM RD/W 500 pJ
- Efficient off-chip link
- 1 nJ

20 mm
DATA GRAVITY

Also applies to data versus data

- How we process ML training
- How we should process ML training
- Feed the hungry GPUs
Optimizing Data Placement
A key optimization to develop 3-5 years out

• Pressures to get to even lower power
  – Long tail of extreme personalization + Privacy concerns/laws ⇒ Learning in the field
  – Evolving world requires always (online) learning algorithms

• Pressures to get to even higher performance
  – Ad hoc queries against petabytes of data in real time (this talk)
  – Long standing queries (context aware computing)
  – HTAP (Analytics and context mediated transactions)

• Compilers and runtimes do not even recognize this as a problem yet
  – Yet, leaders in industry and academia believe this is one of the most important problems

• E.g. carefully placing matrices and vectors in such a way that dot products, matvecs, gemms, and tensor products can be computed w/o data movement

• Now, add memristive logic
THE ULTIMATE QUESTION BEFORE COMPUTER ARCHITECTS

Is this also the von Neumann vs non-von-Neumann question?

Compute that remembers

Memory that computes

© Sören Boyn / CNRS/Thales physics joint research unit.