Blue Gene Active Storage Architecture

and the

Non-Volatile Memory Verbs Provider Interface

Blake G. Fitch (bgf@us.ibm.com)
The Active Storage Project

• The Active Storage Project is aimed at
  • enabling close integration of emerging solid-state storage technologies with high performance networks and integrated processing capability
  • and exploring the application and middleware opportunities presented by such systems
  • anticipating future scalable systems comprised of very dense Storage Class Memories (SCM) with fully integrated processing and network capability

• Specifically, we extend the Blue Gene/Q architecture by integrating SCM (currently Flash) into the node

• This presentation will give a very quick overview of:
  • the Blue Gene Active Storage extensions to BG/Q
  • a key Storage Class Memory (including Flash) access method
  • and some early performance results
Blue Gene/Q: A Quick Review

1. Chip: 16+2 $\lambda P$ cores

2. Single Chip Module

3. Compute card:
   One chip module,
   16 GB DDR3 Memory,
   Heat Spreader for H$_2$O Cooling

4. Node Card:
   32 Compute Cards,
   Optical Modules, Link Chips; 5D Torus

5a. Midplane:
   16 Node Cards

5b. IO drawer:
   8 IO cards w/16 GB
   8 PCIe Gen2 x8 slots
   3D I/O torus

6. Rack: 2 Midplanes
   SC13 NVM in HPC BoF

7. System:
   96 racks, 20PF/s

- Sustained single node perf: 10x P, 20x L
- MF/Watt: (6x) P, (10x) L (~2GF/W, Green 500 criteria)
- Software and hardware support for programming models for exploitation of node hardware concurrency
Blue Gene Active Storage (BGAS) Concept System

“How to” guide:
• Remove 512 of 1024 BG/Q compute nodes in rack – to make room for solid state storage
• Integrate 512 Solid State (Flash+) Storage Cards in BG/Q compute node form factor

Standard BG/Q Compute Fabric

Node card
16 BQC + 16 PCI Flash cards

BGAS Rack Targets

Flash Storage 2012 Targets
Raw Capacity 2 TB
I/O Bandwidth 2 GB/s
IOPS 200 K

BGAS Rack Targets

Nodes 512
Raw Capacity 1 PB
I/O Bandwidth 1 TB/s
Random IOPS 100 Million
Compute Power 104 TF
Network Bisect. 512 GB/s
External 10GbE 512

System Software Environment
- Linux OS enabling storage + embedded compute
- OFED RDMA & TCP/IP over BG/Q Torus – failure resilient
- Standard middleware – GPFS, DB2, MapReduce, Streams

Active Storage Target Applications
- Parallel File and Object Storage Systems
- Graph, Join, Sort, order-by, group-by, MR, aggregation
- Application specific storage interface

Key architectural balance point: All-to-all throughput roughly equivalent to Flash throughput

© 2012 IBM Corporation
The Blue Gene Q Integrates processors, memory and networking logic into a single chip. Each BQC supports a PCIe 2.0 x8 interface.

- 360 mm² Cu-45 technology (SOI)
- 16 user + 1 service PPC processors
  - all processors are symmetric
  - each 4-way multi-threaded
  - 64 bits
  - 1.6 GHz
  - L1 I/D cache = 16kB/16kB
  - L1 prefetch engines
  - each processor has Quad FPU
  - peak performance 204.8 GFLOPS @ 55 W
- Central shared L2 cache: 32 MB
  - eDRAM
  - multiversioned cache — supports transactional memory, speculative execution.
  - supports scalable atomic operations
- Dual memory controller
  - 16 GB external DDR3 memory
  - 42.6 GB/s DDR3 bandwidth (1.333 GHz DDR3)
    (2 channels each with chip kill protection)
- Chip-to-chip networking
  - 5D Torus topology + external link
    → 5 x 2 + 1 high speed serial links
  - each 2 GB/s send + 2 GB/s receive
  - DMA, remote put/get, collective operations
- External IO -- when used as IO chip
  - 3D Torus
  - PCIe Gen2 x8 interface (4 GB/s Tx + 4 GB/s Rx)
- Hybrid Solid State Storage Device
  - PCIe 2.0 x8
  - 2x 10Gb Ethernet
  - 2 TB SLC (raw)
  - 8 GB DRAM

The Blue Gene Active Storage Node

The Blue Gene/Q Integrates processors, memory and networking logic into a single chip. Each BQC supports a PCIe 2.0 x8 interface.
BGAS Platform Performance – Emulated Storage Class Memory

- Software Environment
  - Linux Operating System
  - OFED RDMA on BG/Q Torus Network
  - Fraction of 16 GB DRAM used to emulate Flash storage (RamDisk) on each node
  - GPFS uses emulated Flash to create a global shared file system

- Tests
  - IOR Standard Benchmark
    - all nodes do large contiguous writes – tests A2A BW internal to GPFS
  - All-to-all OFED RDMA verbs interface
  - MPI All-to-all in BG/Q product environment
    - a light weight compute node kernel (CNK)

- Results
  - IOR used to benchmark GPFS
    - 512 nodes → 0.8 TB/s bandwidth to emulated storage
  - Linux Network software efficiency for all-to-all
    - BG/Q MPI (CNK baseline): 95%
    - OFED RDMA verbs 80%
    - GPFS IOR 40% - 50%
Blue Gene Active Storage – 64 Node Research Prototype

Hybrid Scalable Solid State Storage
PCIe 2.0 x8 FHHL Flash device

Blue Gene Active Storage 64 Node Prototype (4Q12)
- IBM 19 inch rack w/ power, BG/Q clock, etc
- PowerPC Service Node
- 8 IO Drawers
- 3D Torus (4x4x4)
- System specification targets:
  - 64 BG/Q Nodes – 12.8TF
  - 128 TB Flash Capacity (SLC) raw
  - 128 GB/s Flash I/O Bandwidth
  - 128 GB/s Network Bisection Bandwidth
  - 4 GB/s Per node All-to-all Capability
  - 128x10GbE External Connectivity
  - 256 GB/s I/O Link Bandwidth to BG/Q Compute Nodes
A BG/Q I/O Drawer With Eight HS4s Installed
The Non-Volatile Memory Verbs Provider
Typical application code path to storage: 20,000 Instructions!

- Moving Data Between Tier1 & Tier2 (DRAM and Flash)....
- Requires “Disk I/O”
- Syscall, pinning, DMA setup, Interrupt, etc.

**Diagram:**

- Application
  - Read/Write
  - Syscall

- FileSystem
  - Strategy()
  - iodone()

- LVM
  - Strategy()
  - iodone()

- Disk & Adapter DD
  - Pin buffers, Translate, Map DMA, Start I/O
  - Interrupt, unmap, unpin, iodone scheduling

**Workload**

- CPUs
- L1 Cache
- L2 Cache
- System Memory (Cache)
- PCIe Flash SSD (Cache)

**Spinning Disk Storage**

**Media Access Time**

- Actual
  - 1 ns
  - 10 ns
  - 60 ns
  - 50 µs (read)
  - 1 ms (write)
  - 5 ms

- Scale
  - 1 ns
  - 10 ns
  - 100 ns
  - 10⁴ ns
  - 10⁵ ns
  - 10⁶ ns
  - 10⁷ ns
The OFED Stack For Network AND Non-Volatile Memory RDMA

Adapted from: www.openfabrics.org/ofed-for-linux-ofed-for-windows/ofed-overview.html
Non-Volatile Memory Verbs Provider (NVP) – Use Model

- Standard OFED interfaces are used to
  - Open an OFED device
  - Create a protection domain (PD)
  - Create a create a queue pair (QP) and transition it to RTS
  - All NVP QPs connect to an Embedded Storage Peer (ESP)
  - NVP EPS handles access and control flow
  - NVP EPS uses HS4 HAL to issue device level requests
  - NVP enables zero-copy on data path (card ↔ user space buffer)

- NVP user opens a Flash partition using an RPC via RDMA SEND/RECV
  - RDMA SEND a message to an NVP embedded storage peer
  - Embedded Storage Peer receives the ‘open’ request
  - Open accesses a logical partition (above the FTL) of the Flash device
  - The NVP considers the partition to be a registered memory region of the embedded storage peer and creates a memory region ‘key’
  - The embedded storage peer uses RDMA SEND to return the Flash memory region tag to the NVP client

- The NVP user registers a memory region with OFED
  - This is registration is via normal OFED ibv_reg_mr() which returns a key

- Client does OFED WRITE and READ operations on the NVM QP do data transfers to/from flash
  - system call stack bypass and device DMA data placement
  - ibv_post_send() used with a work request containing the local and remote memory keys
  - RDMA READ and WRITEs are byte addressable per OFED semantics
  - Fully asynchronous work queue based interface per OFED semantics
NVP Prototype Early Results

First simple tests:
- \( \mu \)-bench: low-level access inside driver, full page (8k) physical addressing (upper boundary of what's possible from host side of PCIe)
- FIO: single process, standard linux raw block device, full page access
- NVP: single process, single QP, random read test, full page access (*write measured in different env. only)

<table>
<thead>
<tr>
<th></th>
<th>Read [MiB/s]</th>
<th>Write [MiB/s]</th>
<th>Read [kIOPS]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu )-bench</td>
<td>2450 (8k)</td>
<td>1820 (8k)</td>
<td>610</td>
</tr>
<tr>
<td>FIO</td>
<td>550 (4k)</td>
<td>181 (4k)</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>1303 (8k)</td>
<td>264 (8k)</td>
<td></td>
</tr>
<tr>
<td>NVP</td>
<td>2070 (4k)</td>
<td>n/a</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>2410 (8k)</td>
<td>*1450 (8k)</td>
<td></td>
</tr>
</tbody>
</table>
NVP Clients

Workload Optimized Flash Management

Standard POSIX File System I/O

Scalable Key/Value Store (SKV)

Direct NVP Applications and Middleware

High Level Test and Diagnostics

Kernel Space

HAL

User Space

Linux File Systems (Ext4, GPFS, etc)

Verbs Block Dev

NVP Client

Non-Volatile Memory Verbs Provider (NVP)

Embedded Storage Peer (a thread)

HS4 Hardware Abstraction Layer

PCIe 2.0 x8

FPGA

Flash

DRAM

10 GbE
Non-volatile Memory Verbs Provider (NVP) Overview

Fast Path:
- Zero copy rd/wr
- Byte addressable
- Kernel & User Space
- Modify middleware to use – e.g. GPFS
- HDF5 -> MPI-IO -> SKV provides a direct path from application to NVMem

Legacy path