

keep track of the upload status to ensure that all data parts have been received before a job record is created. In some embodiments, job tracker 230 also obtains a data object identifier associated with the data to be stored and provides the data object identifier, for example, to a front end service to be returned to a customer. In an embodiment, such data object identifier may be
5 obtained from data plane 214 services such as storage node manager 244, storage node registrar 248, and the like, described below.

[0075] In some embodiments, control plane for direct I/O 210 includes a job tracker store 232 for storing job entries or records. In various embodiments, job tracker store 230232 may be implemented by a NoSQL data management system, such as a key-value data store, a relational
10 database management system (“RDBMS”) or any other data storage system. In some embodiments, data stored in job tracker store 230232 may be partitioned to enable fast enumeration of jobs that belong to a specific customer, facilitate efficient bulk record deletion, parallel processing by separate instances of a service and the like. For example, job tracker store 230232 may implement tables that are partitioned according to customer account identifiers
15 and that use job identifiers as range keys. In an embodiment, job tracker store 230232 is further sub-partitioned based on time (such as job expiration time) to facilitate job expiration and cleanup operations. In an embodiment, transactions against job tracker store 232 may be aggregated to reduce the total number of transactions. For example, in some embodiments, a job tracker 230 may perform aggregate multiple jobs corresponding to multiple requests into one
20 single aggregated job before inserting it into job tracker store 232.

[0076] In an embodiment, job tracker 230 is configured to submit the job for further job scheduling and planning, for example, by services in common control plane 212. Additionally, job tracker 230 may be configured to monitor the execution of jobs and update corresponding job records in job tracker store 232 as jobs are completed. In some embodiments, job tracker 230
25 may be further configured to handle customer queries such as job status queries. In some embodiments, job tracker 230 also provides notifications of job status changes to customers or other services of the archival data storage system. For example, when a data retrieval job is completed, job tracker 230 may cause a customer to be notified (for example, using a notification service) that data is available for download. As another example, when a data storage job is

stored in storage node manager job store 240. In an embodiment, a storage node manager 244 scans incoming jobs and sort the jobs by the type of data operation (e.g., read, write or delete), storage locations (e.g., volume, disk), customer account identifier and the like. The storage node manager 244 may then reorder, coalesce, group in batches or otherwise manipulate and schedule the jobs for processing. For example, in one embodiment, the storage node manager 244 may batch process all the write operations before all the read and delete operations. In another embodiment, the storage node manager 224244 may perform operation coalescing. For another example, the storage node manager 224244 may coalesce multiple retrieval jobs for the same object into one job or cancel a storage job and a deletion job for the same data object where the deletion job comes after the storage job.

[0083] In an embodiment, storage node manager job store 240 is partitioned, for example, based on job identifiers, so as to allow independent processing of multiple storage node managers 244 and to provide even distribution of the incoming workload to all participating storage node managers 244. In various embodiments, storage node manager job store 240 may be implemented by a NoSQL data management system, such as a key-value data store, a RDBMS or any other data storage system.

[0084] In an embodiment, request balancer 238 provides a service for transferring job items from job request queue 236 to storage node manager job store 240 so as to smooth out variation in workload and to increase system availability. For example, request balancer 238 may transfer job items from job request queue 236 at a lower rate or at a smaller granularity when there is a surge in job requests coming into the job request queue 236 and vice versa when there is a lull in incoming job requests so as to maintain a relatively sustainable level of workload in the storage node manager store 240. In some embodiments, such sustainable level of workload is around the same or below the average workload of the system.

[0085] In an embodiment, job items that are completed are removed from storage node manager job store 240 and added to the job result queue 242. In an embodiment, data plane 214 services (e.g., storage node manager 244) are responsible for removing the job items from the storage node manager job store 240 and adding them to job result queue 242. In some

embodiments, job request queue 242 is implemented in a similar manner as job request queue ~~235~~236, discussed above.

[0086] Referring now to data plane 214 illustrated in FIG. 2. In various embodiments, data plane 214 provides services related to long-term archival data storage, retrieval and deletion, data management and placement, anti-entropy operations and the like. In various embodiments, data plane 214 may include any number and type of storage entities such as data storage devices (such as tape drives, hard disk drives, solid state devices, and the like), storage nodes or servers, datacenters and the like. Such storage entities may be physical, virtual or any abstraction thereof (e.g., instances of distributed storage and/or computing systems) and may be organized into any topology, including hierarchical or tiered topologies. Similarly, the components of the data plane may be dispersed, local or any combination thereof. For example, various computing or storage components may be local or remote to any number of datacenters, servers or data storage devices, which in turn may be local or remote relative to one another. In various embodiments, physical storage entities may be designed for minimizing power and cooling costs by controlling the portions of physical hardware that are active (e.g., the number of hard drives that are actively rotating). In an embodiment, physical storage entities implement techniques, such as Shingled Magnetic Recording (SMR), to increase storage capacity.

[0087] In an environment illustrated by FIG. 2, one or more storage node managers 244 each controls one or more storage nodes 246 by sending and receiving data and control messages. Each storage node 246 in turn controls a (potentially large) collection of data storage devices such as hard disk drives. In various embodiments, a storage node manager 244 may communicate with one or more storage nodes 246 and a storage node 246 may communicate with one or more storage node managers 244. In an embodiment, storage node managers 244 are implemented by one or more computing devices that are capable of performing relatively complex computations such as digest computation, data encoding and decoding, job planning and optimization and the like. In some embodiments, storage nodes ~~244~~246 are implemented by one or more computing devices with less powerful computation capabilities than storage node managers 244. Further, in some embodiments the storage node manager 244 may not be included in the data path. For example, data may be transmitted from the payload data cache 228

data storage system 1504) are to be activated and/or during what time periods. Additionally, the rotation schedule 1504 may be determined based at least in part on several factors and/or may be provided by the data storage node manager 224244 and/or the data storage node, as described above.

[0185] In some aspects, a controller 1506 may be configured to control one or more hard drives 1508(1), ..., 1508(N) (collectively, "hard drives 1508"). By way of example only, the hard drives 1508 are illustrated in an array or ordered list; however, any ordering and/or other organization may be envisioned for determining which particular hard drives 1508 or sets of hard drives come before or after other particular hard drives 1508 (or sets of hard drives). For example, the array may be utilized by the controller for pointing to a next hard drive to activate (e.g., based at least in part on some logical mapping) and not for ordering the hard drives 1508. However, in other examples, the array may be utilized to dynamically order the hard drives (e.g., when an on-demand rotation schedule is implemented).

[0186] In some examples, a window 1510 (or sliding scale) may be envisioned for activating the hard drives 1508. Based at least in part on a predefined period of time, the window 1510 may move along the array to become windows 1512, 1514 and 1516, for example, at each period. In other words, the window 1510 may represent the hard drives 1508 that are active or are to be activated by the controller 1506 during a first period. That is, by way of example and not limitation, during the first period, the hard drives 1508(1)-1508(5) may be active (i.e., provided power or taken out of sleep mode). In the example shown in FIG. 15, at the end of the first period, the window 1510 may move to become the window 1512, where hard drives 1508(2)-1508(6) become active. In some examples, when the transition occurs (i.e., the window 1510 slides), the hard drive 1508(1) may be powered down prior to the controller 1506 activating the hard drive 1508(6). In this way, the power draw associated with operating the array of hard drives 1508 may not exceed a certain level.

[0187] Additionally, in some aspects, the windows 1510, 1512, 1514, 1516 may be defined by a length 1518. The length may be determined based at least in part on demand, a percentage of the total hard drives 1508, a predetermined number of the total hard drives 1508, some predetermined level of power consumption, temperature, cost, etc. or the like. In addition, the window may be configured to minimize power consumption of the hard drives 1508, minimize operational costs associated with the hard drives 1508, and/or maximize the operational lifetime

the archival data storage service 1502 to modify the indication or provide an instruction to activate a different hard drive 1508 instead.

[0189] As described with reference to FIG. 15, in some aspects, the total number of hard drives 1508 may be divided into a predefined number of groups. Then, these groups may be rotated in and out of operation based at least in part on the rotation schedule 1504. Such an approach may allow the storage node registrars 248 to determine when and where to place new write operations or when it may be possible to execute read or delete operations. In some examples, the data storage node managers 224 may be isolated from this knowledge and may follow directions from the storage node registrars 248. However, in other examples, the rotation schedule 804 may be determined on-demand. For example, based at least in part on read requests from one or more data storage node managers 224, a storage node registrar 248 may select a number of the hard drives 808 that should come in to a rotation, and then send messages to the data storage nodes. Alternatively, the data storage node manager ~~224~~244 may directly contact the data storage nodes. In some examples, particular constraints may influence the rotation schedule 1504 including, but not limited to, a goal not to starve, for too long, a request to a drive which has only one read request outstanding.

[0190] FIGS. 16-18 illustrate example flow diagrams showing respective processes 1600-1800 for providing data storage power management. These processes are illustrated as logical flow diagrams, each operation of which represents a sequence of operations that can be implemented in hardware, computer instructions, or a combination thereof. In the context of computer instructions, the operations represent computer-executable instructions stored on one or more computer-readable storage media that, when executed by one or more processors, perform the recited operations. Generally, computer-executable instructions include routines, programs, objects, components, data structures and the like that perform particular functions or implement particular data types. The order in which the operations are described is not intended to be construed as a limitation, and any number of the described operations can be combined in any order and/or in parallel to implement the processes.

[0191] Additionally, some, any or all of the processes may be performed under the control of one or more computer systems configured with executable instructions and may be implemented as code (e.g., executable instructions, one or more computer programs, or one or more

CLAIMS

WHAT IS CLAIMED IS:

- 1 1. A computer-implemented method comprising:
2 under the control of one or more computer systems configured with executable
3 instructions,
4 receiving a data retrieval request to retrieve a data object, the data retrieval
5 request specifying a data object identifier;
6 creating a data retrieval job corresponding to the data retrieval request;
7 providing a job identifier associated with the data retrieval job that is usable for
8 obtaining information about the data retrieval job; and
9 after providing the job identifier, processing the data retrieval job using at least in
10 part the data object identifier to provide access to the data object.
- 1 2. The computer-implemented method of claim 1, wherein the data object
2 identifier is provided in response to a previous storage request to store the data object.
- 1 3. The computer-implemented method of claim 1, wherein processing the
2 data retrieval job comprises:
3 selecting the data retrieval job for execution;
4 determining, based at least in part on the data object identifier, one or more
5 storage entities for storing one or more encoded data components, the one or more encoded data
6 components are generated based at least in part on the data object;
7 causing retrieval of at least some of the one or more encoded data components
8 from the determined storage entities; and
9 decoding the retrieved encoded data components to obtain the retrieved data
10 object.
- 1 4. The computer-implemented method of claim 3, wherein selecting the data
2 retrieval job is based at least in part on a batch processing schedule.
- 1 5. The computer-implemented method of claim 1, further comprising
2 providing a status of the data retrieval job in response to a job status request that specifies the job
3 identifier.

1 6. A system for providing archival data storage services, comprising:
2 one or more archival data storage devices;
3 a transient data store;
4 one or more processors;
5 memory, including executable instructions that, when executed by the one or
6 more processors, cause the one or more processors to collectively at least:
7 receive a data storage request to store a data object;
8 cause storage of the data object in the transient data store;
9 provide a data object identifier associated with the data, the data object identifier
10 encoding at least storage location information usable to locate the data object; and
11 after providing the data object identifier, cause storage of the data object in
12 location specified by the storage location information.

1 7. The system of claim 6, wherein the executable instructions, when executed
2 by the one or more processors, further cause the one or more processors to collectively create a
3 data storage job corresponding to the data storage request and wherein causing storage of the
4 data object in location specified by the storage location information comprises processing the
5 data storage job based at least in part on the storage location information.

1 8. The system of claim 6, wherein causing storage of the data object in
2 location specified by the storage location information comprises:
3 obtaining the data object from the transient data store;
4 obtaining one or more data encoding schemes;
5 encoding the data object with the one or more data encoding schemes to obtain a
6 plurality of encoded data components; and
7 causing storage of the plurality of encoded data components in at least some of
8 the one or more archival data storage devices.

1 9. The system of claim 6, wherein the data object identifier encodes at least
2 data validation information usable to validate integrity of the data object.

1 10. The system of claim 9, wherein the executable instructions, when executed
2 by the one or more processors, further cause the one or more processors to collectively validate
3 integrity of the data object based at least in part on the data validation information

1 11. The system of claim 6, wherein the executable instructions, when executed
2 by the one or more processors, further cause the one or more processors to:

3 receive a data retrieval request for the data object cause aggregation of the data
4 retrieval request with other data retrieval requests to form a request batch; and
5 cause validation of the request batch based at least in part on data object
6 identifiers corresponding to the request batch.

1 12. The system of claim 6, wherein the executable instructions, when executed
2 by the one or more processors, further cause the one or more processors to receive a data
3 retrieval request for the data object; and
4 cause processing of the data retrieval request based at least in part on a power
5 management schedule.

1 13. The system of claim 6, wherein the executable instructions, when executed
2 by the one or more processors, further cause the one or more processors to cause partitioning of
3 pending data retrieval requests to facilitate parallel processing.

1 14. The system of claim 6, wherein the executable instructions, when executed
2 by the one or more processors, further cause the one or more processors to provide status of a
3 data retrieval request in response to a status request that specifies the data object identifier.

1 15. The system of claim 6, wherein the executable instructions, when executed
2 by the one or more processors, further cause the one or more processors to cause validation of
3 data integrity for the data object.