

**PARALLEL ACCESS TO DATA IN A DISTRIBUTED FILE
SYSTEM**

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to U.S. Application Serial No. 14/090,434,
5 filed on November 26, 2013.

BACKGROUND

This invention relates to parallel access to data in a distributed file system.

An example of a distributed file system is the Hadoop Distributed File System (HDFS). HDFS is a distributed, scalable, and portable file system written in Java.

10 The HDFS has a set of nodes ("datanodes") that hold data for the files in the file system, and can server up blocks of the files over a data network. Each file is generally distributed over multiple nodes. A directory for the file system is maintained by a set of nodes ("namenodes"). This directory can be used to identify the locations of the distributed blocks for each named file in the filesystem.

15 Referring to FIGS. 1A-B, one way of processing data in a distributed file system, for example in the HDFS, is using a MapReduce programming model. Generally, a MapReduce program includes a Map procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce procedure that performs a summary operation (such as counting
20 the number of students in each queue, yielding name frequencies). A user of the system specifies the Map and Reduce procedures, but does not necessarily determine the number of instances (or invocations) of each procedure (i.e., "processes") or the nodes on which they execute. Rather, a "MapReduce System" (also called "infrastructure", "framework") orchestrates by marshaling a set of distributed nodes,
25 running the various tasks (e.g., the Map and Reduce procedures and associated communication) in parallel, managing all communications and data transfers between the various parts of the system, providing for redundancy and failures, and overall management of the whole process. A MapReduce system can schedule execution of instances of Map or Reduce procedures with an awareness of the data location. In
30 FIG. 1A, the HDFS is an example of a distributed filesystem 110 and the Hadoop framework is an example of a distributed processing system, such that a set of map procedures 24 execute on processors 122 in parallel to process parts 14 of a

distributed file and store output files 16 in the filesystem 110. In FIG. 1B a set of reduce procedures 26 process the outputs of the map procedures to produce output files 18 completing the MapReduce execution.

Although there are computational advantages that can be gained using a distributed file system that is integrated with a MapReduce system, access to data for processing with other programming paradigms can be inefficient. For example, extraction of data may result in a bottleneck, for example, passing all the data for a file through a single data stream, even if the processing of the data is itself parallel. For example, the HDFS provides a filesystem command (`hadoop fs -cat file`) which passes the content of the *file* to the standard output and potentially streamed to a process or server where the contents may be processed in parallel.

One approach to avoiding this type of bottleneck is to explicitly divide large files into separate named parts (i.e., with individual entries in the directory), and then cause each part to be extracted in a separate stream. However, such approaches may have disadvantages, for instance, requiring a prior decision as to the number of parts, and potential suboptimal selection of nodes on which to extract the parts, for example, because the named parts are themselves distributed.

Another approach to providing data from a distributed filesystem to an application, for instance to a database engine, is to implement the application within or tightly coupled to the filesystem. However, such an approach can be limited to processing using a specific application, and does not necessarily benefit from advances in implementations of the application that are not ported into the filesystem.

SUMMARY

In one aspect, in general, a new approach to parallel extraction of data from a distributed filesystem provides parallel access to one or more named units (e.g., files) in the filesystem by creating multiple parallel data streams such that all the data of the desired units is partitioned over the multiple streams. In some examples, the multiple streams form multiple inputs to a parallel implementation of a computation system, such as a graph-based computation system, dataflow-based system, and/or a (e.g., relational) database system.

One implementation of the approach makes use of a supervisory or manager function of the file storage system that accepts a specification of one or more named units (e.g., files) in the file system and a specification of a set of destinations for the

content of the specified units. The supervisory function then creates multiple executing instances of an extraction procedure. Each instance of the extraction procedure is configured by the supervisory function to access a part of the desired content, and is configured to transfer that content to one of the set of specified destinations, for example, using a data streaming (e.g., TCP streams, local named pipes, etc.). In some examples, the supervisory function also receives other information such as a desired number of (or limits on) instances of the extraction procedures, particular computational procedures to execute on the extracted content before transmission to the destinations, or information regarding desired locations (e.g., nodes) at which the extraction procedures are to be hosted.

A variety of approaches can be used to specify the set of destinations and to establish the connections between the extraction procedures and the consumers of the data from those procedures. In one example, the set of destinations has a name, and each extraction procedure is configured with a way to access (e.g., a network address of) a shared “broker”. Before transferring data, the extraction procedure communicates with the broker, which provides a specific location (e.g., a network address) to the destination of the content from the requesting extraction process.

One implementation of the extraction procedure makes use of a standard Map function (e.g., with no Reduce stage). A standard supervisory function determines where to execute instances of the Map and the segments of the desired files that are processed by the Map. The specific procedure performed by the map (e.g., a user specified output filter of the Map) then streams the content to the desired destination rather than causing it to be stored on the file system.

In some versions of the system, instances of the processes that are consumers or potential consumers of the extracted content execute on the same computation nodes that are available to host the extraction procedures, and a decision process implemented by the broker may attempt to co-locate the extraction procedures and the consumers of the data extracted by the extraction procedures, thereby permitting local transfer of the content for processing (e.g., through in-memory messaging, UNIX named pipes, etc.). In some such versions, the broker selects destinations for the extracted data from each extraction procedure, for instance, by selecting a destination process that is co-located on a same computation node as the extraction procedure.

In some implementations, certain processing may be efficiently implemented within or tightly linked with the extraction procedures. For example, certain data

filtering, format restructuring, or in some cases certain aggregation tasks, may be hosted within the extraction procedure.

In one aspect, in general, a method for processing data includes: receiving a specification of one or more named units stored in a distributed filesystem of a distributed processing system; receiving a specification for establishing data connections to a computation system separate from the distributed processing system; invoking a first plurality of processes on the distributed processing system, each process of the first plurality of processes establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem; using the specification for establishing the data connections to form a plurality of data connections between the distributed processing system and the computation system, at least one data connection being formed between each process of the first plurality of processes and the computation system; and passing data concurrently over the plurality of data connections from the distributed processing system to the computation system.

Aspects can include one or more of the following features.

Passing the data concurrently includes extracting the data from the filesystem by the processes of the first plurality of processes, and transmitting the extracted data from said processes to the computation system without further storage in the distributed file system.

Accepting the specification of the named units includes accepting a text filename of the named unit.

The parts of the named units are unnamed.

Accepting the specification for establishing the data connections includes accepting a specification of a broker and forming the data connections includes each processes of the first plurality of processes communicating with the broker to determine a further specification for establishing a data connection with the computation system.

Accepting the specification for establishing the data connections includes accepting a specification for a broadcast to the computation system.

The computation system includes a second plurality of processes, and wherein forming the data connection between each process of the first plurality of processes and the computation system includes forming data connections between the processes of the first plurality of processes and processes of the second plurality of processes.

The method further includes invoking the processes of the second plurality of processes of the computation system.

At least some of the first plurality of processes and some of the second plurality of processes are executed on a common set of processors.

5 The distributed processing system and the computation system use distinct computing resources coupled over a data network.

The distributed processing system and the computation system share computing resources.

10 In another aspect, in general, software is stored on a non-transitory computer-readable medium, for processing data. The software includes instructions for causing a system to: receive a specification of one or more named units stored in a distributed filesystem of a distributed processing system; receive a specification for establishing data connections to a computation system separate from the distributed processing system; invoke a first plurality of processes on the distributed processing system, each
15 process of the first plurality of processes establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem; use the specification for establishing the data connections to form a plurality of data connections between the distributed processing system and the computation system, at least one data connection being formed
20 between each process of the first plurality of processes and the computation system; and pass data concurrently over the plurality of data connections from the distributed processing system to the computation system.

25 In another aspect, in general, a system for processing data includes: a distributed processing system that includes a distributed filesystem; and a computation system separate from the distributed processing system. The distributed processing system is configured to: receive a specification of one or more named units stored in the distributed filesystem; receive a specification for establishing data connections to the computation system; invoke a first plurality of processes, each
30 process of the first plurality of processes establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem; use the specification for establishing the data connections to form a plurality of data connections between the distributed processing system and the computation system, at least one data connection being formed
between each process of the first plurality of processes and the computation system;

and pass data concurrently over the plurality of data connections to the computation system.

In another aspect, in general, a method for processing data includes: providing a specification of one or more named units stored in a distributed filesystem;

5 providing a specification for establishing data connections with a computation system; providing a specification for processes of a first plurality of processes for invocation on a distributed processing system, each process of the first plurality of processes being specified for establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the

10 distributed filesystem; receiving requests to form a plurality of data connections between the distributed processing system and the computation system, and providing information for forming at least one data connection being between each process of the first plurality of processes and the computation system; and receiving data concurrently over the plurality of data connections from the first plurality of processes

15 at the computation system.

Aspects can include one or more of the following features.

The method further includes invoking a second plurality of processes in the computation system, wherein the data connections are formed between the processes of the first plurality of processes and processes of the second plurality of processes.

20 Invoking the second plurality of processes includes invoking at least some of said processes on processors hosting the first plurality of processes, and wherein at least some of the data connections are local said processors.

Providing the information for forming a data connection includes selecting an endpoint for the data connection from a plurality of available endpoints according to a

25 location of a requesting process and locations of the available endpoints.

In another aspect, in general, software is stored on a non-transitory computer-readable medium, for processing data. The software includes instructions for causing a system to: provide a specification of one or more named units stored in a distributed filesystem; provide a specification for establishing data connections with a

30 computation system; provide a specification for processes of a first plurality of processes for invocation on a distributed processing system, each process of the first plurality of processes being specified for establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem; receive requests to form a plurality of data

connections between the distributed processing system and the computation system, and provide information for forming at least one data connection being between each process of the first plurality of processes and the computation system; and receive data concurrently over the plurality of data connections from the first plurality of processes at the computation system.

In another aspect, in general, a system for processing data includes: a distributed filesystem; a distributed processing system; a computation system; and a client of the distributed processing system configured to: provide a specification of one or more named units stored in the distributed filesystem; provide a specification for establishing data connections with the computation system; provide a specification for processes of a first plurality of processes for invocation on the distributed processing system, each process of the first plurality of processes being specified for establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem; receive requests to form a plurality of data connections between the distributed processing system and the computation system, and provide information for forming at least one data connection being between each process of the first plurality of processes and the computation system; and receive data concurrently over the plurality of data connections from the first plurality of processes at the computation system.

In another aspect, in general, a method for processing data, the data being provided from a distributed processing system implementing a map-reduce data processing framework, includes: providing to the distributed processing system a specification for a map procedure for invocation on the distributed processing system, the specification for the map procedure identifying one or more named units in a distributed filesystem for processing and including a specification for establishing data connections with a computation system separate from the distributed processing system; causing execution of a plurality of instances of the map procedure on the distributed processing system; receiving requests to form a plurality of data flow connections between executing instances of the map procedure and the computation system, and providing information for forming at least one data flow connection being between each executing instance of the map procedure and the computation system; and receiving data concurrently over the plurality of data flow connections and processing the received data in the computation system.

Aspects can include one or more of the following features.

The distributed processing system includes an Apache Hadoop ® system.

The data flow connections include transport layer data network communication connections.

In another aspect, in general, software is stored on a non-transitory computer-readable medium, for processing data, the data being provided from a distributed processing system implementing a map-reduce data processing framework. The software includes instructions for causing a system to: provide to the distributed processing system a specification for a map procedure for invocation on the distributed processing system, the specification for the map procedure identifying one or more named units in a distributed filesystem for processing and including a specification for establishing data connections with a computation system separate from the distributed processing system; cause execution of a plurality of instances of the map procedure on the distributed processing system; receive requests to form a plurality of data flow connections between executing instances of the map procedure and the computation system, and provide information for forming at least one data flow connection being between each executing instance of the map procedure and the computation system; and receive data concurrently over the plurality of data flow connections and process the received data in the computation system.

In another aspect, in general, a system for processing data includes: a distributed filesystem; a distributed processing system; a computation system separate from the distributed processing system; and a client of the distributed processing system configured to: provide to the distributed processing system a specification for a map procedure for invocation on the distributed processing system, the specification for the map procedure identifying one or more named units in the distributed filesystem for processing and including a specification for establishing data connections with the computation system; cause execution of a plurality of instances of the map procedure on the distributed processing system; receive requests to form a plurality of data flow connections between executing instances of the map procedure and the computation system, and provide information for forming at least one data flow connection being between each executing instance of the map procedure and the computation system; and receive data concurrently over the plurality of data flow connections and processing the received data in the computation system.

One or more aspects of the approach address the technical problem of efficient access to data stored in a distributed filesystem. A technical solution of invoking

multiple instances of an extraction procedure (e.g., under the control of supervisor or manager), each providing a stream of data to a computation system, permits the multiple instances of the extraction procedure to have efficient parallel access to the data. This solution has a technical advantage of avoiding a bottleneck in extraction of the data, for example using a native filesystem command (e.g., the “hadoop fs -cat *file*” command). By avoiding the bottleneck, it can be feasible to perform data processing in a separate computation system, thereby not being limited to the forms of computation (e.g., MapReduce) that may be available on the filesystem.

One or more aspects provide computational advantages over a conventional Hadoop system. Ordinarily, Hadoop works on a MapReduce paradigm where the outputs of the Map jobs as well as the outputs of the Reduce jobs are written to files. While this is highly parallel, it suffers from input/output challenges due to multiple file writes. Ordinarily, this is tolerated because of the benefits of parallelism that people get with Hadoop. Nevertheless, the benefits of parallelism can be achieved without the input/output cost by the approaches described herein. In some examples, efficiency is achieved by intercepting the output of the Map jobs in parallel and, without writing those results to files and instead transmitting them via TCP/IP to a parallel operating system that is capable of processing the data in parallel. Such a method thereby achieves both parallel speed and no file input/output cost. Another benefit is that MapReduce is relatively inflexible and not nimble, and does not join well with other databases, and by bringing the data out of the MapReduce paradigm, additional flexibility, interoperability, efficiency, speed, etc. can be obtained.

Other features and advantages of the invention will become apparent from the following description, and from the claims.

DESCRIPTION OF DRAWINGS

FIGS 1A and 1B are block diagrams illustrating map and reduce operations, respectively.

FIG. 2 is a block diagram of a system for providing parallel access to a distributed file system.

FIG. 3 is a block diagram of the system that illustrates control communication for establishing the parallel access to the filesystem.

DESCRIPTION

Referring to FIG. 2, a computing system 100 includes a distributed file system 110, a distributed processing system 120, and also includes or has access to a computation system 130. One example of a file system 110 of this type is a Hadoop Distributed File System (HDFS), and the distributed processing system 120 is the Hadoop framework, but it should be understood that the approaches described herein are not limited to use in conjunction with a HDFS. The distributed file system 110 includes storage for a number of named units, which are referred to below as “files” without intending to connote particular attributes with the word “file.” In general, the names of the files may include paths that reference containing units, such as folders. In general, each file may have portions of it stored on different data storages 112 (e.g., disk subsystems) of the file system.

In some implementations, the approaches described above make use of a Hadoop framework to cause execution of copies of map procedures 124 in parallel on processors 122, such that the map procedures 124 efficiently access parts 114 of a named file in the Hadoop filesystem. However, rather than using the Hadoop framework to cause the map procedures to store intermediate files in the Hadoop filesystem, in a manner as illustrated in FIG. 1A, for further processing in a reduce phase in a manner illustrated in FIG. 1B, the map procedures 124 are configured to stream data determined from the file parts 114 directly to a computation system 130 in multiple separate streams without requiring intermediate data to be stored in the Hadoop filesystem. The Hadoop system can determine the number of and appropriate processors 122 on which to execute the map procedures 124 should execute, thereby providing efficient access to the parts 114. In such implementations, because the data is passed directly from the map procedures, the reduce procedures can be omitted. Note that the description below is not limited to the implementations that make use of a Hadoop infrastructure.

One function that is supported by the computing system 100 is a parallel extraction of the contents of one or a collection of named units in the file system 110 for processing in the computation system 130. In general, the computation system 130 has the capability of receiving and processing multiple streams of data more efficiently than if all the data were passed through a single stream. In some examples, these streams make use of TCP/IP based sessions over a local area network, and other forms of data transfer (e.g., using Unix named pipes) are used in other examples.

Note that the reference to the data being transferred as “streams” should be understood in its generality, and not to connote any particular mode of data transfer (e.g., “streaming”).

A representative named unit is represented in FIG. 2 as having parts 114
5 resident in multiple of the data storages 112. The parallel extraction function provides a transfer of data in these parts 114 to processes 134 in the computation system 130. For example, each process 134 may be able to access one or more data streams. So in some cases, there may be a one-to-one correspondence between parts 114 of the data and processes 134, but more generally there may be a many-to-one relationship
10 between the parts and the processes. In some embodiments, a named unit stored in the distributed file system 110 is made up of a set of subunits, which for the sake of description are referred to here as “records” without intending to connote particular characteristics to these subunits. Each part 114 of the named unit in the file system is made up of one or more of these records. In the discussion below, these records are
15 treated as being unordered. However, the records may be ordered being associated with (e.g., stored in conjunction with the records) record numbers that can be used to reconstruct the order for the entire file.

The distributed processing system 120 has a number of separate processors
122 (e.g., physical or virtual computation nodes, servers, and/or other units providing
20 computation services). Each processor has access to one or more of the data storages 112 (e.g., physical or logical disks, multiple disk subsystems, etc.). In general, a particular processor 122 can access one or more of the data storages 112 more efficiently than others. For example, a processor 122 may be able to access a storage 112 that has a local data connection to that processor more efficiently than it is able to
25 access a storage 112 that requires the data to be transferred over a local area network. During performance of the extraction function, an extraction process 124 (e.g., an operating system process or a computation unit within an operating system process) is executed on each or multiple of the processors 122 of the distributed processing system. Each process accesses one or more parts 114 of the file to be extracted, and
30 contents of that part (e.g., the records) pass over data connection to one of the processes 134 of the computation system 130.

Referring to FIG. 3, an embodiment of the parallel extraction function makes use of a number of elements that coordinate the invocation of processes involved in the extraction process and aid in the establishing of data connections between these

processes. FIG. 3 shows the procedures and other component that are used to coordinate establishing connections between the processes 124 (e.g., Hadoop map procedures) and processes 134, where data provided by the processes 124 is further processed. The dotted lines illustrate paths of control communication used in

5 establishing the connections. In this embodiment, a coordinator 142 is responsible for causing the processing by a number of processes 134 of data for a particular file or file collection that is resident in the distributed file system 110. The coordinator 142 causes the processes 134 to be invoked (e.g., as operating system processes or other computation units within operating system processes). The coordinator 142 also

10 passes information to each invoked process 134 so that that process can contact a broker 144, and each invoked process 134 passes information, including information about its location (e.g., identification of the processor hosting the process), to the broker. In some implementation, the broker 144 is also invoked by the coordinator 142, or the functions of the coordinator 142 and broker 144 are combined into one

15 unit. The coordinator 142 then communicates to a manager 115 of the distributed file system 110 and associated processing system 120. The coordinator 142 passes the name of the file or file collection to be processed, as well as the location of the broker 144, to the manager 115.

At this time, or in a prior configuration, the coordinator 142 identifies (e.g., by

20 name, code) to the manager 115 the procedure to be invoked on the processors 120 to extract the parts of the named file. The manager 115 determines where the parts 114 (see FIG. 1) of the named file are resident, and based on this determines on which processors 120 (see FIG. 1) to invoke the procedure as one of the processes 124. The invoked processes 124 are informed at the time they are invoked or through

25 subsequent communication with the manager 115 as to how to contact the broker 144. Each process 124 contacts the broker requesting information, such as an address, at which to contact one of the processes 134 that will receive the data. In at least some implementations, the broker selects one of processes 134 according to the location of the process, for example, to reduce communication overhead in the data transfer from

30 the requesting extraction process 124 and the receiving process 134 (e.g., by co-locating the processes on one processor), and sends the address information for the selected processes 134 back to the extraction process 124. The processes 124 then contacts the process 134 to establish the data connection.

At this point the data connections between the data storages 112, the processes 124, and the processes 134, (shown in bold lines in FIG. 2) are established, and the processes 124 begin to transfer data in parallel from the distributed file system 110 to the processes 134 of the computation system 130.

5 Note that the function of an explicit broker 144 is not necessary in all implementations. For example, the processes 124 can use a network broadcast approach to invite processes 134 to respond to form the connections, and the processes 134 can respond individually, or coordinate as a group to respond to the processes 124 to form the connections.

10 In one specific implementation in which the distributed file system is a Hadoop Distributed File System (HDFS), the manager 115 is implemented as a native part of a MapReduce System. In this implementation, the coordinator 142 (which is not native to the MapReduce System) specifies the file name and a Map function to execute, without specifying a corresponding Reduce processing. The manager 115
15 follows conventional MapReduce processing to orchestrate the execution of the specified Map function as the processes 124 according to its awareness of the locations of the parts of the named file in the filesystem. The specification of the Map function incorporates the information necessary for each instance of the Map function to communicate with the broker 144, and thereby forms the data connection with a
20 process 134 of the computation system 130.

 It should be understood that the distributed processing system 120 and the computation system 130 are not necessarily hosted on different computer processing nodes. For example, as introduced above, a common set of processors (e.g., processing nodes, servers) may be used by the manager 115 to host the processes 124,
25 and be used by the coordinator 142 to host the processes 134, which actually process the data. In some examples, when the communication between a process 124 and a process 134 does not pass between nodes, an efficient data passing approach can be used, for example, using Unix named pipes, message passing through shared memory, etc. Note that using separate extraction processes 124 and processes 134 which
30 receive the data permits different types of software to be executed while maintaining efficient communication between them. For example, the extraction processes may be implemented in Java as separate tasks within a Java Virtual Machine (JVM) while a process 134 may be implemented as an operating system process, for example, implemented in C or C++.

The computation system 130 may have various forms. As one example, the system implements a dataflow graph that include vertices (representing data processing components) connected by directed links (representing flows of work elements, i.e., data) between the vertices. For example, such an environment is
5 described in more detail in U.S. Publication No. 2007/0011668, titled “Managing Parameters for Graph-Based Applications,” incorporated herein by reference. A system for executing such graph-based computations is described in U.S. Patent 5,966,072, titled “EXECUTING COMPUTATIONS EXPRESSED AS GRAPHS,” incorporated herein by reference. In this example, the processes 134 may implement
10 a subset of the components of the dataflow graph, and provide outputs to yet other components. For example, a dataflow graph may include a component that is configured to execute in parallel with multiple instances of that component executing on different computation nodes. Each of the processes 134 may be executing as one of the instances of a parallel component that reads data from the distributed
15 processing system 120.

As other examples, the processes 134 may form part of a database system. For example, the parts 114 of a file may represent parts of a database table, and the processes 134 are configured to process a database table in parallel, for example, as part of executing a query involving one or more tables.

20 In some implementations, the function of the manager 115 is combined with the coordinator 142, and optionally the functions of the extraction processes 124 and destination processes 134 may be combined. In one such implementation, the coordinator/manager uses the name of the file or file collection to access a catalog in the distributed file system to determine the storages 112 on which the parts of those
25 files are stored, and optionally to determine the record structure of the data in those files. The coordinator/manager then invokes the extraction procedures 124, or combine extraction and processing procedures on the appropriate processors selected in the same or similar manner as described above for the manager 115. In some
30 implementations, the same named file or file collection is first processed to build and store an index of records in the file. Later, when the coordinator/manager wishes to extract a subset of the records matching a query, the index is read in and the extraction procedures are invoked on the processors so that less than the whole file or file collection needs to be read, for example, reading only the matching records of the file or file collection, thereby increasing the efficiency of the extraction function.

Note that there are many alternatives for implementation of the approaches described above. In particular, the components that are referred to as “processes” are not necessarily implemented as operating system processes (e.g., executed in a Unix environment with an `exec()` system call). Alternatives include lightweight processes (e.g., threads) within a context of an operating system process, applications executing with a virtual machine environment (e.g., Java applications or applets executing with a Java Virtual Machine (JVM)), tasks explicitly scheduled within the context of a process without the use of operating system primitives such as threads, or virtual machines executing within a context of hypervisors.

In some embodiments, the processes 124 executed in the distributed processing system 120 are not necessarily limited to transfer of data between the filesystem 110 and the computation system 130. In some examples, these processes are further configured (e.g., by the coordinator 142, see FIG. 2) to perform certain computations on the data. For example, the each process 124 may extract and/or transform fields of records from retrieved from the filesystem.

The approach described above can be implemented using a computing system executing suitable software. For example, the software may include procedures in one or more computer programs that execute on one or more programmed or programmable computing system (which may be of various architectures such as distributed, client/server, or grid) each including at least one processor, at least one data storage system (including volatile and/or non-volatile memory and/or storage elements), at least one user interface (for receiving input using at least one input device or port, and for providing output using at least one output device or port). The software may include one or more modules of a larger program, for example, that provides services related to the design, configuration, and execution of dataflow graphs. The modules of the program (e.g., elements of a dataflow graph) can be implemented as data structures or other organized data conforming to a data model stored in a data repository.

The software may be provided on a tangible, non-transitory medium, such as a CD-ROM or other computer-readable medium (e.g., readable by a general or special purpose computing system or device), or delivered (e.g., encoded in a propagated signal) over a communication medium of a network to a tangible, non-transitory medium of a computing system where it is executed. Some or all of the processing may be performed on a special purpose computer, or using special-purpose hardware,

such as coprocessors or field-programmable gate arrays (FPGAs) or dedicated, application-specific integrated circuits (ASICs). The processing may be implemented in a distributed manner in which different parts of the computation specified by the software are performed by different computing elements. Each such computer
5 program is preferably stored on or downloaded to a computer-readable storage medium (e.g., solid state memory or media, or magnetic or optical media) of a storage device accessible by a general or special purpose programmable computer, for configuring and operating the computer when the storage device medium is read by the computer to perform the processing described herein. The inventive system may
10 also be considered to be implemented as a tangible, non-transitory medium, configured with a computer program, where the medium so configured causes a computer to operate in a specific and predefined manner to perform one or more of the processing steps described herein.

A number of embodiments of the invention have been described.
15 Nevertheless, it is to be understood that the foregoing description is intended to illustrate and not to limit the scope of the invention, which is defined by the scope of the following claims. Accordingly, other embodiments are also within the scope of the following claims. For example, various modifications may be made without departing from the scope of the invention. Additionally, some of the steps described
20 above may be order independent, and thus can be performed in an order different from that described.

What is claimed is:

1. A method for processing data, the method including:
receiving a specification of one or more named units stored in a distributed filesystem of a distributed processing system;
receiving a specification for establishing data connections to a computation system separate from the distributed processing system;
invoking, by a manager of the distributed processing system, a first plurality of processes on the distributed processing system, each process of the first plurality of processes establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem;
using the specification for establishing the data connections to form a plurality of data connections between the distributed processing system and the computation system, at least one data connection being formed between each process of the first plurality of processes on the distributed processing system and a respective process of a second plurality of processes on the computation system, the first and second plurality of processes being separate from the manager of the distributed processing system; and
passing data concurrently over the plurality of data connections from the distributed processing system to the computation system.
2. The method of claim 1 wherein passing the data concurrently includes extracting the data from the filesystem by the processes of the first plurality of processes, and transmitting the extracted data from said processes to the computation system without further storage in the distributed file system.
3. The method of claim 1 wherein accepting the specification of the named units includes accepting a text filename of the named unit.

4. The method of claim 1 wherein the parts of the named units are unnamed.
5. The method of claim 1 wherein accepting the specification for establishing the data connections includes accepting a specification of a broker and forming the data connections includes each processes of the first plurality of processes communicating with the broker to determine a further specification for establishing a data connection with the computation system.
6. The method of claim 1 wherein accepting the specification for establishing the data connections includes accepting a specification for a broadcast to the computation system.
7. The method of claim 1 further including invoking the processes of the second plurality of processes of the computation system.
8. The method of claim 1 wherein at least some of the first plurality of processes and some of the second plurality of processes are executed on a common set of processors.
9. The method of claim 1 wherein the distributed processing system and the computation system use distinct computing resources coupled over a data network.
10. The method of claim 1 wherein the distributed processing system and the computation system share computing resources.
11. Software stored on a non-transitory computer-readable medium, for processing data, the software including instructions for causing a system to:
 - receive a specification of one or more named units stored in a distributed filesystem of a distributed processing system;
 - receive a specification for establishing data connections to a computation system separate from the distributed processing system;

invoke, by a manager of the distributed processing system, a first plurality of processes on the distributed processing system, each process of the first plurality of processes establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem;

use the specification for establishing the data connections to form a plurality of data connections between the distributed processing system and the computation system, at least one data connection being formed between each process of the first plurality of processes on the distributed processing system and a respective process of a second plurality of processes on the computation system, the first and second plurality of processes being separate from the manager of the distributed processing system; and

pass data concurrently over the plurality of data connections from the distributed processing system to the computation system.

12. A system for processing data, the system including:

a distributed processing system that includes a distributed filesystem; and

a computation system separate from the distributed processing system;

wherein the distributed processing system is configured to:

receive a specification of one or more named units stored in the distributed filesystem;

receive a specification for establishing data connections to the computation system;

invoke, by a manager of the distributed processing system, a first plurality of processes on the distributed processing system, each process of the first plurality of processes establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem;

use the specification for establishing the data connections to form a plurality of data connections between the distributed processing system and the computation system, at least one data connection being formed between each process of the first plurality of processes on the distributed processing system and a respective process of a second plurality of processes on the computation system, the first and second plurality of processes being separate from the manager of the distributed processing system; and

pass data concurrently over the plurality of data connections to the computation system.

13. A method for processing data, the method including:

providing a specification of one or more named units stored in a distributed filesystem;

providing a specification for establishing data connections with a computation system;

providing a specification for processes of a first plurality of processes for invocation on a distributed processing system by a manager of the distributed processing system, each process of the first plurality of processes being specified for establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem;

receiving requests to form a plurality of data connections between the distributed processing system and the computation system, and providing information for forming at least one data connection being between each process of the first plurality of processes on the distributed processing system and a respective process of a second plurality of processes on the computation system, the first and second plurality of processes being separate from the manager of the distributed processing system; and

receiving data concurrently over the plurality of data connections from the first plurality of processes at the computation system.

14. The method of claim 13 wherein invoking the second plurality of processes includes invoking at least some of said processes on processors hosting the first plurality of processes, and wherein at least some of the data connections are local said processors.

15. The method of claim 13 wherein providing the information for forming a data connection includes selecting an endpoint for the data connection from a plurality of available endpoints according to a location of a requesting process and locations of the available endpoints.

16. Software stored on a non-transitory computer-readable medium, for processing data, the software including instructions for causing a system to:

provide a specification of one or more named units stored in a distributed filesystem;

provide a specification for establishing data connections with a computation system;

provide a specification for processes of a first plurality of processes for invocation on a distributed processing system by a manager of the distributed processing system, each process of the first plurality of processes being specified for establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem;

receive requests to form a plurality of data connections between the distributed processing system and the computation system, and provide information for forming at least one data connection being between each process of the first plurality of processes on the distributed processing system and a respective process of a second plurality of processes on the computation system, the first and second plurality of processes being separate from the manager of the distributed processing system; and

receive data concurrently over the plurality of data connections from the first plurality of processes at the computation system.

17. A system for processing data, the system including:

a distributed filesystem;

a distributed processing system;

a computation system; and

a client of the distributed processing system configured to:

provide a specification of one or more named units stored in the distributed filesystem;

provide a specification for establishing data connections with the computation system;

provide a specification for processes of a first plurality of processes for invocation on the distributed processing system by a manager of the distributed processing system, each process of the first plurality of processes being specified for establishing a data connection with a storage element of the distributed filesystem for accessing a corresponding part of the named unit in the distributed filesystem;

receive requests to form a plurality of data connections between the distributed processing system and the computation system, and provide information for forming at least one data connection being between each process of the first plurality of processes on the distributed processing system and a respective process of a second plurality of processes on the computation system, the first and second plurality of processes being separate from the manager of the distributed processing system; and

receive data concurrently over the plurality of data connections from the first plurality of processes at the computation system.

18. A method for processing data, the data being provided from a distributed processing system implementing a map-reduce data processing framework, the method including:

providing to the distributed processing system a specification for a map procedure for invocation on the distributed processing system by a manager of the distributed processing system, the specification for the map procedure identifying one or more named units in a distributed filesystem for processing and including a specification for establishing data connections with a computation system separate from the distributed processing system;

causing execution of a plurality of instances of the map procedure on the distributed processing system;

receiving requests to form a plurality of data flow connections between executing instances of the map procedure and the computation system, and providing information for forming at least one data flow connection being between each executing instance of the map procedure and a respective process executing on the computation system; and

receiving data concurrently over the plurality of data flow connections and processing the received data in the computation system.

19. The method of claim 18 where the distributed processing system includes an Apache Hadoop ® system.

20. The method of claim 18 where the data flow connections include transport layer data network communication connections.

21. Software stored on a non-transitory computer-readable medium, for processing data, the data being provided from a distributed processing system implementing a map-reduce data processing framework, the software including instructions for causing a system to:

provide to the distributed processing system a specification for a map procedure for invocation on the distributed processing system by a manager of the distributed processing system, the specification for the map procedure identifying one or more named units in a distributed filesystem for processing and including a specification for establishing data connections with a computation system separate from the distributed processing system;

cause execution of a plurality of instances of the map procedure on the distributed processing system;

receive requests to form a plurality of data flow connections between executing instances of the map procedure and the computation system, and provide information for forming at least one data flow connection being between each executing instance of the map procedure and a respective process executing on the computation system; and

receive data concurrently over the plurality of data flow connections and process the received data in the computation system.

22. A system for processing data, the system including:

a distributed filesystem;

a distributed processing system;

a computation system separate from the distributed processing system; and

a client of the distributed processing system configured to:

provide to the distributed processing system a specification for a map procedure for invocation on the distributed processing system by a manager of the distributed processing system, the specification for the map procedure identifying one or more named units in the distributed filesystem for processing and including a specification for establishing data connections with the computation system;

cause execution of a plurality of instances of the map procedure on the distributed processing system;

receive requests to form a plurality of data flow connections between executing instances of the map procedure and the computation system, and provide information for forming at least one data flow connection being between each executing instance of the map procedure and a respective process executing on the computation system; and

receive data concurrently over the plurality of data flow connections and processing the received data in the computation system.