

VECTOR PROCESSING FOR SEGMENTATION HASH VALUES CALCULATION

BACKGROUND

The present invention, in some embodiments thereof, relates to segmenting an input data stream and, more specifically, but not exclusively, to segmenting an input data stream using vector processing.

In present times being an information era, the volumes of data which need to be stored and/or transferred between locations rapidly increase. The enormous quantities of data may present major cost and/or complexity challenges with respect to storage space for storing the data and/or network bandwidth for transferring it.

One solution commonly used for reducing the amount of data for storage and/or transfer is data deduplication (often called "intelligent compression" or "single-instance storage") which is a method of reducing the data volume by eliminating redundant data. While there are methods for file deduplication, block deduplication may present better results with respect to data compression. In block deduplication only one unique instance of a data segment (block) of a data stream is actually retained while redundant data segment(s) which are identical to the already retained data segment are replaced with a pointer to a copy of the retained data segment. Block deduplication processes a data stream that may be one of multiple data types, for example, data files, media files, stream data and the like to identify unique instances of one or more data segments (blocks). A unique number (hash value) is generated for each segment using a hash algorithm. A cryptographic strength hash algorithm is usually used for this purpose, for example, MD5 and/or SHA-1. The hash value generated for each segment is compared to existing hash values generated for previous segments and in case the hash value equals to an existing hash value, the segment is not retained but rather replaced with a pointer to the copy of the existing segment. Furthermore, in case the segment is updated, only the changed data may be retained while the remaining unchanged data which may include a significant amount of the segment is not retained.

One of the main challenges is effectively segmenting the data stream such that the segments are affected as little as possible by changes to the segments' data contents. Rolling hash techniques may be used for segmenting the data stream as known in the industry. Using a rolling hash, a hash value is calculated for shifting sequences of data in the data stream (in each rolling sequence an ending data item is omitted and a new data item is inserted). The calculated hash value is checked for compliance with pre-defined one or more segmentation criteria and in case the compliance is identified, the end of the respective rolling sequence is designated as a segment boundary or cut point.

SUMMARY

According to a first aspect of the present invention there is provided a system for segmenting an input data stream using vector processing, comprising a processor adapted to repeat the following steps throughout an input data stream to create a segmented data stream consisting a plurality of segments: apply a rolling sequence over a sequence of consecutive data items of an input data stream, the rolling sequence includes a subset of consecutive data items of the sequence; calculate, concurrently, a plurality of partial hash values, each by one of a plurality of processing pipelines of the processor, each for a respective one of a plurality of partial rolling sequences each including evenly spaced data items of the subset; determine compliance of each of the plurality of partial hash values with at least one respective partial segmentation criterion; and designate the sequence as a variable size segment when at least some of the plurality of partial hash values comply with the respective at least one partial segmentation criterion.

The term concurrent has thereby the meaning of overlapping in duration also including the meaning of simultaneous, e.g. happening at the same time.

According to a first implementation form of the first aspect of the present invention as such the processor is a single-instruction-multiple data, SIMD, processor.

According to a second implementation form of the first aspect as such or according to the first implementation form of the first aspect the processor is adapted to calculate each of the plurality of partial hash values as a partial rolling hash value using a respective partial hash value of a respective previous partial rolling sequence, an omitted data item which is omitted from the respective partial rolling sequence and an added data item which is added to the respective partial rolling sequence.

According to a third implementation form of the first aspect as such or according to any of the first or second implementation form of the first aspect the processor is adapted to designate the sequence as the variable size segment when a number of consecutive complying partial hash values calculated for successive subsets of the sequence which comply with the respective at least one partial segmentation criterion exceeds the number of the plurality of partial rolling sequences.

According to a fourth implementation form of the first aspect as such or according to any of the first to third implementation form of the first aspect the sequence includes a pre-defined minimum number of the consecutive data items.

According to a fifth implementation form of the first aspect as such or according to any of the first to fourth implementation form the processor is adapted to designate at least one large sequence of the plurality of sequences as the variable size segment when a size of the large sequence exceeds a pre-defined maximum value before detecting compliance of the partial hash values with the respective at least one partial segmentation criterion.

According to a second aspect of the present invention there is provided a method for segmenting an input data stream using vector processing, comprising using a processor adapted to repeat the following steps throughout an input data stream to create a segmented data stream consisting a plurality of segments: apply a rolling sequence over a sequence of consecutive data items of an input data stream, the rolling sequence includes a subset of consecutive data items of the sequence; calculate, concurrently, a plurality of partial hash values, each by one of a plurality of processing pipelines of processor, each for a respective one of a plurality of partial rolling sequences each including evenly spaced data items of the subset; determine compliance of each of the plurality of partial hash values with at least one respective

partial segmentation criterion; and designate the sequence as a variable size segment when at least some of the plurality of partial hash values comply with the respective at least one partial segmentation criterion.

According to a first implementation form of the second aspect of the present invention as such each of the plurality of partial hash values is calculated as a partial rolling hash value using a respective partial hash value of a respective previous partial rolling sequence, an omitted data item which is omitted from the respective partial rolling sequence and an added data item which is added to the respective partial rolling sequence.

According to a second implementation form of the second aspect as such or according to the first implementation form of the second aspect the sequence is designated as the variable size segment when a number of consecutive complying partial hash values calculated for successive subsets of the sequence which comply with the respective at least one partial segmentation criterion exceeds the number of the plurality of partial rolling sequences.

According to a third implementation form of the second aspect as such or according to any of the first or second implementation form of the second aspect the sequence includes a pre-defined minimum number of the consecutive data items.

According to a fourth implementation form of the second aspect as such or according to any of the first to third implementation form of the second aspect at least one large sequence of the plurality of sequences is designated as the variable size segment when a size of the large sequence exceeds a pre-defined maximum value before detecting compliance of the partial hash values with the respective at least one partial segmentation criterion.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

Some embodiments of the invention are herein described, by way of example only, with reference to the accompanying drawings. With specific reference now to the drawings in detail, it is stressed that the particulars shown are by way of example and

for purposes of illustrative discussion of embodiments of the invention. In this regard, the description taken with the drawings makes apparent to those skilled in the art how embodiments of the invention may be practiced.

In the drawings:

FIG. 1 is a schematic illustration of an exemplary system for segmenting an input data stream using vector processing, according to some embodiments of the present invention;

FIG. 2 is a flowchart of an exemplary process for segmenting an input data stream using vector processing, according to some embodiments of the present invention;

FIG. 3A is a schematic illustration of an exemplary process of calculating rolling hash values for segmenting an input data stream;

FIG. 3B is a schematic illustration of an exemplary process of calculating rolling hash values for segmenting an input data stream;

FIG. 3C is an exemplary formulation of rolling hash values calculation for segmenting an input data stream;

FIG. 4A is a schematic illustration of an exemplary process of calculating partial rolling hash values for segmenting an input data stream using vector processing, according to some embodiment of the present invention;

FIG. 4B is a schematic illustration of an exemplary process of calculating partial rolling hash values for segmenting an input data stream using vector processing, according to some embodiment of the present invention;

FIG. 4C is a schematic illustration of an exemplary formulation of partial rolling hash values calculation for segmenting, according to some embodiment of the present invention;

FIG. 5A is a schematic illustration of an exemplary process of qualifying partial rolling hash values with partial segmentation criteria, according to some embodiment of the present invention;

FIG. 5B is a schematic illustration of an exemplary process of qualifying partial rolling hash values with partial segmentation criteria, according to some embodiment of the present invention;

FIG. 5C is a schematic illustration of an exemplary process of qualifying partial rolling hash values with partial segmentation criteria, according to some embodiment of the present invention; and

FIG. 5D is a schematic illustration of an exemplary process of qualifying partial rolling hash values with partial segmentation criteria, according to some embodiment of the present invention.

DETAILED DESCRIPTION

The present invention, in some embodiments thereof, relates to segmenting an input data stream and, more specifically, but not exclusively, to segmenting an input data stream using vector processing.

The present invention presents systems and methods for segmenting an input data stream using vector processing as part of a deduplication process applied to the input data stream in order to reduce the amount of data of the input data stream by removing redundant (duplicated) data segments. The deduplication process for reducing the amount of data of the input data stream, for example, data files, media files, streaming data and the like is performed in order to reduce storage space and/or network bandwidth required for storing and/or transferring the input data stream. The segmentation is done by concurrently or simultaneously processing a plurality of partial rolling sequences constituting a rolling sequence which is gradually shifted through a sequence of consecutive data items, for example, bytes, words, double-words and/or pixels of the input data stream. The partial rolling sequences are concurrently processed to calculate a partial rolling hash value for each of the partial

rolling sequences. Each of the plurality of partial rolling hash values is calculated in a respective one of a plurality of processing pipelines of one or more vector processors, for example a single instruction multiple data (SIMD) processor. While the rolling sequence includes a subset of consecutive data items of the sequence, each of the plurality of partial rolling sequences includes evenly spaced data items of the subset of the rolling sequence. The sequence may be designated a variable size segment in case at least some of the partial rolling hash values calculated for each of the partial rolling sequences comply with (satisfy) respective one or more partial segmentation criteria. The process is repeated for a plurality of following rolling sequences throughout the input data stream to create a segmented data stream where each rolling sequence starts at the point where a previous rolling sequence ends.

Simultaneous segmentation using vector processing may present significant advantages compared to existing sequential segmentation methods. Vector processing technology is rapidly advancing in many aspects, for example, number of processing pipelines available in modern processors, number of data items that may be processed in parallel and/or processing power of the processor(s). While efficient segmentation of the input data stream may have a major contribution to the effectivity of the entire deduplication process it may be one of the major time consuming and/or processor intensive operations of the deduplication process. Since the data items of the input sequence may be regarded as independent from each other with respect to calculating the hash value for the rolling sequences, simultaneous processing of the partial rolling sequences as part of the segmentation process may take full advantage of the vector processing technology such as a processor(s) having the SIMD engine. Processing the input data stream using the vector processor(s) may significantly reduce the segmentation time compared to the sequential segmentation employed by current segmentation methods. As the segmentation time is significantly reduced, the entire deduplication process may be significantly shortened and may possibly remove bottlenecks in data transfer and/or data storage operations.

Before explaining at least one embodiment of the invention in detail, it is to be understood that the invention is not necessarily limited in its application to the details of construction and the arrangement of the components and/or methods set forth in the following description and/or illustrated in the drawings and/or the Examples. The

invention is capable of other embodiments or of being practiced or carried out in various ways.

The present invention may be a system, a method, and/or a computer program product. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention.

The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing.

Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network.

The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (e.g., through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present invention.

Aspects of the present invention are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and

computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

Reference is now made to FIG. 1, which is a schematic illustration of an exemplary system for segmenting an input data stream using vector processing, according to some embodiments of the present invention. A system 100 includes an input/output (I/O) interface 102 for receiving and/or designating an input data stream 120 and outputting a segmented data stream 130, a processor(s) 104 comprising a plurality of processing pipelines 106 for segmenting the input data stream 120 to create the segmented data stream 130 and a program store 108. The input data stream 120 may be received in one or more formats, for example, a data file, a media file, a streaming data and the like. The input data stream 120 includes a plurality of data items, for example, bytes, words, double-words and/or pixels which may be arranged in sequence as a stream. The I/O interface 102 may include one or more interfaces, for example, a network interface, a memory interface and/or a storage interface. The I/O interface 102 may be used, for example, for receiving and/or transmitting the data streams 120 and/or 130 over a network, for fetching and/or storing the data streams

120 and/or 130 to a memory device and/or a storage device and the like. The processor(s) 104 may include one or more processors each having one or more cores. The processor(s) 104 further includes vector processing capabilities such as the plurality of processing pipelines 106. The processing pipelines 106 support multiple independent execution pipelines each capable of processing data, for example, multiple processors, multiple cores and/or vector processors, for example, a SIMD processor and/or a SIMD engine. The SIMD engine comprises a plurality of processing pipelines 106 capable of concurrently processing multiple data items at a single instruction of the processor(s) 104. The program store 108 may include one or more non-transitory persistent storage devices, for example, a hard drive, a Flash array and the like.

Reference is also made to FIG. 2, which is a flowchart of an exemplary process for segmenting an input data stream using vector processing, according to some embodiments of the present invention. A segmentation process 200 for segmenting an input data stream may be performed by a system such as the system 100. The segmentation process 200 employs the plurality of processing pipelines 106 to concurrently process partial sequences of the input data stream 120 to produce the segmented data stream 130.

The segmentation process 200 may be done by one or more software modules such as, for example, a coordinator 110 and/or a worker 112 which comprise a plurality of program instructions executed by the processor(s) 104 and/or the processing pipelines 106 from the program store 108. Optionally, the workers 112 may include one or more microcode modules embedded with the processing pipelines 106, where the microcode modules include program instructions executed by the processing pipelines 106. The coordinator 110 may be executed, for example, by a processing unit of the processor(s) 104. The coordinator 110 may manage and/or coordinate the segmentation process, for example, distribute data between the pluralities of processing pipelines 106, collect data from the plurality of processing pipelines 106, synchronize data, synchronize tasks, coordinate the workers 112, designate segments and/or the like. The processor(s) 104 and/or each of the processing pipelines 106 may execute an instance of a plurality of workers 112 to process concurrently the partial sequences of the input data stream 120. In case the processor(s) 104 is a vector processor comprising the processing pipelines 106 that are independent processing

units, each processing pipeline 106 may independently execute a worker 112 instance. However, in case the processor(s) 104 incorporates the SIMD engine, the worker 112 may be executed by the processor(s) 104 that assigns data to each of the processing pipelines 106 of the SIMD engine. The processor(s) 104 may then initiate a single instruction to instruct all the processing pipelines 106 of the SIMD engine to concurrently execute the same operation (instruction), each processing pipeline 106 processing its respective assigned data.

As shown at 202, the process 200 starts with the coordinator 110 receiving the input data stream 120 from the I/O interface 102.

Before explaining the simultaneous segmentation process using the vector processing, the segmentation process using rolling sequences and rolling hash is first described.

Reference is now made to FIG. 3A and FIG. 3B which are schematic illustrations of an exemplary process of calculating rolling hash values for segmenting an input data stream. An input data stream 120 is processed by applying a shifting rolling sequence over the input data stream 120 comprising a plurality of data items, for example, bytes, words, double words, pixels and the like. The rolling sequences are shifted over the input data stream 120 starting at the beginning of the data stream 120 and shifted towards the end of the input data stream 120. The rolling sequence is shifted until a segment is identified and designated. Once a segment is designated, a following sequence starts from the end of the previous designated segment and a respective rolling sequence is shifted towards the end of the input data stream 120. This process is repeated until the entire input data stream is segmented to produce the segmented data stream 130. The number of the data items included in the rolling sequences may vary according to one or more aspects, for example, segmentation requirements, data throughput, processing power and the like. The exemplary input data stream 120 is processed with rolling sequences each comprising 64 data items, however as explained the number of data items may be different. A hash value H_K 312A is calculated for the rolling sequence rolling sequence 310A which includes data items S_K 310A0 through S_{K-63} 310A63. The rolling sequence is then shifted to the left to define the rolling sequence 310B such that the data item S_{K-63} 310A63 is omitted

and a new data item S_{K+1} 310B1 is added. A new hash value H_{K+1} 312B is calculated for the rolling sequence 310B which includes data items S_{K+1} 310B1 through S_{K-62} 310A62. The rolling sequence is shifted through the sequence of the input data stream 120 until a segment is detected as explained hereinafter. The hash values H_K 312A and H_{K+1} 312B as well as following hash values calculated for the following sequences of the input data stream 120 may be calculated using one of many rolling hash functions, for example, a Rabin-Karp rolling hash and/or a Buzhash.

While it is possible to calculate the hash values such as the hash values H_K 312A and H_{K+1} 312B from their respective rolling sequences such as the rolling sequences 310A and 310B, it is evident from the above, that each of the hash values H_K 312A and H_{K+1} 312B depends on a previous hash value, an omitted data item from the respective sequence and an added data item to the respective sequence. Each hash value may therefore be considered as a rolling hash value and may be calculated using the values of the previous hash value, the omitted data item and the added data item thus avoiding redundant complex computation of the entire respective rolling sequence such as the rolling sequences 310A and 310B.

Reference is now made to FIG. 3C which is an exemplary formulation of rolling hash values calculation for segmenting an input data stream. A rolling hash value such as the rolling hash value H_{K+1} 312B for a respective rolling sequence such as the rolling sequence 310B applied to a sequence of an exemplary input stream such as the input stream 120 may be calculated as a function of a previous hash value such as the hash value H_K 312A, an omitted data item such as the omitted data item S_{K-63} 310A63 and an added data item such as the added data item S_{K+1} 310B1.

Each of the hash values such as the hash values H_K 312A and H_{K+1} 312B is compared for compliance against one or more segmentation criteria to identify a point in which the sequence is “cut” to designate a segment. The one or more segmentation criteria may define for example, a data pattern such as, for example, checking that the last 12 bits of the hash value equal a pre-defined value. This will typically produce a “cut” on average once every 4,096 bytes. Naturally, the size of the segment is variable and depends on the location within the sequence in which the calculated rolling hash value complies with the one or more segmentation criteria. A minimum size may be

pre-defined for each of the plurality of variable size segments such that the processed sequence of the input data stream 120 starts from a minimum block size and increases until “cut”. A maximum size may be pre-defined for each of the plurality of variable size segments such that if the pre-defined maximum size is reached before identifying a “cut” point in the processed sequence, the segment is “cut” even if the hash value does not comply with the one or more segmentation criteria.

The segmentation process continues over following sequences of the input data stream 120 to the end of the input data stream 120 to create a segmented data stream such as the segmented data stream 130.

Reference is made once again to FIG. 2. As shown at 204, the coordinator 110 applies a rolling sequence such as the rolling sequences 310A and/or 310B over a sequence of consecutive data items of an input stream such as the input stream 120. The coordinator 110 selects the first sequence to start from the beginning of the input data stream 120 and advances towards the end of the input data stream 120. The coordinator 110 shifts the rolling sequences such as the rolling sequences 310A and 310B over the processed sequence. As discussed above, the number of data items included in each of the rolling sequences, for example, 32, 64 and/or 128 depends on one or more aspects of the segmentation process and/or of processing resources characteristics and/or capabilities.

As shown at 206, the coordinator 110 splits the rolling sequence to a plurality of partial rolling sequences which are concurrently (concurrently) processed, each by a respective one of the plurality workers 112 each executed by a respective one of the processing pipelines 106. In case the process 200 is performed by the processor(s) 104 having the SIMD engine, the worker 112 is executed by the processor(s) 104 that assigns data of the respective partial rolling sequence to each of the processing pipelines 106 of the SIMD engine. The number of the partial rolling sequences is set to fit the number of available processing pipelines 106, for example, 4, 8, 16, 32, 64 and/or 128. Each of the partial rolling sequences includes evenly spaced data items of the subset of the rolling sequence such as the rolling sequences 310A and 310B. For example, the rolling sequences 310A and 310B are split to 4 partial rolling sequences to fit 4 workers 112 executed by a vector processor(s) 104 having 4 processing

pipelines such as the processing pipelines 106 or a vector processor(s) 104 with a SIMD engine having 4 processing pipelines 106. Each partial rolling sequence of the rolling sequences 310A and 310B includes every 4th data item of the rolling sequence 310A or 310B. A first partial rolling sequence may comprise data items 0, 4, 8..., a second data sub-stream may comprise data items 1, 5, 9..., a third data sub-stream may comprise data items 2, 6, 10... and a fourth data sub-stream may comprise data items 3, 7, 11... Similarly, as another example, in case the rolling sequences 310A and 310B are split to 8 partial rolling sequences to fit 8 workers 112 executed by a vector processor(s) 104 having 8 processing pipelines such as the processing pipelines 106 and 8 SIMD processing pipelines 106, each partial rolling sequence includes every 8th data item of the subset of the rolling sequence. The first partial rolling sequence may comprise data items 0, 8, 16..., the second partial rolling sequence may comprise data items 1, 9, 17 ..., the third partial rolling sequence may comprise data items 2, 10, 18... the fourth partial rolling sequence may comprise data items 3, 11, 19... a fifth partial rolling sequence may comprise data items 4, 12, 20... a sixth partial rolling sequence may comprise data items 5, 13, 21... a seventh partial rolling sequence may comprise data items 6, 14, 22... and an eighth partial rolling sequence may comprise data items 7, 15, 23 and so on. Similarly, the coordinator 110 may split the sequence to 16, 32, 64, 128, 256 and the like to fit the number of available workers 112, i.e. the number of available processing pipelines 106.

Each of the workers 112 executed by a respective one of the plurality of processing pipelines 106 processes a respective partial rolling sequence to calculate a partial rolling hash value for respective partial rolling sequences. The workers 112 may calculate the partial hash values using one of many rolling hash functions, for example, a Rabin-Karp rolling hash and/or a Buzhash. The plurality of workers 112 process their respective partial rolling sequence concurrently such that all partial rolling sequences are processed concurrently. For the processor(s) 104 having the SIMD engine, the worker 112 executed by the processor(s) 104 initiates a single command (instruction) to instruct all the SIMD engine processing pipelines 106 to concurrently calculate the partial rolling hash values of their respective partial rolling sequences.

Reference is now made to FIG. 4A and FIG. 4B which are schematic illustrations of an exemplary process of calculating partial rolling hash values for segmenting an input data stream using vector processing, according to some embodiment of the present invention.

As seen in FIG. 4A, the coordinator 110 splits a rolling sequence such as the rolling sequence 310A to 8 partial rolling sequences 410A0 through 410A7 to fit an exemplary vector processor(s) 104 having 8 processing pipeline 106. The coordinator 110 may distribute the partial rolling sequences 410A0 through 410A7 to 8 respective workers 112. For the processor(s) 104 having the SIMD engine, a single worker 112 executed by the processor(s) 104 assigns each of the SIMD engine processing pipelines 106 with data of the respective one of the partial rolling sequences 410A0-410A7. However, as presented herein above the number of partial rolling sequences may be different according to the number of workers 112 and/or available processing pipelines 106. Each of the partial rolling sequences 410A0 through 410A7 is processed by a respective worker 112 executed by a respective one of the plurality of processing pipelines 106. For the processor(s) 104 having the SIMD engine, the SIMD engine processing pipelines 106 concurrently process the respective partial rolling sequences 410A0-410A7 at a single instruction by the worker 112 (executed by the processor(s) 104).

The exemplary process 200 presented herein follows the example presented herein before where the each of the rolling sequences such as the rolling sequence 310A comprises a subset of 64 data items hence the coordinator 110 splits the rolling sequence 310A to 8 partial rolling sequences 410A0 through 410A7 each including 8 data items. Therefore, the exemplary partial rolling sequences 410A0 includes 8 data items S_K 310A0, S_{K-8} 310A8, S_{K-16} 310A16, S_{K-24} 310A24, S_{K-32} 310A32, S_{K-40} 310A40, S_{K-48} 310A48 and S_{K-56} 310A56, the exemplary partial rolling sequences 410A7 includes 8 data items S_{K-7} 310A7, S_{K-15} 310A15, S_{K-23} 310A23, S_{K-31} 310A31, S_{K-39} 310A39, S_{K-47} 310A47, S_{K-55} 310A55 and S_{K-63} 310A63 and so on. It is emphasized once again that each of the plurality of partial rolling sequences such as the partial rolling sequences 410A0 through 410A7 each includes 8 data items, however the number of data items may vary according to one or more aspects, for example, a segmentation requirement, a hash function(s) type, a data throughput, a

processor(s) architecture and the like. The processor(s) architecture may include one or more characteristics, for example, a register's width, a cache line width, a memory interface width and/or speed, a network interface width and/or speed, a storage media interface width and/or speed and the like.

The eight workers 112 concurrently process the eight partial rolling sequences 410A0 through 410A7 to calculate a partial rolling hash value such as partial rolling hash values H'_K 412A0 through H'_{K-7} 412A7 for each of the respective partial rolling sequences 410A0 through 410A7. Additionally and/or alternatively, in case the processor(s) 104 include the SIMD engine, the worker 112 executed by the processor(s) 104 instructs the eight SIMD engine processing pipelines 106 process the eight partial rolling sequences 410A0 through 410A7. The eight SIMD engine processing pipelines 106 calculate the partial rolling hash values H'_K 412A0 through H'_{K-7} 412A7 for each of the respective one of the partial rolling sequences 410A0-410A7. The worker(s) 112 may use one or more hash functions, for example, Rabin-Karp or Buzhash to calculate the partial rolling hash values H'_K 412A0 through H'_{K-7} 412A7.

As seen in FIG. 4B, each of the processing pipelines 106 shifts the partial rolling sequences to the right to create partial rolling sequences 410B0 through 410B7 such that the left most data item of each of the previous rolling sequences 410A0-410A7 is omitted and a new data item is added. For example, the partial rolling sequence 410B0 corresponds to a shifted partial rolling sequence 410A0 in which the data item S_{K-56} 310A56 is omitted and a new data item S_{K+8} 310B8 is added. Similarly, the partial rolling sequence 410B7 corresponds to a shifted partial rolling sequence 410A7 in which the data item S_{K-63} 310A63 is omitted and a new data item S_{K+1} 310B1 is added.

The eight processing pipelines 106 concurrently process their respective partial rolling sequences 410B0-410B7 to calculate a partial hash value such as the hash values H'_{K+8} 412B0 through H'_{K+1} 412B7 for each of the respective partial rolling sequences 410B0-410B7.

Similarly to hash values such as the hash values H_K 312A and H_{K+1} 312B, each of the partial hash values such as the partial hash values H'_K 412A0 – H'_{K-7} 412A7 and

H'_{K+8} 412B0 – H'_{K+1} 412B7 may be calculated for their respective partial sequences 410A0 – 410A7 and 410B0 – 410B7. However as evident from the above, each of the partial hash values H'_K 412A0 – H'_{K-7} 412A7 and H'_{K+8} 412B0 – H'_{K+1} 412B7 depends on a previous respective partial hash value, an omitted data item from the respective partial rolling sequence and an added data item to the respective partial rolling sequence. Each partial hash value H'_K 412A0 – H'_{K-7} 412A7 and H'_{K+8} 412B0 – H'_{K+1} 412B7 may therefore be considered as a rolling hash. The worker(s) 112 may therefore calculate the partial hash value H'_K 412A0 – H'_{K-7} 412A7 using the values of the previous respective partial hash value, the respective omitted data item and the respective added data item thus avoiding redundant computation of the entire respective partial rolling sequence 410A0 – 410A7 and/or 410B0 – 410B7.

Reference is now made to FIG. 4C which is an exemplary formulation of partial rolling hash values calculation for segmenting an input data stream, according to some embodiment of the present invention. A respective processing pipeline 106 may calculate a partial rolling hash value such as the partial rolling hash value H'_{K+1} 412B7 for a respective partial rolling sequence such as the partial rolling sequence 410B7, as a function of a previous partial rolling hash value such as the partial rolling hash value H'_{K-7} 412A7, an omitted data item such as the omitted data item S_{K-63} 310A63 and an added data item such as the added data item S_{K+1} 310B1. Similarly, each of the other processing pipelines 106 may calculate the partial rolling hash values H'_{K+2} 412B6 through H'_{K+8} 412B0 using the same methodology.

Reference is made once again to FIG. 2. As shown at 208, each of the calculated partial rolling hash values is checked for compliance with and/or satisfies respective one or more partial segmentation criteria to determine if the complete sequence comprising the rolling sequences such as the rolling sequences 310A and 310B (each comprising the respective partial rolling sequences) may be designated as a segment. Each of the respective partial hash values such as the partial rolling hash values H'_K 412A0 – H'_{K-7} 412A7 and/or H'_{K+8} 412B0 – H'_{K+1} 412B7 are checked for compliance to identify a point in which the sequence may be “cut” to designate a segment. The one or more partial segmentation criteria may define, for example, a data pattern such as the partial rolling hash value 412 is larger than a predefined value and/or a predefined number of bits in the partial rolling hash value 412 equal a

predefined value. The number of consecutive partial hash values that each complies with the respective partial segmentation criteria affects the average segment size. The number of consecutive partial hash values that are required to comply with their respective partial segmentation criteria may vary according to a maximum allowed and/or desired data segment size.

As shown at 210, in case the partial rolling hash values comply with the respective partial segmentation criterion(s), the coordinator 110 may “cut” the sequence on which the rolling sequence is shifted (and which comprises the partial rolling sequences) and designates the sequence as a data segment. The coordinator 110 may use one or more segmentation criteria for “cutting” the sequence and designating it as a segment where the segmentation criterion is a combination of the one or more partial segmentation criteria.

Naturally, the size of the segment is variable and depends on the location within the sequence in which compliance is identified for the respective calculated partial rolling hash values with the respective one or more partial segmentation criteria. Typically, for data deduplication, the average size of the data segments may be, for example, 4 kilo bytes (KB), 6KB and/or 8KB which may best fit current processing, storage and/or networking throughput and performance such that the overhead of processing the data segments is optimized with respect to the number of segments.

Optionally, a minimum size may be pre-defined for each of the plurality of variable size segments such that the partial rolling sequences starts from a minimum block size and increases until “cut”. Based on the typical segment size, the pre-defined minimum size may be for example, 2KB.

Optionally, a maximum size may be pre-defined for each of the plurality of variable size segments such that if the pre-defined maximum size is reached before identifying a “cut” point, the segment is “cut” even if the hash value does not match the one or more segmentation criteria. Based on the typical segment size, the pre-defined maximum size may be for example, 16KB.

Reference is now made to FIG. 5A and 5B which are schematic illustrations of an exemplary process of qualifying partial rolling hash values with partial segmentation criteria, according to some embodiment of the present invention. The coordinator 110 may identify compliance with a segmentation criterion when the workers 112 identify compliance of their respective partial rolling hash value associated with one of a plurality of partial rolling sequences with the respective partial segmentation criteria. In case the processor(s) 104 has the SIMD engine, the worker 112 executed by the processor(s) collects the partial rolling hash values each calculated by a respective one of the SIMD processing pipelines 106 and compares the partial rolling hash values against the respective partial segmentation criteria. For example, in case partial rolling hash values such as the partial rolling hash values H'_K 412A0 – H'_{K-7} 412A7 of respective partial rolling sequences such as the partial rolling sequences 410A0 – 410A7 all comply with respective partial segmentation criteria, the coordinator 110 may designate the sequence to the point of data item S_K 310A0 as a segment.

As shown in FIG. 5A, Q_K 510A is a segmentation criteria which is a combination of a set of partial segmentation criteria Q'_{K-7} 510A7 – Q'_K 510A0 where a “1” indicates compliance of the respective partial hash value with the respective partial segmentation criteria and “0” indicates non-compliance of the respective partial hash value with the respective one or more partial segmentation criteria. For example, the partial segmentation criterion Q'_{K-7} 510A7 indicates compliance of the partial hash value H'_{K-7} 412A7. Since the partial segmentation criteria Q'_{K-5} 510A5, Q'_{K-3} 510A3 and Q'_{K-1} 510A1 do not comply with the respective partial segmentation criteria, the entire segmentation criterion Q_K 510A is not met.

On the other hand, as shown in FIG. 5B all the partial segmentation criteria Q'_{K-7} 510A7 – Q'_K 510A0 comply with their respective partial segmentation criteria and therefore the entire segmentation criterion Q_K 510A is met and a “cut” 520 is made to designate the sequence which includes the rolling sequences comprising the respective partial rolling sequences as a segment.

Optionally, the coordinator 110 checks for compliance of consecutive partial rolling hash values of successive rolling sequences, for example, a previous rolling

sequence and/or a subsequent rolling sequence with the respective partial segmentation criteria. The number of consecutive partial hash values that need to comply with the respective partial segmentation criteria may exceed the number of partial rolling hash values (which is naturally the number of partial rolling sequences and the number of the processing pipelines 106). The compliance of the rolling sequence with the segmentation criteria is therefore a function of compliance of the partial hash values of the successive partial rolling sequences. As discussed above, the number of consecutive partial rolling hash values that are checked for compliance with the partial segmentation criteria may be set to fit the required and/or desired typical data segment size since the number of the consecutive complying partial rolling hash values affects the average segment size. The example, presented herein presents compliance of 8 consecutive partial rolling hash values with the respective partial segmentation criteria, however other numbers of consecutive partial rolling hash values may be used according to the required/desired typical size of the data segments.

By checking for compliance of consecutive partial hash values of two successive groups of partial rolling sequences the coordinator may present an additional advantage as the resolution and/or granularity for segmenting the sequence is increased compared to identifying compliance of an original rolling sequence such as the rolling sequences 310A and/or 310B.

Reference is now made to FIG. 5C and 5D, which are schematic illustrations of an exemplary process of qualifying partial rolling hash values with partial segmentation criteria, according to some embodiment of the present invention.

As shown in FIG. 5C, Q_K 510A is a segmentation criteria which is a combination of a set of partial segmentation criteria Q'_{K-7} 510A7 – Q'_K 510A0. Since the partial segmentation criteria Q'_{K-5} 510A5, Q'_{K-3} 510A3 and Q'_{K-1} 510A1 do not comply with the respective partial segmentation criteria, the coordinator 110 may decide the entire segmentation criterion Q_K 510A is not satisfied. The same applies to the segmentation criterion Q_{K+8} 510B which is a combination of a set of partial segmentation criteria Q'_{K+1} 510B7 – Q'_{K+8} 510B0 and is not met since not all the partial segmentation criteria Q'_{K+1} 510B7 – Q'_{K+8} 510B0 comply with the respective partial segmentation criteria. Furthermore, there is no sequence of partial segmentation

criteria in the two following segmentation criterion Q_K 510A and Q_{K+8} 510B which all comply with their respective partial segmentation criteria so the corresponding sequence may not be designated a segment boundary or “cut” point.

On the other hand, as shown in FIG. 5D the coordinator 110 may identify that a set of 8 consecutive partial segmentation criteria Q'_{K+6} 510B2 through Q'_{K+13} 510C3 all comply with their respective partial segmentation criteria and therefore the coordinator may insert “cut” 520 to designate the corresponding sequence as a segment, where the sequence includes the data items up to a rolling sequence 510C3 (included).

The coordinator 110 may then proceed to identify a following segment in a sequence of the input data stream 120 where the following sequence start where the previously detected segment ends.

Reference is made once again to FIG. 2. As shown at 212, which is a decision point, in case the coordinator 110 detects the end of the input data stream 120 is reached, the process 200 is branched to 214. In case the coordinator 110 detects that the input data stream 120 comprises additional data items, the coordinator 110 may branch to 204 to repeat the segmentation process for the following sequence of the input data stream 120. The coordinator 110 repeats steps 204 through 212 until reaching the end of the input data stream 120 to create the segmented data stream 130.

As shown at 214, once the segmentation process 200 for the input data stream 120 ends, the coordinator 110 may output the segmented data stream 130 using, for example, the I/O interface 102.

Following the segmentation, the coordinator 110 may output the segmented data stream to be used for one or more of a plurality of applications, for example, data deduplication.

The descriptions of the various embodiments of the present invention have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit

of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

It is expected that during the life of a patent maturing from this application many relevant vector processing technologies such as SIMD will be developed and the scope of the term SIMD is intended to include all such new technologies a priori.

As used herein the term “about” refers to $\pm 10\%$.

The terms “comprises”, “comprising”, “includes”, “including”, “having” and their conjugates mean “including but not limited to”. This term encompasses the terms “consisting of” and “consisting essentially of”.

The phrase “consisting essentially of” means that the composition or method may include additional ingredients and/or steps, but only if the additional ingredients and/or steps do not materially alter the basic and novel characteristics of the claimed composition or method.

As used herein, the singular form “a”, “an” and “the” include plural references unless the context clearly dictates otherwise. For example, the term “a compound” or “at least one compound” may include a plurality of compounds, including mixtures thereof.

The word “exemplary” is used herein to mean “serving as an example, instance or illustration”. Any embodiment described as “exemplary” is not necessarily to be construed as preferred or advantageous over other embodiments and/or to exclude the incorporation of features from other embodiments.

The word “optionally” is used herein to mean “is provided in some embodiments and not provided in other embodiments”. Any particular embodiment of the invention may include a plurality of “optional” features unless such features conflict.

It is appreciated that certain features of the invention, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable sub combination or as suitable in any other described embodiment of the invention. Certain features described in the context of various embodiments are not to be considered essential features of those embodiments, unless the embodiment is inoperative without those elements.

CLAIMS

1. A system for segmenting an input data stream using vector processing, comprising:

a processor adapted to repeat the following steps throughout an input data stream to create a segmented data stream consisting a plurality of segments:

apply a rolling sequence over a sequence of consecutive data items of an input data stream, the rolling sequence includes a subset of consecutive data items of the sequence;

calculate, concurrently, a plurality of partial hash values, each by one of a plurality of processing pipelines of the processor, each for a respective one of a plurality of partial rolling sequences each including evenly spaced data items of the subset;

determine compliance of each of the plurality of partial hash values with at least one respective partial segmentation criterion; and

designate the sequence as a variable size segment when at least some of the plurality of partial hash values comply with the respective at least one partial segmentation criterion.
2. The system according to claim 1, wherein the processor is a single-instruction-multiple data, SIMD, processor.
3. The system according to claim 1 or 2, wherein the processor is adapted to calculate each of the plurality of partial hash values as a partial rolling hash value using a respective partial hash value of a respective previous partial rolling sequence, an omitted data item which is omitted from the respective partial rolling sequence and an added data item which is added to the respective partial rolling sequence.
4. The system according to any of claims 1 to 3, wherein the processor is adapted to designate the sequence as the variable size segment when a number of consecutive complying partial hash values calculated for successive subsets of

the sequence which comply with the respective at least one partial segmentation criterion exceeds the number of the plurality of partial rolling sequences.

5. The system according to any of claims 1 to 4, wherein the sequence includes a pre-defined minimum number of the consecutive data items.
6. The system according to any of claims 1 to 5, wherein the processor is adapted to designate at least one large sequence of the plurality of sequences as the variable size segment when a size of the large sequence exceeds a pre-defined maximum value before detecting compliance of the partial hash values with the respective at least one partial segmentation criterion.
7. A method for segmenting an input data stream using vector processing, comprising:

using a processor adapted to repeat the following steps throughout an input data stream to create a segmented data stream consisting a plurality of segments:

apply a rolling sequence over a sequence of consecutive data items of an input data stream, the rolling sequence includes a subset of consecutive data items of the sequence;

calculate, concurrently, a plurality of partial hash values, each by one of a plurality of processing pipelines of processor, each for a respective one of a plurality of partial rolling sequences each including evenly spaced data items of the subset;

determine compliance of each of the plurality of partial hash values with at least one respective partial segmentation criterion; and

designate the sequence as a variable size segment when at least some of the plurality of partial hash values comply with the respective at least one partial segmentation criterion.

8. The method according to claim 7, wherein each of the plurality of partial hash values is calculated as a partial rolling hash value using a respective partial hash value of a respective previous partial rolling sequence, an omitted data

item which is omitted from the respective partial rolling sequence and an added data item which is added to the respective partial rolling sequence.

9. The method according to claim 7 or 8, wherein the sequence is designated as the variable size segment when a number of consecutive complying partial hash values calculated for successive subsets of the sequence which comply with the respective at least one partial segmentation criterion exceeds the number of the plurality of partial rolling sequences.
10. The method according to any of claims 7 to 9, wherein the sequence includes a pre-defined minimum number of the consecutive data items.
11. The method according to any of claims 7 to 10, wherein at least one large sequence of the plurality of sequences is designated as the variable size segment when a size of the large sequence exceeds a pre-defined maximum value before detecting compliance of the partial hash values with the respective at least one partial segmentation criterion.