Erasure Code Offload for Distributed Software Defined Storage

Dror Goldenberg

7th Annual TCE Conference – June 21-22/2017
Enable the Most Resilient Cloud Infrastructure

This cloud helps Real Madrid open one stadium to 450 million fans.
Cloud Infrastructure – The Software Defined Data Center

- Resource virtualization
- Efficient services
  - VM, Containers
  - uServices scale out arch
  - DevOps automation
- Tenant isolation & security
- Visibility and telemetry
Software Defined Storage

- Software Defined Datacenter
  - Software Defined Networks
  - Software Defined Storage

- Scale Out Advantage

- Scale-out Storage Examples
  - Ceph, Swift, Gluster and many more
Storage Feeds and Speeds Drive the Need for Efficient Network

Network BW is critical to enable Scale Out

Low latency is imperative for Software Defined Storage
“To make storage cheaper we use lots more network!
How do we make Azure Storage scale? RoCE (RDMA over Converged Ethernet) enabled at 40GbE for Windows Azure Storage, achieving massive COGS savings”

Microsoft Keynote at Open Networking Summit 2014 on RDMA
RoCE enables Virtualized Infrastructure without Compromise

Without RDMA

With RDMA

6X The Throughput
Compared to iWARP

<1usec Latency
VM to VM Communication

<2% CPU Utilization
Delivering I/O at 100Gb/s

Performance
Higher Throughput & IOPS

Scalability
Reduce Overhead

Efficiency
Lower CPU Utilization

Achieve Tomorrow’s Cloud Efficiency Today
Data Availability

- Standalone
- Cluster
- Hot swap
- RAID 0
- RAID 1
- RAID 5
- RAID 0+1

https://www.nextofwindows.com/raid-terms-explained-in-water-cooler
Ensuring Data Availability

Data Replication

D0
D1
D2
D3

D0
D1
D2
D3

D0
D1
D2
D3

Erasure Coding

D0
D1
D2
D3

P0
P1
P2
P3
Ensuring Data Availability

Data Replication

Erasure Coding
Ensuring Data Availability - Summary

**Data Replication**
- Capacity: 3x data typical
- Resilient to 2 failures

**Erasure Coding**
- Capacity: 1.4x data typical
- Better failure resilience
  - e.g. 10+4: 1.4x capacity, 4 failures
- CPU & network hungry
- Longer & data intensive rebuild
- Partial update requires read
Erasure Codes Theory

- **K data + M parity = N total**
  - Tolerates up to M failures
  - Total overhead N/K

- **Systematic Codes** – encoded data contains original data

- **Maximal Distance Separable (MDS)**
  - Can survive any M erasures

- **Reed Solomon Coding**
  - Many other codes exist: RAID 6, XOR, Pyramid, LRC, …
Reed Solomon Encoding

Generating parity is matrix multiplication

\[ B \cdot * = \{D, P\} \]
Reed Solomon Decoding (1/2)

- **In case of failures**
  - Matrix is reduced
  - Calculate Inverse matrix
  - Recover Data
  - Recover Parity

\[ D = \{D,P\}' \]
Reed Solomon Decoding (2/2)

- Recovery - how?
  - $B' * D = S$
  - $B'^{-1} * B' * D = B'^{-1} * S$
  - $D = B'^{-1} * S$
  - $\{D, P\} = B * D$
  
  $S = \text{Survivors vector}$
Algebra Magic (1/2) – Matrix Generation

- All submatrices must be invertible
- Vandermonde matrix is used as baseline
- Derived matrix through elementary operations
Algebra Magic (2/2) – Arithmetic OPs

- Needed: finite field, multiplicative inverse

- Operation done over Galois Field – GF($2^w$)
  - Sum is XOR operation
  - Multiplication is more complicated…
    - Numbers are multiplied and then divided by an irreducible polynomial

- N must be $\leq 2^w$
High CPU Demand

- Matrix multiplication compute needed
  - $O(k*m)$ multiply-add operations

- Cache & TLB intensive (large data sets)
Erasure Code History

- Évariste Galois 1811-1832
- Alexandre-Théophile Vandermonde 1735 – 1796
- Irving S. Reed 1923-2012
- Gustave Solomon 1930-1996
Network Traffic – Sunny Day Scenario (Replication)

**READ Operation**
- Client
- OSD
- OSD
- OSD

- Read
- Read Reply

**No extra cluster network traffic**

**WRITE Operation**
- Client
- OSD
- OSD
- OSD

- Write
- Replications
- Write Ack

**Typically 2x cluster network traffic (N-1) x**
Network Traffic – Sunny Day Scenario (Erasure Coding)

READ Operation
- Client sends Read request
- OSDs read shards
- Read Reply is decoded
- 

\[ \text{~1x cluster network traffic} \]

\[ \frac{(k-1)}{k} \cdot x \]

Typically 

\[ \text{Typically ~1.4x cluster network traffic} \]

\[ \frac{(k+m-1)}{k} \cdot x \]

WRITE Operation
- Client sends Write request
- OSDs encode shards
- Write Ack
- 

\[ \text{Typically ~1.4x cluster network traffic} \]

\[ \frac{(k+m-1)}{k} \cdot x \]
Network Traffic – Recovery (Replication)

- Example - Time to recover
  - Net networking time to move data
  - 20TB system @40GE 1.1hrs
  - 200TB system @40GE 11.1hrs

§ Similar flows for scrubbing
Network Traffic – Recovery (Erasure Coding)

- Example - Time to recover (10+4)
  - Net networking time to move data
  - 20TB system @40GE 14.4hrs
  - 200TB system @40GE 144.4hrs

- Similar flows for scrubbing

Tradeoff: recovery time vs storage efficiency
Typical Workflow - Long Write (Encode)

write(*data)

D

Long Write

Data stripe

Encode

Send

Calculation

B*D=P
Typical Workflow - Reconstruct (Decode)

- Decode (*data)
  - Retrieve Available Elements
  - Decode (recover lost data)
  - Encode (calculate parity)
  - Send
Onload vs Offload

- **Onload**
  - Computation all done on CPU
  - CPU at 100% during calculation
  - Cache/TLB pollution
  - Example: ISA-L

- **Offload**
  - Computation all done in accelerator
  - CPU at 0% during calculation
  - Cache/TLB unaffected
  - Example: ec_offload APIs
Synchronous vs Asynchronous

- **Synchronous APIs block until operation completes**
  - Suitable to the current common API semantics
  - CPU computes – runs to completion
  - Onload operation – faster ISA, but still 100% CPU utilization

- **Asynchronous enables CPU to focus on computation**
  - Suitable to offload semantics
  - Operation starts, completion reported upon callback
  - Can implement Synchronous using Asynchronous calls
    - Easy fit for today’s integrations
## EC Offload APIs Cheat Sheet

<table>
<thead>
<tr>
<th></th>
<th><strong>Synchronous</strong></th>
<th><strong>Asynchronous</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialization</strong></td>
<td><code>ibv_exp_alloc_ec_calc()</code></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><code>ibv_exp_dealloc_ec_calc()</code></td>
<td></td>
</tr>
<tr>
<td><strong>Encode</strong></td>
<td><code>ibv_exp_ec_encode_sync()</code></td>
<td><code>ibv_exp_ec_encode_async()</code></td>
</tr>
<tr>
<td></td>
<td><code>ibv_exp_ec_decode_sync()</code></td>
<td><code>ibv_exp_ec_decode_async()</code></td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td><code>ibv_exp_ec_update_sync()</code></td>
<td><code>ibv_exp_ec_update_async()</code></td>
</tr>
<tr>
<td><strong>Encode &amp; Send</strong></td>
<td>-</td>
<td><code>ibv_exp_ec_encode_send()</code></td>
</tr>
<tr>
<td><strong>Book keeping</strong></td>
<td>-</td>
<td><code>ibv_exp_ec_poll()</code></td>
</tr>
</tbody>
</table>
Encoding Performance (Single Core)

K=10, M=4 Encoding Performance

- EC Offload CPU (%)
- ISA-L CPU (%)
- EC Offload BW (MB/s)
- ISA-L BW (MB/s)
- 100Gb/s

4x Faster
50x Lower CPU%
Encoding Performance (Single Core) - ARM

K=10, M=4 Encoding Performance - ARM

Vector Size [KB]

- EC Offload CPU(%)
- ARM CPU(%)
- EC Offload BW(MB/s)
- ARM-BW(MB/s)
- 100Gb/s
Summary

- Cloud infrastructure requires efficient and scalable storage
- Software Defined Storage drives scale out storage
- Erasure Codes enable data availability at lower capacity
  - Tradeoff: CPU & Network intensive, complexity
- Erasure Codes offload offers
  - Better performance
  - Lower cost
- Library available today
  - Integration to storage systems underway
- Challenges
  - Efficient use of asynchronous acceleration
  - Combining Erasure Codes offload and networking
Thank You