chassis, multiple motherboards in multiple chassis, or a heterogeneous combination of these. The memory **110** used to hold storage volumes and snapshot volumes may be distributed across multiple subsystems in various ways. For example, the first memory cache **170** may be located in one sub-system, while the corresponding second memory cache **180** may be located in one or more other sub-systems. When sub-systems are installed in different geographic locations, or are associated with different clients, the storage system **100** may be used to perform rapid data migration, such as a migration of a virtual machine from one compute node to another. Similarly, the cache allocated to each volume may be spread and/or duplicated across multiple such sub-systems to provide increased data protection through redundancy and/or dispersion, or increased application performance through multiple points of access. Such spreading/duplication may be accomplished using error-correcting codes, such as used in RAID (redundant array of inexpensive disks) systems.

The storage system **100** may be implemented in many different ways. For example, although some features are shown stored in computer-readable memories (e.g., as logic implemented as computer-executable instructions or as data structures in memory), all or part of the system and its logic and data structures may be stored on, distributed across, or read from other machine-readable media. The media may include hard disks, floppy disks, CD-ROMs, a signal, such as a signal received from a network or received over multiple packets communicated across the network.

The processing capability of the storage system **100** may be distributed among multiple entities, such as among multiple processors and memories, optionally including multiple distributed processing systems. Parameters, databases, and other data structures may be separately stored and managed, may be incorporated into a single memory or database, may be logically and physically organized in many different ways, and may implemented with different types of data structures such as linked lists, hash tables, or implicit storage mechanisms. Logic, such as programs or circuitry, may be combined or split among multiple programs, distributed across several memories and processors, and may be implemented in a library, such as a shared library (for example, a dynamic link library (DLL)).

All of the discussion, regardless of the particular implementation described, is exemplary in nature, rather than limiting. For example, although selected aspects, features, or components of the implementations are depicted as being stored in memories, all or part of systems and methods consistent with the innovations may be stored on, distributed across, or read from other computer-readable media, for example, secondary storage devices such as hard disks, floppy disks, and CD-ROMs; a signal received from a network; or other forms of ROM or RAM either currently known or later developed. Moreover, the various modules and screen display functionality is but one example of such functionality and any other configurations encompassing similar functionality are possible.

Furthermore, although specific components of innovations were described, methods, systems, and articles of manufacture consistent with the innovation may include additional or different components. For example, a processor may be implemented as a microprocessor, microcontroller, application specific integrated circuit (ASIC), discrete logic, or a combination of other type of circuits or logic. Similarly, memories may be DRAM, SRAM, Flash or any other type of memory. Flags, data, databases, tables, entities, and other data structures may be separately stored and managed, may be incorporated into a single memory or database, may be distributed, or may be logically and physically organized in many different ways. Programs may be parts of a single program, separate programs, or distributed across several memories and processors.

The respective logic, software or instructions for implementing the processes, methods and/or techniques discussed above may be provided on computer-readable media or memories or other tangible media, such as a cache, buffer, RAM, removable media, hard drive, other computer readable storage media, or any other tangible media or any combination thereof. The tangible media include various types of volatile and nonvolatile storage media. The functions, acts or tasks illustrated in the figures or described herein may be executed in response to one or more sets of logic or instructions stored in or on computer readable media. The functions, acts or tasks are independent of the particular type of instructions set, storage media, processor or processing strategy and may be performed by software, hardware, integrated circuits, firmware, micro code and the like, operating alone or in combination. Likewise, processing strategies may include multiprocessing, multitasking, parallel processing and the like. In one embodiment, the instructions are stored on a removable media device for reading by local or remote systems. In other embodiments, the logic or instructions are stored in a remote location for transfer through a computer network or over telephone lines. In yet other embodiments, the logic or instructions are stored within a given computer, central processing unit ("CPU"), graphics processing unit ("GPU"), or system.

While various embodiments of the innovation have been described, it will be apparent to those of ordinary skill in the art that many more embodiments and implementations are possible within the scope of the innovation. Accordingly, the innovation is not to be restricted except in light of the attached claims and their equivalents.

What is claimed is:

1. A storage system to form at least one snapshot of at least one storage volume comprising:
- a storage hardware interface;
- a memory; and
- at least one processor in communication with the memory and the storage hardware interface, the memory comprising:
  - a first solid state memory cache;
  - a second solid state memory cache, the first and second solid state memory caches being addressable with the at least one processor in a common address space; and
  - storage controller logic executable with the at least one processor to:
    - provide block-level access to the at least one storage volume over the storage hardware interface;
    - store all data blocks of the at least one storage volume in the first solid state memory cache; and
    - form the at least one snapshot of the at least one storage volume, all data blocks of the at least one snapshot being stored in the second solid state memory cache.

2. The storage system of claim **1**, wherein the storage controller logic executable to form the at least one snapshot is further executable to copy all data blocks of the at least one storage volume from the first solid-state memory cache to the second solid-state memory cache.

3. The storage system of claim **2**, wherein the storage controller logic is executable with the at least one processor to:

track dirty blocks written to the at least one storage volume during a time period in which all data blocks of the at least one storage volume are copied to the second solid-state memory cache;

stop acceptance of write requests, the write requests being for the at least one storage volume;

copy the dirty blocks to the second solid-state memory cache; and

resume the acceptance of write requests after the dirty blocks are copied.

4. The storage system of claim **3**, wherein the storage controller logic is executable with the at least one processor to track the dirty blocks based on adjusted region descriptors.

5. The storage system of claim **1**, further comprising a memory controller and a system bus, wherein the at least one processor and the memory controller are coupled to the system bus, the memory controller is coupled to the memory, and the at least one processor is in communication with the memory through the memory controller.

6. The storage system of claim **1**, further comprising a single block storage device, the single block storage device comprising the memory, the at least one processor, and the storage hardware interface.

7. The storage system of claim **1**, wherein the storage controller logic is executable with the at least one processor to provide the block-level access in accordance with a storage protocol.

8. A tangible computer-readable storage medium encoded with computer executable instructions, the computer executable instructions executable with at least one processor, the computer-readable medium comprising instructions executable to provide block-level access to a storage volume over a storage hardware interface in accordance with a storage protocol, the instructions executable to provide the block-level access including:

instructions executable to provide the storage volume in a first memory cache, the first memory cache comprising all data stored in the storage volume; and

instructions executable to form a snapshot of the storage volume in a second memory cache, the second memory cache comprising all data stored in the snapshot of the storage volume, the first and second memory caches being addressable by the at least one processor.

9. The tangible computer-readable storage medium of claim **8**, wherein the instructions executable to form the snapshot of the storage volume further comprise:

instructions executable to determine a first memory location at which at least one block of the storage volume is stored in the first memory cache;

instructions executable to determine a second memory location at which at least one block of the snapshot is stored in the second memory cache, the at least one block of the snapshot corresponding to the at least one block of the storage volume; and

instructions executable to perform a memory copy command, the memory copy command executable with the at least one processor to copy data stored at the first memory location to the second memory location.

10. The tangible computer-readable storage medium of claim **8**, wherein the instructions executable to form the snapshot of the storage volume further comprise:

instructions executable to reject write requests received after the snapshot of the storage volume is initiated but before the snapshot of the storage volume is complete; and

instructions executable to resume acceptance of write requests received after the snapshot of the storage volume is complete.

11. The tangible computer-readable storage medium of claim **8**, wherein the instructions executable to form the snapshot of the storage volume further comprise:

instructions executable to delay writes to the storage volume requested in write requests received after the snapshot of the storage volume is initiated; and

instructions executable to perform the writes after the snapshot of the storage volume is complete.

12. The tangible computer-readable storage medium of claim **8**, wherein the storage volume is a configured area of storage accessible at the block level via a storage protocol.

13. The tangible computer-readable storage medium of claim **8**, wherein the at least one processor is configured to address the first and second memory cache caches over a system bus.

14. The tangible computer-readable storage medium of claim **8**, wherein the instructions executable to form the snapshot of the storage volume further comprise instructions executable to provide block-level access to a snapshot volume, wherein all data blocks of the snapshot volume are mapped to the snapshot of the storage volume.

15. A computer-implemented method to form a snapshot volume from a storage volume, the method comprising:

providing block-level access to the storage volume over a storage hardware interface in accordance with a storage protocol;

providing the storage volume in a first memory cache, the first memory cache comprising all data of the storage volume;

initiating the snapshot of the storage volume with the at least one processor; and

copying at least a portion of the data of the storage volume from the first memory cache to a second memory cache with the at least one processor, the second memory cache comprising all data of the snapshot volume, the first and second memory caches being addressable by the at least one processor over a system bus.

16. The computer-implemented method of claim **15**, further comprising copying all data blocks of the snapshot volume, after the snapshot of the storage volume is complete, from the second memory cache to a storage medium having a slower write rate than the memory, the memory being solid state memory.

17. The computer-implemented method of claim **15**, wherein copying the at least a portion of the data of the storage volume comprises copying all data blocks of the storage volume from the first memory cache to the second memory cache in response to initiation of the snapshot of the storage volume.

18. The computer-implemented method of claim **17**, further comprising, with the at least one processor:

tracking any updated data blocks in the first memory cache updated during the copying of all data blocks of the storage volume from the first memory cache to the second memory cache;

ceasing to accept new write requests to the storage volume after all data blocks of the storage volume are copied;

copying any updated data blocks from the first memory cache to the second memory cache; and

resuming acceptance of new write requests after the updated data blocks are copied.

19. The computer-implemented method of claim **15**, further comprising:

receiving, with the at least one processor, a plurality of requests to write updated data blocks to the storage volume;

writing the updated data blocks with the at least one processor to the first memory cache in response to the requests;

indicating at least one memory region of the first memory cache that includes the updated data blocks is dirty by setting at least one dirty flag;

copying data blocks included in the at least one memory region from the first memory cache to the second memory cache and clearing the at least one dirty flag;

ceasing to accept new write requests to the storage volume in response to initiation of the snapshot of the storage volume if any of the at least one dirty flag is not yet cleared; and

resuming acceptance of new write requests with the at least one processor after all remaining data blocks in the at least one memory region are copied from the first memory cache to the second memory cache, the at least a portion of the data of the storage volume including the all remaining data blocks.

20. The computer-implemented method of claim **15**, wherein copying the at least a portion of the data blocks of the storage volume from the first memory cache to the second memory cache comprises copying data blocks of the storage volume subject to a write request received after the snapshot of the storage volume is initiated, the second memory cache overlapping the first memory cache except for a portion of the second memory cache in which the at least a portion of the data blocks of the storage volume are stored.

21. A computer-implemented method to form a snapshot volume from a storage volume, the method comprising:

storing the storage volume in a first memory cache with at least one processor, the first memory cache comprising all data of the storage volume, the storage volume being included in one storage device;

initiating the snapshot of the storage volume with the at least one processor; and

copying at least a portion of the data of the storage volume from the first memory cache to a second memory cache with the at least one processor, the second memory cache comprising all data of the snapshot volume, the first and second memory caches included in one area of memory in the one storage device.

* * * * *