

A dark blue background with abstract light blue and white streaks and glowing points, suggesting a network or data flow. The text is positioned on the right side of this graphic.

BlueArc Mercury™ Architecture

The Next Evolution in Network Storage

Abstract

BlueArc Mercury delivers a unique technology framework that combines a high performance hybrid-core hardware platform with the SiliconFS hardware accelerated file system as well as advanced software features such as data protection and data management. Mercury's open architecture excels across a variety of application environments, including general-purpose file systems, database, messaging and online fixed content.

This white paper discusses the implementation and advantages of the BlueArc Mercury Hybrid-Core hardware platform and the fundamentals of the BlueArc operating system design.

Table of Contents

Introduction 2

Advantages of a Hybrid-Core Architecture 3

A Solid Hardware Foundation 4

 Mercury Chassis 4

 Filesystem Interfaces 4

 Management Interfaces 5

 Mercury Filesystem Board (MFB) 5

 LVDS Fastpath 5

 Field Programmable Gate Arrays (FPGA) 6

 Memory (Buffers and Caches) 7

 Network Interface FPGA (NI) 8

 Data Movement (TFL) 8

 Disk Interface (DI) 10

 SiliconFS™ (WFS) 10

 Mercury Management Board (MMB) 12

Fundamentals of the Mercury Operating System 12

 BlueArcOS and Linux Incorporated (BALI) 13

 Platform API (PAPI) 13

 System Management Unit (SMU) 14

 Monitoring Processes 15

Beyond the Server – Storage System Architecture 15

Life of a Packet 16

 Write Example 16

 Read Example 17

Conclusion 18

Introduction

For more than a decade, BlueArc has continuously proven itself as the vendor of choice for customers that demand reliable, high-speed, highly scalable network storage solutions to accelerate their time to market and meet or exceed revenue goals. BlueArc servers are at the core of blockbuster movies using the latest in 2D and 3D special effects, providing high-speed data access to render farms with thousands of computing nodes. Large universities employ BlueArc systems to deliver storage capacity to thousands of researchers and students simultaneously, providing IT administrators an easy to manage storage environment that will handle any type of workload a researcher may throw at it. Internet service providers and cloud service providers deploy BlueArc servers in their data centers, taking advantage of BlueArc's massive scalability and integrated security features to provide secure storage to customers of all sizes. Whatever the application, BlueArc's customers benefit from a network storage system that can tackle a variety of access patterns, variable workloads, and unexpected spikes in demand without comprising system performance and reliability.

BlueArc performance and versatility is a result of an evolution in the architecture of the network storage server. In 1998, BlueArc took a fresh approach to network attached storage by creating its patented hardware accelerated architecture, delivering massive data path parallelization and providing the foundation for BlueArc's SiliconFS hardware based filesystem. Aptly named the SiliconServer, the first generation system utilized field programmable gate arrays (FPGAs) to move core filesystem functionality into silicon, dramatically improving the performance potential for network storage systems while adhering to established standards for network and storage system access. Subsequent generations of the SiliconServer and Titan server architecture have followed this core design methodology while continuously improving performance and scalability as well as delivering advanced data protection and data management features.

As the sixth generation of proven BlueArc platforms, BlueArc Mercury represents the next wave of network storage innovation, introducing a new Hybrid-Core Architecture that builds upon the proven FPGA based architecture of Titan 3000 servers while adding additional capabilities leveraging multi-core microprocessors. This not only separates processes that normally compete for system resources, but also utilizes the most appropriate processing resource for a given task. With Mercury servers, BlueArc extends its technology leadership beyond the needs of high performance customers to address the needs of mid-range commercial customers requiring cost effective network storage solutions designed to consolidate multiple applications and simplify storage management – all without compromising performance and scalability.

This white paper discusses the implementation and advantages of the BlueArc Mercury Hybrid-Core hardware platform and the fundamentals of the BlueArc operating system design. Core filesystem concepts such as the object store, NVRAM protection and clusters are included in this white paper for clarity and a greater understanding of how those functions operate within the physical system design.

The *SiliconFS™ White Paper*, available at www.bluearc.com, provides a deeper discussion of the SiliconFS filesystem as well as BlueArc's intelligent virtualization, tiered storage, data protection and data management features. Together, the Mercury Architecture Guide and the SiliconFS White Paper provide a solid foundation for understanding how BlueArc systems allow customers to break storage barriers without compromising performance, scalability or flexibility.

Advantages of a Hybrid-Core Architecture

The best software in the world is just a CD without a hardware platform to run the application. The fastest switch on the planet would be a doorstop without robust software to operate it. For BlueArc, it's not just about a powerful hardware platform or the versatile SiliconFS filesystem, it's about a unified system design that forms the foundation of the BlueArc Mercury storage solution. Mercury is a unique server that integrally links its hardware and software together in design, features and performance to deliver a robust storage platform as the foundation of the BlueArc storage system. On top of this foundation, BlueArc layers intelligent virtualization, data protection and data management features to deliver a flexible, scalable storage system.

One of the key advantages of using a hybrid-core architecture is the ability to optimize and separate data movement and management processes that would normally contend for system resources. Where traditional computers require arbitration between processes, Mercury's architecture allows data to be transferred between logical blocks in a parallel fashion, ensuring no conflicts or bottlenecks in the data path. This architecture is vastly different from conventional file servers where all I/O must navigate through shared buses and shared memory, which can cause significant performance reductions and fluctuations depending on how busy the CPUs are, how much system RAM is available, or how much data is currently moving through the PCI buses. This separation of data movement and management prevents competition for system resources, maintaining performance levels while concurrently supporting advanced file system functions. No other network storage platform can deliver this capability.

Another advantage of the hybrid-core architecture is the ability to target functions to the most appropriate processing element for that task, and this aspect of the architecture takes full advantage of the innovations in multi-core processing. High speed data movement is a highly repeatable task that is best executed in the FPGAs, but higher level functions such as protocol session handling, packet decoding, and error / exception handling need a flexible processor to handle these computations quickly and efficiently. Mercury's hybrid-core architecture integrates these two processing elements seamlessly within the operating and file system structure, using a dedicated core within the CPU to work directly with the FPGA layers within the architecture.

The remaining core(s) within the CPU are dedicated to pure system management processes, maintaining the separation between data movement and management. This is an innovation for Mercury over previous generation Titan servers, which could only be managed using a dedicated, external system management appliance (SMU). By leveraging the spare cores for system management, Mercury has the flexibility to run the java based system management GUI natively on the Mercury Management Board (MMB), or to integrate into an existing BlueArc environment managed by an existing SMU.

The primary design goal for the Mercury hybrid-core architecture is to deliver a flexible, reliable platform that can scale with the needs of mid-range customers, starting small with integrated management, and scaling out to multi-node clusters as well as scaling up to multi-petabytes of storage. With BlueArc Mercury, users know that they will get:

- **Simplified System Management** – The integrated system management provides simplified management of large, dynamic data sets to reduce administrative complexity
- **Lower Cost of Ownership** – With its ability to scale out and scale up, Mercury can be used to consolidate multiple storage systems into a single platform, dramatically improving overall cost of ownership
- **Reliable Performance** – By removing contention for resources in the data path, Mercury delivers reliable performance, regardless of the amount or type of workload applied to the system

A Solid Hardware Foundation

Mercury’s hybrid-core hardware architecture is a combination of high performance BlueArc proprietary technology and cost effective standard hardware components. Utilizing this combination of proprietary and off-the-shelf components is an important aspect of Mercury’s flexible architecture, and provides a strong foundation for future growth. Separating proprietary file serving from standard system management not only enables more effective utilization of each system resource, it also provides the ability to scale file serving performance or system management capabilities independently of each other. As an example, advances in multi-core processing are easily incorporated into the server’s management layer without affecting the network file serving performance, and vice versa.

The Mercury server consists of three major components: the Mercury server chassis; the Mercury Filesystem Board (MFB), a proprietary FPGA board that hosts the SiliconFS; and the Mercury Management Board (MMB), a standard motherboard that runs management level functions on Debian Linux. Each of these server components is discussed in detail below.



Mercury Front View



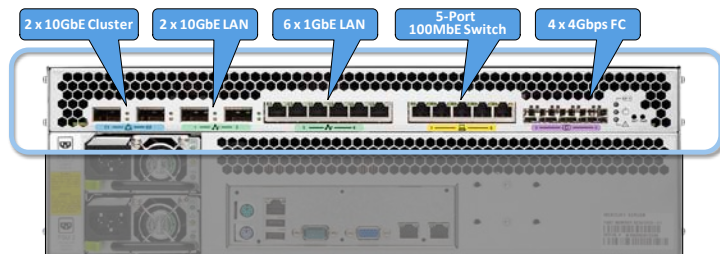
Mercury Rear View

Mercury Chassis

The Mercury server chassis is a 3U, 19” rack mountable enclosure that contains all of the major and minor components of the server and provides the external interfaces into the system. Behind the front fascia of the chassis are redundant, hot swap cooling fans, the NVRAM backup battery, and dual redundant hard drives. Configured as a RAID 1 mirror, the internal hard drives store the primary OS, server software images and system logs. At the rear of the chassis are dual, hot swap power supplies as well as the system interface ports.

Filesystem Interfaces

- *Simultaneous 1GbE and 10GbE access*
- *Direct-attach or SAN attached storage*
- *Integrated management switch*
- *High-speed 10GbE active/active cluster interconnect*

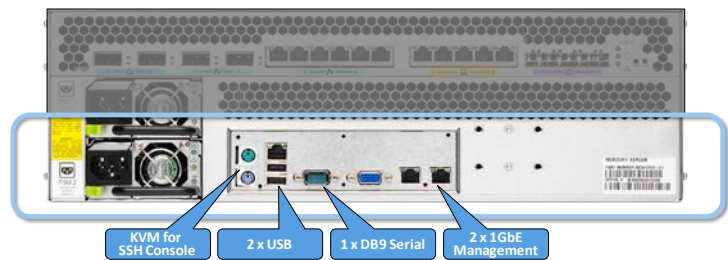


The filesystem interfaces are part of the Mercury Filesystem Board architecture, providing direct, high speed access to the appropriate internal FPGAs depending on port type. A unique system feature for Mercury is the simultaneous 1GbE and 10GbE LAN connectivity, allowing both speeds of ports to be active at the same time. This gives administrators greater flexibility in installing and sharing network storage across multiple internal LANs, regardless of the access speed. Another unique feature is an integrated 10/100 management switch that provides direct management connectivity for small installations.

All of the ports support industry standard connectors and transceivers. The 1GbE LAN ports and management switch ports feature built-in RJ45 connectors supporting CAT5 or better copper cables. The 10GbE LAN and Cluster ports both support standard XFP optical transceivers and the Fibre Channel ports support standard SFP optical transceivers, both of which support single-mode or multi-mode fiber transmissions over fiber optic cables.

Management Interfaces

- *Dedicated management ports supporting internal or external system management*
- *USB ports provide direct configuration backup options*
- *KVM ports allow shell level access for installation and troubleshooting*

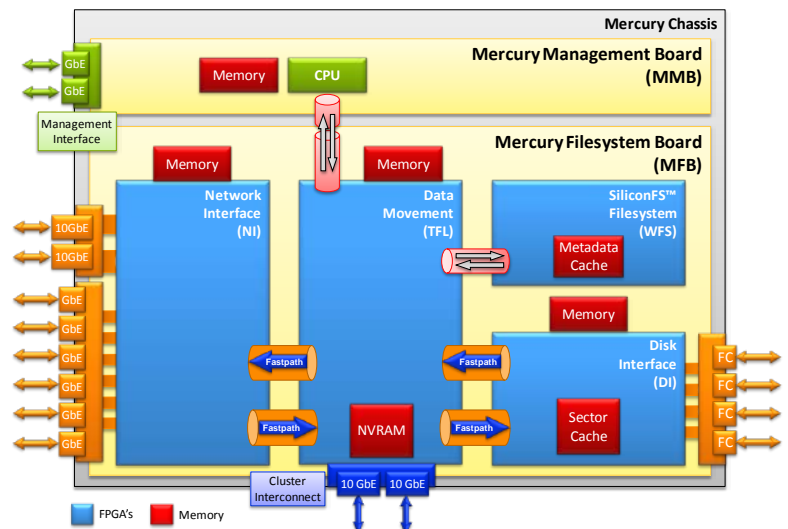


The system management interfaces connect directly to the management software on the Mercury Management Board, segregating system management traffic from file serving traffic. The java based System Management Unit (SMU) is accessed via the GbE network, although the administrator has the option of using the SMU utilities embedded in the server, or managing multiple servers using a standalone SMU appliance. For lower level troubleshooting or installation activities, administrators can connect via KVM to access the system console directly.

Mercury Filesystem Board (MFB)

The Mercury Filesystem Board is the core of the hardware accelerated file system, carrying forward BlueArc’s patented innovation in FPGA architecture and server design. The MFB is responsible for core filesystem functionality such as object, free space and directory tree management in addition to Ethernet and FC protocol handling.

Using dedicated Fastpaths between FPGAs, the MFB enables high-speed data movement between the network (LAN) and backend storage arrays. The MFB works in unison with the Mercury Management Board (MMB), coordinating management tasks across a dedicated PCI-E connection between the two boards.



Each of the FPGAs on the MFB has clear responsibilities, allowing each to operate independently from the others for high-speed data movement. Each of the FPGA’s (as well as the CPU on the MMB) has dedicated memory used for both general-purpose processing and function specific cache. This ensures that there is no contention for memory between any of the major processing elements within the server.

LVDS Fastpath

The MFB incorporates Low Voltage Differential Signaling (LVDS)¹ to guarantee very high throughput with low signal noise between FPGAs on the circuit board. Known as the Fastpath, it provides dedicated, point-to-point connections between FPGA’s for data reads and data writes. This is a key advantage of Mercury’s design, as it removes any need to arbitrate data movement over a shared bus resource; ingress and egress data pipelines have dedicated Fastpaths, ensuring no conflicts or bottlenecks as data moves through the server.

¹ See the ANSI EIA/TIA-644 standard for Low Voltage Differential Signaling (LVDS)

For example, a metadata lookup request is processed independently from a NVRAM mirror request and neither operation would have any impact on data being flushed from the sector cache to disk. Each of these operations would have its own dedicated pipeline between FPGAs and the interface ports. This is vastly different from a storage architecture that exclusively uses a standard CPU/motherboard architecture, where all I/O navigates through shared buses and memory, causing performance reductions and fluctuations.

Field Programmable Gate Arrays (FPGA)

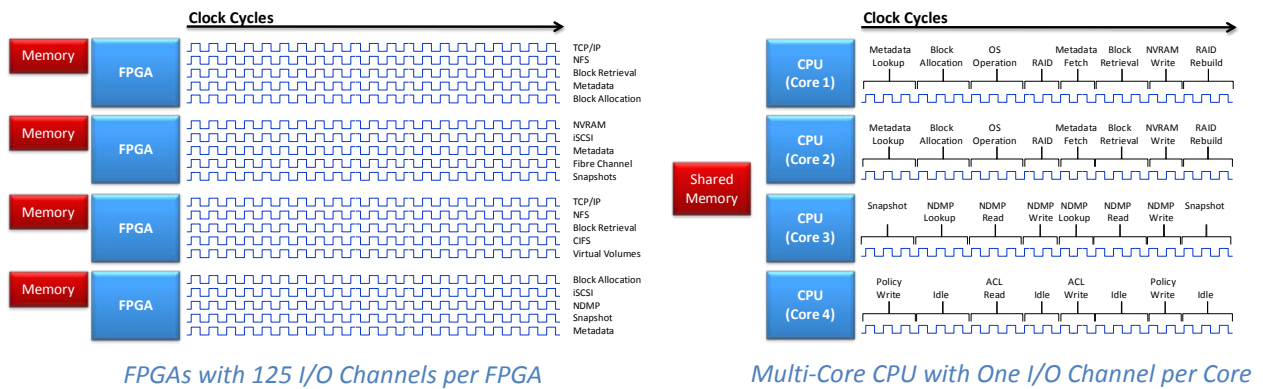
At the heart of the Mercury Filesystem Board architecture is a unique implementation of parallel state machine FPGAs. An FPGA is an integrated circuit that can be reprogrammed at will, enabling it to have the flexibility to perform new or updated tasks, support new protocols or resolve issues. Unlike ASICs (Application Specific Integrated Circuits) that cannot be upgraded after leaving the factory, FPGAs can be upgraded easily with a new firmware image in the same fashion as for switches or routers today.

Today's FPGAs are high-performance hardware components with their own memory, input/output buffers, and clock distribution - all embedded within the chip. In their core design and functionality, FPGAs are similar to ASICs in that they are programmed to perform specific tasks at high speeds. With advances in design, today's FPGAs can scale to handle millions of tasks per clock cycle, without sacrificing speed or reliability. This makes them ideally suited for lower level protocol handling, data movement and object handling.

For an analogy of how the FPGAs work, think of them as little factories. There are a number of loading docks (input/output blocks) and workers (logic blocks) which are connected by assembly lines (programmable interconnects). Data enters through an input block, much like a receiving dock. The data is examined by a logic block and routed along the programmable interconnect to another logic block. Each logic block is capable of doing its task unfettered by whatever else is happening inside the FPGA. These are individual tasks, such as looking for a particular pattern in a data stream, or performing a math function. The logic blocks (aka 'gates') perform their tasks within strict time constraints so that all finish at virtually the same time. This period of activity is measured by the clock cycle of the FPGA, and Mercury FPGAs operate at 125 million clock cycles (125 MHz) per second.

Mercury has over 1 million logical blocks inside its primary FPGAs, giving it a peak processing capacity of about 125 trillion tasks per second – almost 7000 times more tasks than the fastest general purpose CPU. (NOTE: As of this writing, Intel's fastest 8-core microprocessor is rated for 18 billion tasks per second). Because each of the logic blocks is performing well-defined, repeatable tasks, it also means that performance is very predictable, allowing BlueArc to reliably design servers to meet specific performance goals. As new generations of FPGAs increase the density of logic blocks, I/O channel and clock speeds, BlueArc has introduced increasingly more powerful servers.

This isn't to say that FPGAs are always better to use than multi-core CPU's – they aren't. CPU's are best technology choice for advanced functions such as higher-level protocol processes and exception handling, functions that are not easily broken down into well-defined tasks. This makes them extremely flexible as a programming platform, but it comes at a tradeoff in reliable and predictable performance. As more processes compete for a share of the I/O channel into the CPU, performance is impacted. The introduction of multi-core CPU's and hyper-threading have alleviated this I/O congestion, but not eliminated it.



One of the best ways to demonstrate the difference between FPGA’s and CPU’s is shown in the graphic above. With up to 125 high speed I/O blocks per FPGA (loading docks in the example above), FPGAs by design deliver massive parallel processing capabilities which allows data to move through them at extremely high speeds. Multi-core CPU’s on the other hand, are limited to a single channel per core (two if hyper-threaded), and process data in a serial fashion, using arbitration to share the I/O bus among various processes. CPU’s, however, have the advantage over FPGAs in the variety and complexity of the processes that they can address. It is for this reason that Mercury incorporates both FPGAs and multi-core CPU’s in its hybrid-core design: FPGA’s for high-speed data movement, CPU’s for higher level management and administrative functions.

Memory (Buffers and Caches)

In any computing system, memory is a critical component that affects the performance of the system, second only to the core processor (or FPGA in Mercury’s case). For general purpose computing, memory is so ubiquitous with performance that just by knowing the amount of memory, a savvy consumer can quickly determine the performance profile for the system. Inside Mercury, however, it’s not just the amount of memory the system has, but where it is used and how fast it is that determines the performance of the server.

Each major processing element (FPGAs and CPU) in the Mercury server has dedicated memory used for buffering commands and data. One of the key considerations for the type and quantity of memory used is the bandwidth required to support the FPGAs at full speed. Just like CPUs, memory is available in various clock speeds. Although currently reported in the MHz range, the fastest memory at this time runs in the GHz range (ex. 1333Mhz). Mercury’s memory buffers are designed to ensure that the memory banks have enough bandwidth to properly support both the read and write data pipelines without creating performance bottlenecks for the FPGAs.

In addition to general purpose memory buffers dedicated to each FPGA and the multi-core CPU, there are three banks of memory dedicated to critical functions within the server: **Non-Volatile RAM (NVRAM)**, battery backed memory which maintains a record of all transactions in progress; **Metadata Cache**, which holds the Object Store and directory structure information in memory for instant access by the SiliconFS; and **Sector Cache**, memory used to store data to be written to or that has been read from disk. Their purpose and use are described in more detail in the appropriate FPGA section below.

Network Interface FPGA (NI)

Responsible for:

- *High Performance Gigabit Ethernet and 10GbE Connectivity*
- *Hardware processing of protocols*
- *OSI Layers 1-5*

The Network Interface FPGA (NI) is responsible for all Ethernet based I/O functions corresponding to OSI² layers 1 thru 4 and the most frequently accessed commands and functions of OSI layer 5. Functionally equivalent to the Network Interface Module (NIM) on previous generation Titan servers, the NI implements the TCP/IP stack in a dedicated FPGA and handles supported protocols natively within the FPGA. The functions implemented on the NI include processing standard Ethernet and Jumbo Ethernet frames up to 9000 bytes, ARP, IP protocol and routing, and of course the TCP and UDP protocols.

Due to the massive parallel nature of the FPGA architecture, the NI supports over 60,000 concurrent client connections while ensuring that there are no wait-states on the connections. The result is nearly instantaneous network response with the highest performance and lowest latency of any other network storage system available. Similar to how today's TCP Offload Engine (TOE) cards are used general-purpose servers, BlueArc has used dedicated FPGAs for TCP offload since the first generation BlueArc servers introduced in 1998. However, unlike a typical TOE card, NI transmits data over dedicated Fastpath connections within the server rather than relying on a shared bus as found in typical server architectures, ensuring contention free data movement. As with all FPGAs in Mercury, NI has dedicated buffer memory for both transmit and receive operations, further removing contention for resources within the server architecture.

Each of the LAN ports connect directly to the NI through a high-speed data path providing full duplex, dedicated bandwidth to each port: 20Gbps aggregate bandwidth to the two 10GbE ports, and 6Gbps aggregate bandwidth to the six 1GbE ports. Unique to Mercury is the ability to support dual speed server access, allowing both the 1GbE and 10GbE ports to be active at the same time. Dual speed access provides the flexibility for the server to be used both as a high performance storage system for automated system processes over 10GbE, as well as for general file storage for the network users over 1GbE. The NI also supports link-aggregation of same speed ports using the Link Aggregate Control Protocol (LACP)³, allowing dynamic changes to the link aggregation and enabling higher availability and higher throughput of data, critical for shared data environments.

Data Movement (TFL)

Responsible for:

- *Intra-server traffic management*
- *NVRAM Logging and Mirroring*
- *Cluster Interconnect*
- *Cluster Namespace redirection*

The Data Movement FPGA (TFL) is responsible for all data and control traffic routing throughout the server, interfacing with all major processing elements within the server, including the MMB, as well as connecting to companion servers within a Mercury cluster. TFL derives its acronym from its core functions: To disk processing; From disk processing; and Logging. TFL controls all data movement and sits directly on the FastPath pipeline between Network Interface FPGA (NI) and Disk Interface FPGA (DI), transforming, sending and receiving data to

² Refer to ISO/IEC 7498-1 Open Systems Interconnection (OSI) Basic Reference Model

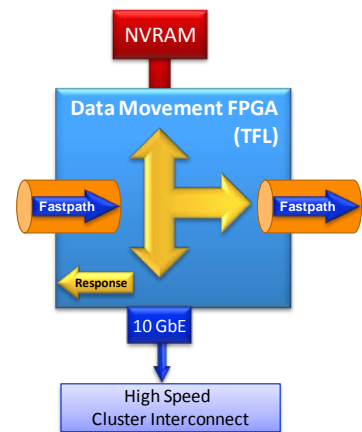
³ See the IEEE 802.3ad standard for Link Aggregation Control Protocol

and from NI and DI. TFL is also responsible for routing command packets between the MMB, WFS (the FPGA responsible for BlueArc's Object Based SiliconFS file system), NI and DI.

TFL is also responsible for intra-cluster communications within a Mercury cluster. Two 10GbE cluster interconnect ports controlled by TFL provide redundant paths to all servers in the cluster. NVRAM mirroring, Cluster Namespace, packet redirection and all other intra-cluster communications travel across this dedicated path between Mercury servers, providing a high degree of data protection and data access to all servers in the cluster. To minimize latency and overhead in the cluster path, only a fraction of the Ethernet stack is used for data transfer, ensuring that the cluster interconnect can scale to multiple nodes in the cluster (up to 8 currently) without impacting the performance and availability of all nodes in the cluster.

When NI sends a client request packet forward, TFL coordinates with the MMB to decode the request, and then executes any required commands on the packet, including any communications required with WFS and DI for data read and write requests. The write example found later in this paper is a powerful example of the parallelism in Mercury's FPGA architecture, but the actions of the TFL in the write operation provide the best demonstration of that parallelism. When TFL receives a request from NI, several functions will execute simultaneously:

- Data is pushed into NVRAM to guarantee the data is captured in battery backup protected memory
- Data is pushed across the cluster interconnect to update the cluster partner NVRAM
- Data is sent over the FastPath pipeline to the DI for further processing
- Metadata is sent to WFS to update the Metadata cache
- A response packet is formed to be returned to the NI



Upon successful completion of all of these commands, TFL transmits the response packet back to NI, which will in turn send the response back to the client. Thus, what would be five serial steps in a traditional file server are collapsed into a single atomic parallel step within TFL.

Non-Volatile RAM (NVRAM)

TFL is responsible for ensuring that copies of all data requests are stored in NVRAM. NVRAM is the only memory bank that has battery backup to preserve the data in memory in the event of an unexpected server outage. Once data is written to NVRAM, it is fully protected and an acknowledgement can be returned to clients immediately, rather than waiting until the data is stored further down the pipeline in sector cache or on disk. If the server shuts down before pending write requests can be flushed to disk, there will still be a valid copy of that data in NVRAM that can be used to recover the filesystem to a consistent state during the next reboot.

NVRAM is partitioned into two halves, so that at any given time one half is actively receiving new write requests from clients, while the other half is stable (check-pointed) and maintaining a consistent image of data while data is flushed to disk from sector cache on DI. If a server fails, the last filesystem checkpoint is used to rollback the filesystem to a consistent known state. Then, the contents of the NVRAM are replayed against the filesystem objects, guaranteeing that the customer data is complete and intact at the end of the recovery or reboot.

When servers are in a cluster, NVRAM is further partitioned into halves again, with one half dedicated to storing local data and the other half to storing the cluster partner's transaction data. This ensures that if the Mercury server fails, the remaining server in the cluster can take over and complete the recovery process with its copy of the NVRAM data.

Disk Interface (DI)

Responsible for:

- *Fibre Channel Processing*
- *SCSI Command Processing*
- *Parallel RAID Striping*
- *Sector Cache Management*

The Disk Interface FPGA (DI) is responsible for connectivity to physical disk or solid-state storage on the back-end Fibre Channel SAN, and for controlling how data is stored and spread across those physical devices. As with NI and its LAN ports, DI connects directly to the Fibre Channel ports through a high-speed data path providing full duplex, dedicated bandwidth to each port: 16Gbps aggregate bandwidth to the four 4Gbps FC ports. These ports provide redundant high-speed connectivity to the attached RAID storage systems. Smaller installations can connect directly to the dual redundant controllers on the RAID arrays, reducing the cost of entry level systems. Larger installations with more than two RAID arrays connect through a FC switched fabric.

DI logically organizes the attached RAID storage into a virtualized pool of storage (Storage Pool), striping data across any number of drives and arrays to provide the best throughput without creating ‘hot-spots’ on any one drive or array. This server level RAID striping across storage systems is a key advantage for BlueArc servers, as it allows more drive spindles to be included in data transfers, ensuring the best storage performance. The virtualized storage architecture that is built on top of these Storage Pools is discussed in depth in the *SiliconFS™ White Paper*.

Sector Cache

DI also controls the Sector Cache memory, a large block of memory that stores data writes waiting to be committed to disk. Once data has been stored in sector cache, it is immediately available for read requests, allowing all users immediate and equal access to newly written data. While not usually a consideration for general file serving to end users, it is an important consideration when using Mercury servers for applications needing parallel access to data for data processing. After the data is written to disk, a copy remains in sector cache to quickly respond to subsequent read requests until space is needed for new incoming data.

SiliconFS™ (WFS)

Responsible for:

- *SiliconFS™ (Hardware File System)*
- *Advanced Features*
- *Object Store Layer*
- *MetaData Cache Management*
- *File System Attribute Caching*

The SiliconFS FGPA (WFS) is responsible for the object based filesystem structure, metadata management, and for executing advanced features such as data management and data protection. It *is* the hardware filesystem within the Mercury server. By moving such fundamental filesystem management tasks as onode and object management into WFS, Mercury achieves an unprecedented amount of parallelism at the filesystem layer that enables consistently high performance, and the ability to stretch beyond typical filesystem size and scalability limitations.

Because of the importance and complexity of the SiliconFS, it is discussed in depth in the *SiliconFS™ White Paper*. However, there is one very important filesystem concept that will be repeated here, and that is the Object Store. Designed as an object-based file system, SiliconFS uses the Object Store to deliver superior file lookup performance, allowing Mercury to maintain consistent, reliable performance even with densely populated directories, very large directories or even multiple petabytes of data.

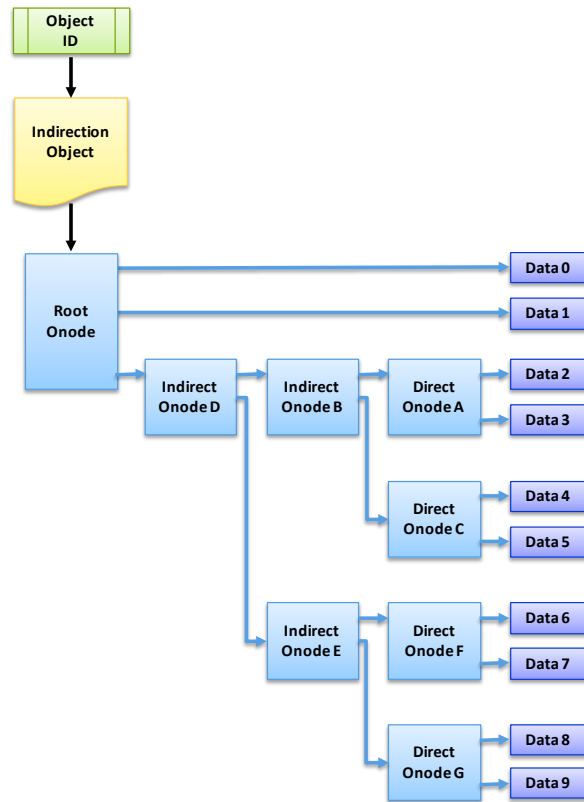
BlueArc’s Unique Object Store

The Object Store is an abstraction layer between the normal presentation of a file system view to the user and the raw blocks of storage managed by DI. Client machines have no concept of objects or the object store, and when accessing Mercury via standard protocols, they expect to work with string names and file handles. WFS transparently converts these string names and file handles into objects and object IDs within the server so that clients only see the standards based representation of files.

In object-based filesystems, objects are manipulated by the filesystem and correspond to blocks of raw data on the disks themselves. Information about these objects, or the object metadata, is called an onode, in much the same way inodes refer to file metadata in a traditional filesystem. In BlueArc’s Object Store the underlying structure used to build up SiliconFS is an “object”, which is any organization of one or more of these raw blocks of data into a tree structure. The object is a container of storage that can be created, written to, read from, deleted, etc.

Each object has a unique 64-bit identifier called the Object Identifier (Object ID) which is used by the filesystem to locate the object on disk. The Object ID points to an index entry in the Indirection Object, which translates the Object ID to the physical location of the Root Onode. The Indirection Object is another abstraction layer, one that allows the Root Onode to be moved on disk without requiring the Object ID to be changed, which is critical since the Object ID can be referenced by other objects. The Indirection Object is an important filesystem construct; by allowing the root onode to move at will, it enables the filesystem to be more efficient, increasing performance as well as enabling more effective disk de-fragmentation.

The Root Onode is the primary element at the base of the objects on disk structure, and contains the metadata relevant to the object, including the Indirection Object information. Root Onodes can point to either a) **Data Blocks**, the location of data on disk, b) **Direct Onodes**, which then point to Data Blocks or c) **Indirect Onodes**, which then point to other Indirect or Direct Onodes. The structure and quantity of the onodes depends on the size of the object being stored.



Different types of objects serve different purposes. User data is contained in a File Object. Directories and subdirectories are contained in a Directory Object and have companion Direct Tree Objects used for binary search tree algorithms that dramatically improve lookup speeds. Directory Objects contain file and directory names in various formats (dos short names, POSIX, etc.), a CRC hash value and file handles (Object IDs) that point to the locations of another objects such as a subdirectory (Directory Object) or a file (File Object).

Some objects, such as the indirection object and free space bitmap, contain critical metadata and have special protections built into the filesystem to guard against corruption, increasing the resiliency of the filesystem. Critical objects have redundant copies maintained on separate physical devices to guard against both hard disk errors as well as filesystem corruption. Key metadata objects are also validated using CRC error checking and other data validation mechanisms. All objects in the filesystem, including the free space bitmap, can be check-pointed (snapshot), and the filesystem and all its objects can be rolled back to any valid checkpoint (128 is the default at this time) when needed for system recovery.

Metadata Cache

WFS manages the metadata cache, which includes the memory dedicated to the Object Store as well as buffer space for other filesystem functions. Due to its importance in the operation of SiliconFS and the amount of work performed by the Object Store, Metadata Cache is the largest memory bank within the Mercury server. The metadata cache also holds active copies of metadata for recently accessed files and directories, which allows many client queries to be fulfilled directly from the metadata cache rather than requiring a read operation from disk.

Mercury Management Board (MMB)

Responsible for:

- System Management
- Security and Authentication
- NFS, CIFS, iSCSI, NDMP
- OSI Layer 5, 6 & 7 Protocols

The Mercury Management Board provides out-of-band data management and system management functions for the Mercury server. The MMB is a commodity motherboard that has been integrated into the proprietary Mercury design. The actual specifications of the motherboard or its multi-core CPU aren't as important as understanding its role in the hybrid-core architecture.

Leveraging the flexibility of multi-core processing, the MMB serves a dual purpose. In support of the MFB, the MMB provides high-level data management and hosts the BlueArc operating system within a dedicate CPU core, in a software stack known as BALI. The remaining cores of the CPU are set aside for the Linux based system management, monitoring processes and application level APIs. These functions will be discussed further in the next section, which discusses the BlueArc Operating System software.

When supporting the MFB for file serving functions, the MMB (BALI) provides high level processing resources for advanced features and functions as needed. When working at the protocol level, the MMB is responsible for the portions of OSI layer 5 not implemented in NI, and all layer 6 and 7 protocol functions such as CIFS state handling. For advanced data management features and exception-processing functions, the MMB provides the command and control, issuing commands to the MFB (via TFL) to execute as required. Snapshot management, quota, file and directory locking are a few examples of higher-level features controlled by the MMB.

Fundamentals of the Mercury Operating System

Mercury's operating system incorporates a blend of high performance BlueArc proprietary software and open-source Linux based software components. As with the hardware, using this combination of proprietary and off-the-shelf software is an important feature in Mercury's flexible architecture, and provides the foundation for advanced software feature deployment. By separating filesystem and management processes both physically (by CPU core) and logically into atomic software units, BlueArc can easily develop and deploy new features and functionality to any component with the system without adversely impacting other processes and performance.

This section will focus on the Linux-based operating system structure that runs on the Mercury Management Board, only lightly touching on the SiliconFS in order to demonstrate how it seamlessly integrates into the overall software architecture, especially it's interaction with the embedded system management application. With the wealth of powerful data protection, data management and virtualization features that are part of the SiliconFS, we encourage you to read the [SiliconFS™ White Paper](#) to learn more.

The Mercury Management Board is powered by a Debian Linux OS kernel. In user space on top of that kernel runs the four primary software functions of the server operating software that work together with SiliconFS to power

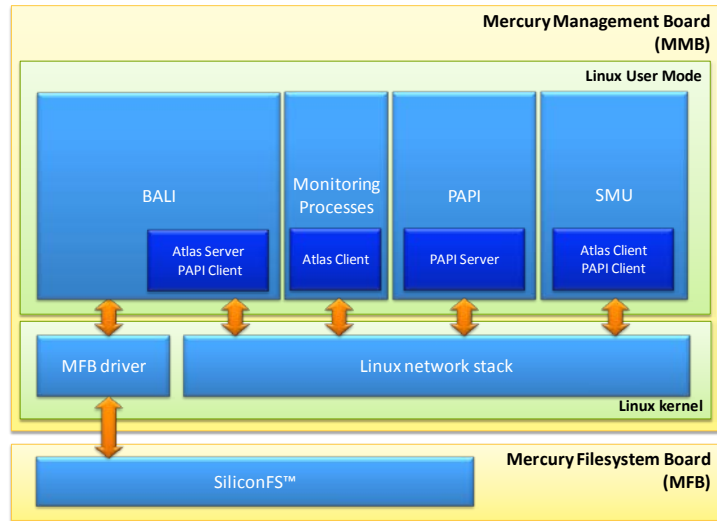
the reliability and performance of the Mercury server. They are: BALI, the BlueArc operating system responsible for the high level filesystem functions of the SiliconFS; PAPI, the platform API that allows propriety BlueArc software to run seamlessly on Linux; SMU, the java-based System Management Unit; and the system monitoring processes that continuously monitor the server.

BlueArcOS and Linux Incorporated (BALI)

Responsible for:

- BlueArc Operating System
- NFS, CIFS, iSCSI, NDMP
- OSI Layer 5, 6 & 7 Protocols
- Atlas API

BALI is the Linux-based incarnation of the BlueArc Operating System (BOS) that lies at the core of the SiliconFS, and is at the heart of the Mercury’s hybrid-core architecture. BALI leverages a decade’s worth of software development and the system maturity of the BlueArc Titan server to provide a solid base for the Mercury system software.



BALI is responsible for functions such as CIFS state handling, and any other OSI layer 5 – 7 protocol functions, while the data payload remains on the MFB. For advanced data management features and exception-processing functions, BALI provides the command and control, issuing commands to the MFB (via TFL) to execute as required. Snapshot management, quota, file and directory locking are a few examples of higher-level features controlled by BALI.

BALI also provides the linkage point between the MFB and MMB functionality. BALI has a dedicated interface to TFL on the MFB (refer to the [MFB block diagram](#)) through the MFB driver. Using internal APIs, BALI communicates with both the Platform API and the internal SMU to accept new user commands and ensure the system is kept up to date. In a Mercury cluster, BALI uses the cluster interconnect link on the MFB to duplicate and sync up the master configuration records among cluster partners, ensuring consistency among the nodes in the cluster.

Atlas API

The Atlas API is an internal API that provides a programmatic interface between BALI and the SMU and system monitoring processes. As with any API, it provides a level of abstraction between processes, allowing BALI and the SMU to be updated independently of one another.

Platform API (PAPI)

Responsible for:

- Linux Management
- PAPI Server

The Platform API (PAPI) is the system configuration interface to the Linux kernel, and ensures that the Linux kernel remains up to date and in sync with any system level changes made in either the SMU or BALI. As an abstraction layer within the Mercury architecture, it allows BALI and the SMU to apply configuration settings and updates to the underlying Linux kernel in a consistent manner, without either process needing to understand the specifics about the kernel itself.

PAPI monitors the Linux configuration settings to detect if any mismatches have occurred, and fixes them automatically, using BALI as the master configuration record. Because BALI maintains duplicate images of the configuration records within a Mercury cluster, PAPI's housekeeping activities ensure all nodes in the cluster have consistent and reliable configurations, providing an additional layer of resiliency within the server software.

System Management Unit (SMU)

Responsible for:

- System and Storage Configuration
- User Administration
- Status Reporting
- Call Home

The System Management Unit (SMU) is a java-based GUI console that provides all of the management functions for both BlueArc Mercury and Titan storage systems. Historically, the SMU has been deployed as a standalone server appliance to manage Titan servers and clusters. Referred to as an **external SMU**, this deployment option has the ability to manage multiple servers and clusters, and can manage both Titan and Mercury servers simultaneously. A new deployment option enabled by Mercury's hybrid-core architecture is to run the SMU application natively within the Mercury server. When running an **internal SMU**, Mercury is a self-managing system.

The SMU is responsible for all system, storage and user administration. Core features such as NFS, CIFS and iSCSI protocol support, snapshots, server and storage virtualization and automated quota management are configured using the SMU GUI interface, but are executed in real time on the MFB as data is stored on the system. The SMU also provides the user interface for system status monitoring, and provides Call Home services, sending server status and error alerts to BlueArc Support at regular intervals. These alerts allow the BlueArc support team to take proactive steps and perform preventative maintenance on failure conditions such as replacing hard drives with excessive disk errors, or replacing backup batteries that have reached the end of their useful life.

It is important to highlight that the SMU is not responsible for executing any of the commands and policies configured via the GUI. This means that if the SMU is offline for any reason, Mercury's file serving functionality isn't affected, and it will continue to move and manage data indefinitely. The only actions that won't be performed are the reporting and call home functions of the SMU.

To reiterate the separation of responsibilities:

- **SMU – Configuration** – The SMU is the user facing configuration utility, where the administrator configures users, policies and filesystems and other server attributes.
- **BALI – Command and Control** – Bali controls the execution of policies and advanced system features such data migration and creates the commands that are sent to the MFB to execute.
- **MFB FPGAs – Execution** – The FPGAs execute data movement and management commands as sent by BALI, leveraging the massive parallel nature of the FPGAs to perform their tasks reliably at high speed.

Using Data Migration as an example, once the administrator creates the data migration targets and migration policies, the SMU saves a copy of this information and transmits the configuration settings to BALI via the Atlas API. At the appropriate time (according to policy), BALI creates and sends commands to the MFB FPGAs to execute the file migration. For larger installations where the SMU is running in an external appliance, the commands are transmitted to BALI over the private management network connection.

Monitoring Processes

Responsible for:

- Chassis Monitoring
- Internal RAID Monitoring
- Server Process Monitoring

The system monitoring processes watch over the health of both the physical server and system drivers running in software. Just as the SiliconFS architecture provides several layers of resiliency and protection for user data, the system monitors ensure that the server itself is also highly resilient and fault tolerant. Chassis monitors look after the power and cooling, sending alerts to the management console when they detect a part that needs attention or replacement. The internal RAID monitor ensures that the internal hard drives holding the operating system images remain in sync and online. The server process monitoring not only performs routine housekeeping as needed, but also automatically restarts stuck processes, preventing server outages and downtime.

Beyond the Server – Storage System Architecture

Mercury's unique architecture is the foundation of a versatile, reliable network storage solution, but it is only the foundation for the solution, not the entire solution. Mercury seamlessly integrates third party RAID storage into the BlueArc storage ecosystem, transforming the storage with intelligent storage tiers and multiple layers of virtualization. The result is the versatile Mercury storage system, a complete solution for mid-range commercial enterprises requiring cost effective network storage solutions designed to consolidate multiple applications and simplify storage management – all without compromising performance and scalability.

Versatility and reliability are core design principles that extend beyond Mercury into the storage ecosystem, and to the storage arrays that are part of it, although these principles carry forward from the earliest days of the Titan system architecture when performance and scalability reigned supreme. The need for a storage solution that provided strong data protection, high performance and scalability into the multi-PB range led BlueArc to design a solution that used third party RAID arrays as building blocks within the storage ecosystem.

First and foremost, the storage systems must protect the data that is stored on them, which means disk level RAID and full redundancy in the data path between server and storage. All storage arrays feature RAID levels 1, 5 and 6 at a minimum, with dual active / active RAID controllers on each RAID array. The arrays connect to the server over a redundant Fibre Channel SAN fabric of at least two switches, providing a fully redundant path between the ports on DI and the storage array.

Using hardware RAID controllers on the back end SAN has the added benefit of increasing both versatility and reliability of the storage system. By leveraging the computing resources of the RAID controller, BlueArc adds both storage capacity and computing capacity in equal measures, ensuring that scalability isn't limited by the computing power of the Mercury server. The backend SAN also provides greater data availability, especially in a cluster, since all Mercury servers in a cluster have equal access to the full storage capacity, and can instantly access and manage shared storage if one of the servers in the cluster becomes unavailable.

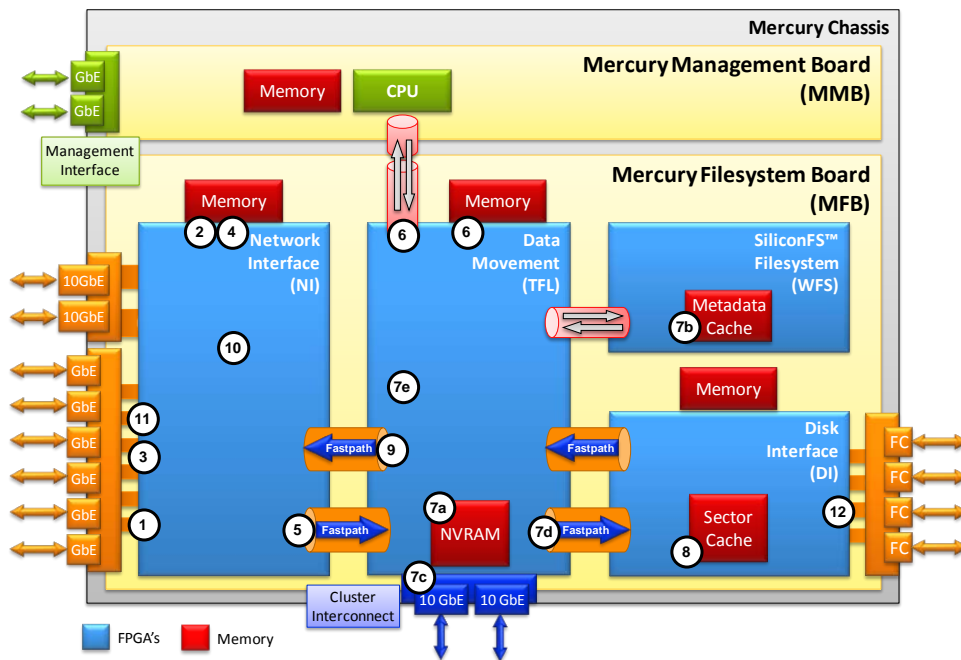
Each of the storage systems virtualizes and protects the physical disk using RAID, and presents storage capacity to the Mercury server as a RAID volume or LUN. DI then creates a stripe (effectively adding RAID 0) across these LUNs and adds that stripe to a virtualized pool of storage (Storage Pool). Because the Storage Pool is a logical, rather than physical, construct, new stripes can be added to the storage pool dynamically, without requiring any downtime for the server or storage pools. With advanced features such as dynamically read and write balancing, this new capacity immediately adds to the overall performance of the storage system, without creating performance 'hot-spots'.

This building block approach to storage is a core feature of BlueArc’s Intelligent Tiered Storage, which allows IT administrators to optimize storage into performance tiers to fit the needs of different tasks or applications, and dynamically manages the transitions of data through each tier of storage. By using the controllers ability to manage the underlying storage media, all BlueArc servers, Titan and Mercury, can support multiple types of storage media to reside concurrently on the back end SAN, whether high performance Solid State Disks and SAS, or high capacity nearline SAS and SATA.

Life of a Packet

This packet walk-thru demonstrates the how the software and hardware architecture work together to speed packet processing through the Mercury architecture. With dedicated read and write FastPath connections and dedicated memory banks per FPGA, Mercury delivers a reliable, versatile platform where performance is maximized for both IOPS and throughput for a variety of application environments.

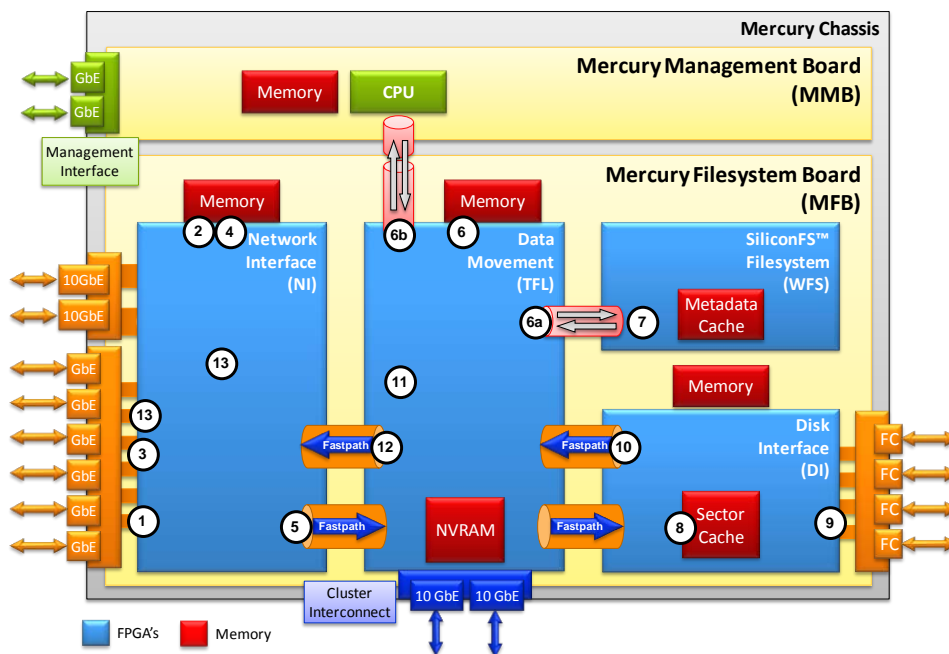
Write Example



1. NI receives a network packet from one of the Ethernet ports
2. NI saves the incoming packet to the memory modules assigned to it
3. If the incoming packet is a network only request, such as a TCP session setup, NI processes the request in full and sends a response to the requesting client
4. Otherwise, NI will gather additional related incoming packets and save those to memory
5. Once all packets of a request are received, the complete request is passed over the LVDS FastPath to TFL
6. TFL saves the request in the memory assigned to it, and sends the header information required for decoding the request to BALI running on the MMB
7. When BALI has finished the initial processing and decoding the request header information, it will send a write request (header information only) to WFS. WFS and TFL work together to simultaneously
 - a. Store the write request to NVRAM
 - b. Update the Metadata Cache

- c. Send a duplicate copy of the write request over the cluster interconnect ports (if the server is part of a cluster)
- d. Send the write request over the FastPath to the DI
- e. Begin formulating a response packet
8. DI stores the write data to Sector Cache and waits for additional write requests or instructions from TFL
9. Once DI has received the write request, and the data is safely stored in NVRAM, TFL sends the response packet to NI. This indicates that the write request has been successfully received and protected in memory, and allows the client to continue processing and sending write requests without needing to wait for the data to be written to disk.
10. NI organizes the response into segments that comply with the appropriate protocol and Ethernet formatting
11. NI transmits the response out the Ethernet Port
12. At specified intervals, usually just a few seconds (or when NVRAM is halfway full), TFL sends a command to DI to flush any outstanding write requests stored in Sector Cache out to the external storage system connected via the Fibre Channel ports. When possible, a technique known as Superflush is used to aggregate large I/O operations to disk in order to maximize throughput to storage.

Read Example



Steps 1 through 5 are virtually the same as the previous example. NVRAM is not involved in a read request.

6. TFL saves the request in the memory assigned to it, and forwards the request for processing to either:
 - a. WFS; if the protocol supports Autoinquiry, the request may be sent directly to WFS
 - b. BALI; the header information required for decoding the request is sent to the BALI software running on the MMB prior to sending the request to WFS
7. WFS will locate the metadata required for the read request in either the Metadata Cache or by loading from disk via the commands sent to DI. Once the metadata is located, WFS creates and issues read requests to DI.
8. DI will first look for the requested data in Sector Cache. If found, there is no need to look for the data on external storage, and the read request is processed immediately

9. If the data is not in Sector Cache, DI will formulate a Fibre Channel request to retrieve the data from disk and store it in Sector Cache. Both metadata and data requests are stored in Sector Cache once retrieved from disk
10. DI sends the retrieved data over the FastPath to TFL
11. TFL will re-apply the RPC layers in order to create a well-formed response packet.
12. TFL sends the response packet with the request read data over the FastPath to NI
13. NI organizes the response into segments that comply with the appropriate protocol and Ethernet formatting
14. NI transmits the response out the Ethernet Port

Conclusion

BlueArc is proud to build reliable, flexible storage solutions to address the needs of customers requiring cost effective network storage systems designed to consolidate multiple applications and simplify storage management – all without compromising performance and scalability. The BlueArc Mercury is a premium, network storage platform for the mid-range that delivers:

- Consolidation of systems and applications to lower the TCO of storage infrastructures
- Simplified management of large, dynamic data sets to reduce administrative complexity
- A hybrid-core file system enabling the highest performance and scalability of any product in its class

Mercury is ideal for a multitude of uses, but it is particularly well suited to consolidate multiple, disparate storage systems into a single, easy to manage storage solution. Mercury supports multiple applications, a large numbers of concurrent users and requires fewer storage devices as compared to traditional storage systems. Mercury's concurrent use of multiple protocols allows the consolidation of Windows clients, UNIX and Linux clients, and clients requiring block data access with a single network storage solution.

Mercury has unmatched scalability as a result of its unique storage system architecture. Because Mercury manages logical volumes rather than raw devices, it can scale far beyond physical storage system limitations into the multi-PB range. Its design will support even higher capacities in the future through software upgrades, ensuring that today's system investment will scale with customers ever increasing capacity needs.

Mercury has proven performance, as evidenced by its outstanding results on the SPECsfs2008® benchmark results, outperforming every other single server and two-node cluster systems of any class. These results are available at the SPEC.org website at <http://www.spec.org/sfs2008/results/sfs2008.html>. The heavy duty and variable workload of the SPECsfs2008 benchmark demonstrates Mercury's versatility and ability to handle any type of workload a user or application may throw at it.

BlueArc Mercury is the right choice for your storage needs, now and for the future.

About BlueArc

BlueArc is a leading provider of high performance unified network storage systems to enterprise markets, as well as data intensive markets, such as electronic discovery, entertainment, federal government, higher education, Internet services, oil and gas and life sciences. Our products support both network attached storage, or NAS, and storage area network, or SAN, services on a converged network storage platform.

We enable companies to expand the ways they explore, discover, research, create, process and innovate in data-intensive environments. Our products replace complex and performance-limited products with high performance, scalable and easy to use systems capable of handling the most data intensive applications and environments. Further, we believe that our energy efficient design and our products' ability to consolidate legacy storage infrastructures, dramatically increases storage utilization rates and reduces our customers' total cost of ownership.



BlueArc Corporation
Corporate Headquarters
50 Rio Robles Drive
San Jose, CA 95134
t 408 576 6600
f 408 576 6601
www.bluearc.com

BlueArc UK Ltd.
European Headquarters
Queensgate House
Cookham Road
Bracknell RG12 1RB, United Kingdom
t +44 (0) 1344 408 200
f +44 (0) 1344 408 202