



- (51) **International Patent Classification:**
H04L 12/861 (2013.01) *H04L 12/58* (2006.01)
- (21) **International Application Number:**
PCT/US2015/024325
- (22) **International Filing Date:**
3 April 2015 (03.04.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).
- (72) **Inventor:** ESTEP, Patrick; 5400 Legacy Dr., Plano, Texas 75024 (US).
- (74) **Agents:** PATEL, Neel K. et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,

HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

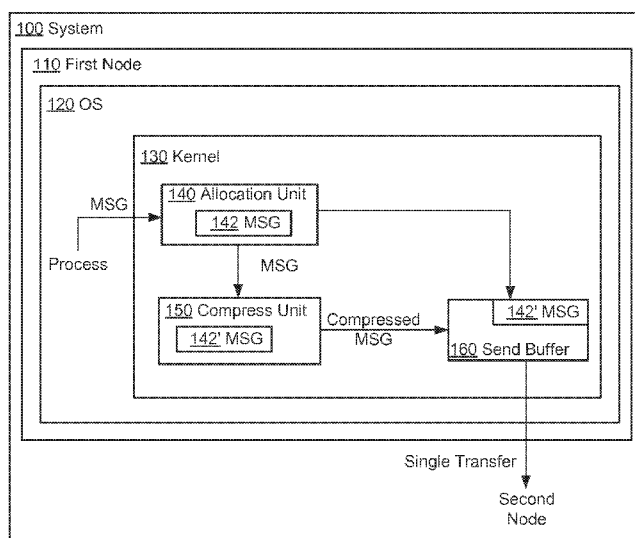
- *as to the identity of the inventor (Rule 4.17(i))*
- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

- *with international search report (Art. 21(3))*

(54) **Title:** COMPRESS AND LOAD MESSAGE INTO SEND BUFFER

FIG. 1



(57) **Abstract:** A message is received from a process. A portion of a send buffer is allocated at a kernel of an operating system (OS) of a first node for the message. The message is compressed and loaded into the send buffer. The message compressed during a window of time that the kernel of the first node is waiting to output the send buffer as a single transfer across a network to a second node.

COMPRESS AND LOAD MESSAGE INTO SEND BUFFER

BACKGROUND

[0001] Networks may have various types of communication topologies. A common communications topology is one with many processes/threads per node, where every process/thread may communicate with every other process/thread in a cluster of nodes. Manufacturers, vendors, and/or service providers are challenged to provide improved communication topologies for more efficient transfer of information between nodes.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The following detailed description references the drawings, wherein:

[0003] FIG. 1 is an example block diagram of a system to compress and load a message into a send buffer.

[0004] FIG. 2 is another example block diagram of a system to compress and load a message into a send buffer;

[0005] FIG. 3 is an example block diagram of a computing device including instructions for compressing and loading a message into a send buffer; and

[0006] FIG. 4 is an example flowchart of a method for compressing and loading a message into a send buffer.

DETAILED DESCRIPTION

[0007] Specific details are given in the following description to provide a thorough understanding of embodiments. However, it will be understood that

embodiments may be practiced without these specific details. For example, systems may be shown in block diagrams in order not to obscure embodiments in unnecessary detail. In other instances, well-known processes, structures and techniques may be shown without unnecessary detail in order to avoid obscuring embodiments.

[0008] Common communications topologies may have many processes/threads per node that can communicate with other processes/threads in a cluster of nodes. These topologies may suffer in performance and scalability. For instance, as a number of nodes and a number of processes/threads increases, the number of connections per node may grow exponentially, e.g. n^2 connections, where n is the number of processes/threads. As the underlying interconnect may have to multiplex/demultiplex each connection onto a single interface, this contention may cause a performance bottleneck.

[0009] Also, there is typically a limit on the number of connections that may be possible per node. As the number of nodes and the number of processes/threads increase, this limit may be reached. Many of the messages being sent in these topologies may be small, which results in poor network throughput. Current systems, which are implemented in user level code, may not fully solve the above problems. Thus, current approaches may have shortcomings from both a performance and scalability perspective.

[0010] Some software systems may aggregate messages to improve performance (e.g. reduce latency) and scalability. These systems may aggregate many smaller messages into a single larger message which is then

sent over the network. However, the smaller messages may remain in the aggregated buffer until the buffer is full or some other threshold is reached (e.g. maximum time to wait). Some systems may do aggregation, but without compression. Conversely, other systems may do compression, but without aggregation.

[0011] During this time period while the aggregation buffer (e.g. send buffer) is held, examples may apply message compression of individual buffers that comprise the aggregation buffer without impacting the total time an aggregation buffer is held before sending. Thus, examples may effectively allow compression to be performed without impacting a latency of the aggregation buffer. An example system may include an allocation unit and a compress unit. The allocation unit may receive a message from a process. The allocation unit may allocate a portion of a send buffer at a kernel of an operating system (OS) of a first node for the message. The compress unit may compress the message and load the compressed message into the send buffer. The message may be compressed during a window of time that the kernel of the first node is waiting to output the send buffer as a single transfer across a network to a second node.

[0012] Thus, examples may reduce the total number of bytes sent over the network to improve throughput. For instance, examples may perform a compression during the inherent latency of message aggregation, which allows the compression to be done without increasing the latency of message transmission. Further, by compressing the messages in the aggregation buffer,

fewer bytes may be sent across the network, thus improving overall capacity with resultant throughput increases.

[0013] Referring now to the drawings, FIG. 1 is an example block diagram of a system 100 to compress and load a message into a send buffer. The system 100 may be any type of message aggregation system, such as a communication network. Example types of communication networks may include wide area networks (WAN), metropolitan area networks (MAN), local area networks (LAN), Internet area networks (IAN), campus area networks (CAN) and virtual private networks (VPN).

[0014] The system 100 is shown to include a first node 110. The term node may refer to a connection point, a redistribution point or a communication endpoint. The node may be an active electronic device that is attached to a network, and is capable of sending, receiving, or forwarding information over a communications channel. Examples of the node may include data communication equipment (DCE) such as a modem, hub, bridge or switch; or data terminal equipment (DTE) such as a digital telephone handset, a printer or a host computer, like a router, a workstation or a server.

[0015] The first node 110 is shown include an operating system (OS) 120. An OS may be a collection of software that manages computer hardware resources and provides common services for applications. Examples types of OSs may include Android, BSD, iOS, Linux, OS X, QNX, Microsoft Windows, Windows Phone, and IBM z/OS.

[0016] The OS 120 is shown to include a kernel 130. The kernel 130 may be a central part of the OS 120 that loads first, and remains in main memory (not

shown). Typically, the kernel 130 may be responsible for memory management, process and task management, and disk management. For example, the kernel 130 may manages input/output requests from an application and translate it into data processing instructions for a central processing unit (not shown) and other electronic components of a node. The kernel 130 may also allocate requests from applications to perform I/O to an appropriate device or part of a device.

[0017] The kernel 130 is shown to include an allocation unit 140, a compress unit 150 and a send buffer 160. The term buffer may refer to a region of physical memory storage used to temporarily store data while it is being moved from one place to another. The kernel 130 may be included in any electronic, magnetic, optical, or other physical storage device that contains or stores information, such as Random Access Memory (RAM), flash memory, a solid state drive (SSD), a hard disk drive (HDD) and the like.

[0018] The allocation and compress units 140 and 150 may include, for example, a hardware device including electronic circuitry for implementing the functionality described below, such as control logic and/or memory. In addition or as an alternative, the allocation and compress units 140 and 150 may be implemented as a series of instructions encoded on a machine-readable storage medium and executable by a processor.

[0019] The send buffer 140 is allocated within the kernel 130 of the OS 120 of the first node 110. Thus, the send buffer 140 may not be paged out of the kernel 130. The allocation unit 140 may receive a message 142 from a process. The term process may refer to an instance of a computer program that is being executed. The process may contain code of the program and/or its current

activity. Depending on a type of the OS 120, the process may be made up of multiple threads of execution that execute instructions concurrently.

[0020] The allocation unit 140 may allocate a portion of the send buffer 160 at the kernel 130 of the OS 120 of the first node 110 for the message 142. The compress unit 150 may compress the message 142' and load the compressed message 142' into the send buffer 160. The message 142 may be compressed during a window of time that the kernel 130 of the first node 110 is waiting to output the send buffer 160 as a single transfer across a network to a second node. The allocation and compress units 140 and 150 are explained in greater detail below with respect to FIG. 2.

[0021] FIG. 2 is another example block diagram of a system 200 to compress and load a message into a send buffer. The system 200 may be any type of message aggregation system, such as a communication network, such as WAN, LAN or VPN. The system 200 is shown to include the first node 210 and a second node 280. The second node 210 may include the functionality and/or hardware of the first node 210 of FIG. 2 and/or vice versa. While the system 200 is shown to only include two nodes 210 and 2280, examples may include more than two nodes, such as a cluster of hundreds or thousands of nodes.

[0022] The first node 210 of FIG. 2 may include the functionality and/or hardware of the first node 110 of FIG. 1. For example, the first node 210 of FIG. 2 includes an OS 220, where the OS 220 includes a kernel 230. The kernel 230 includes an allocation unit 240, a compress unit 250, a send buffer 260 and a plurality of processes 270. The allocation unit 240, compress unit 250 and send

buffer 260 of FIG. 2 may include at least the functionality and/or hardware of the respective allocation unit 140, compress unit 150 and send buffer 160 of FIG. 1.

[0023] The send buffer 260 may be persistently allocated in the kernel 230 for remote direct memory access (RDMA) transfers. By using a persistent RDMA approach, an preparing/un-preparing of the send buffer 260 per message transfer may be avoided.

[0024] As noted above, the allocation unit 240 may receive a message 142 from a process 272. The allocation unit 240 may allocate a portion of a send buffer 260 at a kernel 230 of an operating system (OS) 220 of a first node 210 for the message 142. Further, a plurality of the processes 272-1 to 272-m, where m is a natural number, may transmit a plurality of the messages 142-1 to 142-n, where n is a natural number, to the allocation unit 240. The allocation unit 240 may sequentially allocate the plurality of messages 142-1 to 142-n to the send buffer 260. For example, assume the allocation unit 240 first receives messages 142 from the first process 272-1 and then from the second process 272-2. Here, the allocation unit 240 may first allocate a portion of the send buffer 260 for the messages of the first process 272-1 and then allocation a portion of the send buffer 260 for the messages of the second process 272-2.

[0025] The compress unit 250 may compress the message 142 and load the compressed message 142' into the send buffer 260. The message 142 may be compressed during a window of time that the kernel 230 of the first node 210 is waiting to output the send buffer 260 as a single transfer across a network to a second node 280. Further, the compress unit 250 may concurrently compress and/or load the plurality of messages 142-1 to 142-n to the send buffer 260 this

window of time. For instance, the compress unit 250 may be simultaneously compressing messages 142 of both the first and second process 272-1 and 272-2 and/or simultaneously loading compressed messages 142' of both the first and second process 272-1 and 272-2 to the send buffer 260.

[0026] The allocation unit 240 may update the process 272 with a completion indication after the corresponding message 142 of the process 272 is compressed and/or loaded to the send buffer 260 by the compress unit 250. At least one of the plurality of processes 272-1 to 272-m may include a process buffer 274 to store a message 142 to be sent to the allocation unit 240. The at least one process 272 may reuse the process buffer 274 after receiving the completion indication from the allocation unit 240.

[0027] The compress unit 250 may separately determine a type of compression scheme 252 for each of the plurality of processes 272-1 to 272-m based on a threshold 254. Example compression schemes 252 may include lossy data compression algorithms, such as JPEG, MPEG and the like, as well as lossless data compression algorithms, such as Lempel–Ziv (LZ), LZW (Lempel–Ziv–Welch), LZR (Lempel–Ziv–Renau) and the like.

[0028] In one example, the threshold 254 may include a minimum threshold 255 that relates to a minimum size of a message 142 to be compressed. The compress unit 250 may not compress the message 142 if the size of the message 142 is less than the minimum threshold 255. For instance, a message that is very small may not benefit from compression.

[0029] In another example, the threshold 254 may include a size threshold 256 that relates to the size of the compressed message 142' to load to the send

buffer 260. The allocation unit 240 may vary the size threshold 256 for different types of the processes 262. For example, size threshold 256 may indicate a fixed output size for compressed messages 142' for the first process 271-1 to be 128 kilobytes (KB) and for the second process 271-2 to be 256 KB.

[0030] For instance, the compress unit 250 may compress each of the plurality of messages 142-1 to 142-n of a process 262, such as the first process 262-1, to a same fixed size. Also, the compress unit 250 may compress a remainder of a message 142, such as the first message 142-1 of the first process 262-1, into a separately compressed message 142-2', if an entirety of the first message 142-1 cannot be compressed to the same fixed size.

[0031] In yet another example, the threshold 254 may include a compression threshold 257 to relate to a type of compression scheme 252. The compress unit 250 may base the compression threshold 257 on, for example, a processor load and/or free memory of the system 200. For instance, some compression schemes 252 may provide lower compression with lower CPU cost, while some compression schemes 252 may provide higher compression with higher CPU cost. If less processing resources are available, a less process-intensive compression scheme 252 may be selected.

[0032] In still another example, the threshold 254 may include a history threshold 258 to relate to a compression ratio of a previously compressed message 142'. The compress unit 250 may select a different compression scheme 252 and/or no compression scheme for a current message 142 based on the compression ratio of the previously compressed message 142'. For instance, if a previous message 142 of the first process 272-1 had little or no

compression after the one of the compression schemes 252 was applied, a different compression scheme 252 or no compression scheme 252 may be applied for a next message 142 of the first process 272-1.

[0033] Examples may include more or less than the above four thresholds 255-258. The compression unit 250 may also use any combination of the thresholds 254 to determine a compression scheme 252. Further, the thresholds 254 may be dynamically varied, such as run-time analysis of the compression ratio, time to compress, and the like. Thus, examples may compress messages 142 during a window of time that the send buffer 260 is being held until sending thresholds are met. This may allow for compression to be accomplished without increasing the latency of messaging while also improving overall capacity by sending less data (e.g. compressed data).

[0034] FIG. 3 is an example block diagram of a computing device 300 including instructions for compressing and loading a message into a send buffer. In the embodiment of FIG. 3, the computing device 300 includes a processor 310 and a machine-readable storage medium 320. The machine-readable storage medium 320 further includes instructions 322 and 324 for compressing and loading a message into a send buffer.

[0035] The computing device 300 may be included in or part of, for example, a microprocessor, a controller, a memory module or device, a notebook computer, a desktop computer, an all-in-one system, a server, a network device, a wireless device, or any other type of device capable of executing the instructions 322 and 324. In certain examples, the computing device 300 may include or be connected to additional components such as memories, controllers, etc.

[0036] The processor 310 may be, at least one central processing unit (CPU), at least one semiconductor-based microprocessor, at least one graphics processing unit (GPU), a microcontroller, special purpose logic hardware controlled by microcode or other hardware devices suitable for retrieval and execution of instructions stored in the machine-readable storage medium 320, or combinations thereof. The processor 310 may fetch, decode, and execute instructions 322 and 324 to implement compressing and loading the message into the send buffer. As an alternative or in addition to retrieving and executing instructions, the processor 310 may include at least one integrated circuit (IC), other control logic, other electronic circuits, or combinations thereof that include a number of electronic components for performing the functionality of instructions 322 and 324.

[0037] The machine-readable storage medium 320 may be any electronic, magnetic, optical, or other physical storage device that contains or stores executable instructions. Thus, the machine-readable storage medium 320 may be, for example, Random Access Memory (RAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), a storage drive, a Compact Disc Read Only Memory (CD-ROM), and the like. As such, the machine-readable storage medium 320 can be non-transitory. As described in detail below, machine-readable storage medium 320 may be encoded with a series of executable instructions for compressing and loading the message into the send buffer.

[0038] Moreover, the instructions 322 and 324 when executed by a processor (e.g., via one processing element or multiple processing elements of the

processor) can cause the processor to perform processes, such as, the process of FIG. 4. For example, the determine instructions 322 may be executed by the processor 310 to determine a first output size for a first process and a second output size for a second process.

[0039] The compress instructions 324 may be executed by the processor 310 to compress and load any messages from the first and second processes into a send buffer allocated at a kernel of an OS of a first node (not shown). The messages of the first and second processes may be compressed and/or loaded concurrently. The messages of the first process may be compressed to the first output size and the messages of the second process may be compressed to the second output size.

[0040] The messages of the first and second process may be compressed during a window of time that the kernel is waiting to output the send buffer as a single transfer across a network to a second node. The send buffer may be output from the kernel to the second node without participation from the first and second processes.

[0041] FIG. 4 is an example flowchart of a method 400 for compressing and loading a message into a send buffer. Although execution of the method 400 is described below with reference to the system 200, other suitable components for execution of the method 400 can be utilized, such as the system 100. Additionally, the components for executing the method 400 may be spread among multiple devices (e.g., a processing device in communication with input and output devices). In certain scenarios, multiple devices acting in coordination can be considered a single device to perform the method 400. The method 400 may be

implemented in the form of executable instructions stored on a machine-readable storage medium, such as storage medium 320, and/or in the form of electronic circuitry.

[0042] At block 410, a first node 210 determines output sizes separately for each of a plurality of processes 272-1 to 272-m. The plurality of processes 272-1 to 272-m may transmit a plurality of messages 142-1 to 142-n. At block 420, the first node 210 compresses at least one of the plurality of messages 142-1 to 142-n to the output size corresponding to the process 272 sending the at least one message 142. The compressing, at block 420 may determine a type of compression scheme 252 based on a dynamically varying threshold 254.

[0043] The threshold 254 may be based on at least one of a size of the message 142, a type of the process 256, a compression ratio 258 of a previous message of the process, and a load of a system 257 including the kernel 230. At block 430, the first node 210 loads the compressed message 142' into a send buffer 260 allocated at a kernel 230 of an OS 220 of the first node 210. The plurality of messages 142-1 to 142-n may be concurrently compressed and/or loaded during a window of time that the kernel 230 is waiting to output the send buffer 260 as a single transfer across a network to a second node 280.

CLAIMSWe claim:

1. A system, comprising:
an allocation unit to receive a message from a process, the allocation unit to allocate a portion of a send buffer at a kernel of an operating system (OS) of a first node for the message; and
a compress unit to compress the message and to load the compressed message into the send buffer, wherein
the message is to be compressed during a window of time that the kernel of the first node is waiting to output the send buffer as a single transfer across a network to a second node.
2. The system of claim 1, wherein,
a plurality of the processes are to transmit a plurality of the messages to the allocation unit, and
the allocation unit is to sequentially allocate the plurality of messages to the send buffer.
3. The system of claim 2, wherein the compress unit is to
concurrently at least one of compress and load the plurality of messages to the send buffer.
4. The system of claim 3, wherein the allocation unit is to update the process with a completion indication after the corresponding message of the

process is at least one of compressed and loaded to the send buffer by the compress unit.

5. The system of claim 4, wherein,
at least one of the plurality of processes includes a process buffer to store a message to be sent to the allocation unit, and
the at least one process is to reuse the process buffer after receiving the completion indication from the allocation unit.

6. The system of claim 4, wherein the compress unit is to separately determine a type of compression scheme for each of the plurality of processes based on a threshold.

7. The system of claim 6, wherein,
the threshold includes a minimum threshold to relate to a minimum size of a message to be compressed, and
the compress unit is to not compress the message if the size of the message is less than the minimum threshold.

8. The system of claim 6, wherein,
the threshold includes a size threshold to relate to the size of the compressed message to load to the send buffer, and
the allocation unit is to vary the size threshold for different types of the

processes.

9. The system of claim 6, wherein,
the threshold includes a compression threshold to relate to a type of
compression scheme, and
the compress unit is to base the compression threshold on at least one of
a processor load and free memory of the system.

10. The system of claim 6, wherein,
the threshold includes a history threshold to relate to a compression ratio
of a previously compressed message, and
the compress unit is to select at least one of a different compression
scheme and no compression scheme for a current message based on the
compression ratio of the previously compressed message.

11. The system of claim 2, wherein,
the compress unit is compress each of a plurality of messages of a first
process of the plurality of processes to a same fixed size, and
the compress unit is to compress a remainder of a first message of the
plurality of messages of the first process into a separately compressed
message, if an entirety of the first message cannot be compressed to the same
fixed size.

12. A method, comprising:

determining output sizes separately for each of a plurality of processes, the plurality of processes transmit a plurality of messages;

compressing at least one of the plurality of messages to the output size corresponding to the process sending the at least one message; and

loading the compressed message into a send buffer allocated at a kernel of an operating system (OS) of a first node, wherein

the plurality of messages are to be concurrently at least one of compressed and loaded during a window of time that the kernel is waiting to output the send buffer as a single transfer across a network to a second node.

13. The method of claim 12, wherein,

the compressing is to determine a type of compression scheme based on a dynamically varying threshold, and

the threshold to be based on at least one of a size of the message, a type of the process, a compression ratio of a previous message of the process, and a load of a system including the kernel.

14. A non-transitory computer-readable storage medium storing instructions that, if executed by a processor of a device, cause the processor to:

determine a first output size for a first process and a second output size for a second process; and

compress and load any messages from the first and second processes into a send buffer allocated at a kernel of an operating system (OS) of a first node, wherein

the messages of the first process are compressed to the first output size and the messages of the second process are compressed to the second output size, and

the messages of the first and second process are compressed during a window of time that the kernel is waiting to output the send buffer as a single transfer across a network to a second node.

15. The non-transitory computer-readable storage medium of claim 14, wherein,

the messages of the first and second processes are at least one of compressed and loaded concurrently, and

the send buffer is output from the kernel to the second node without participation from the first and second processes.

FIG. 1

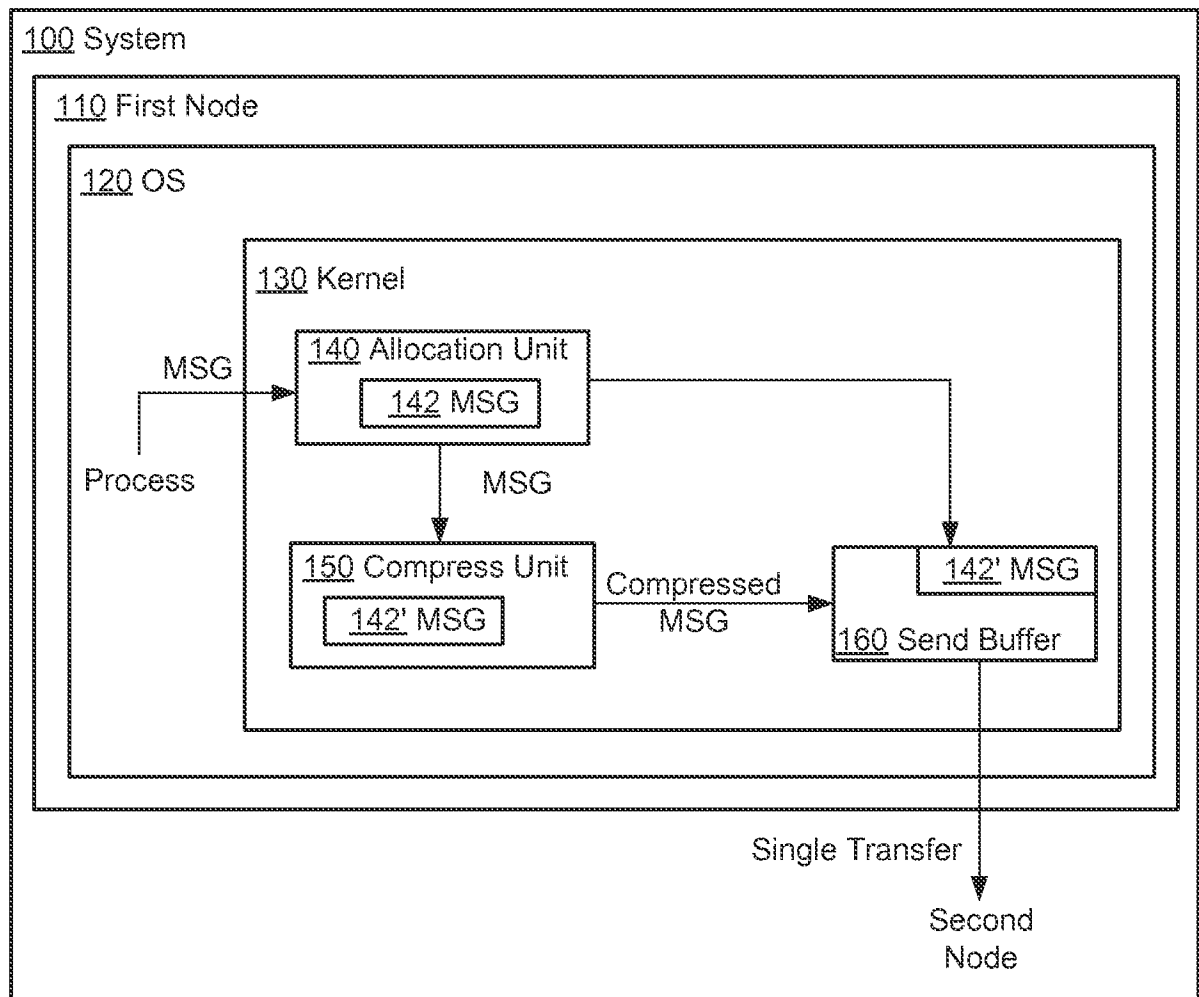


FIG. 2

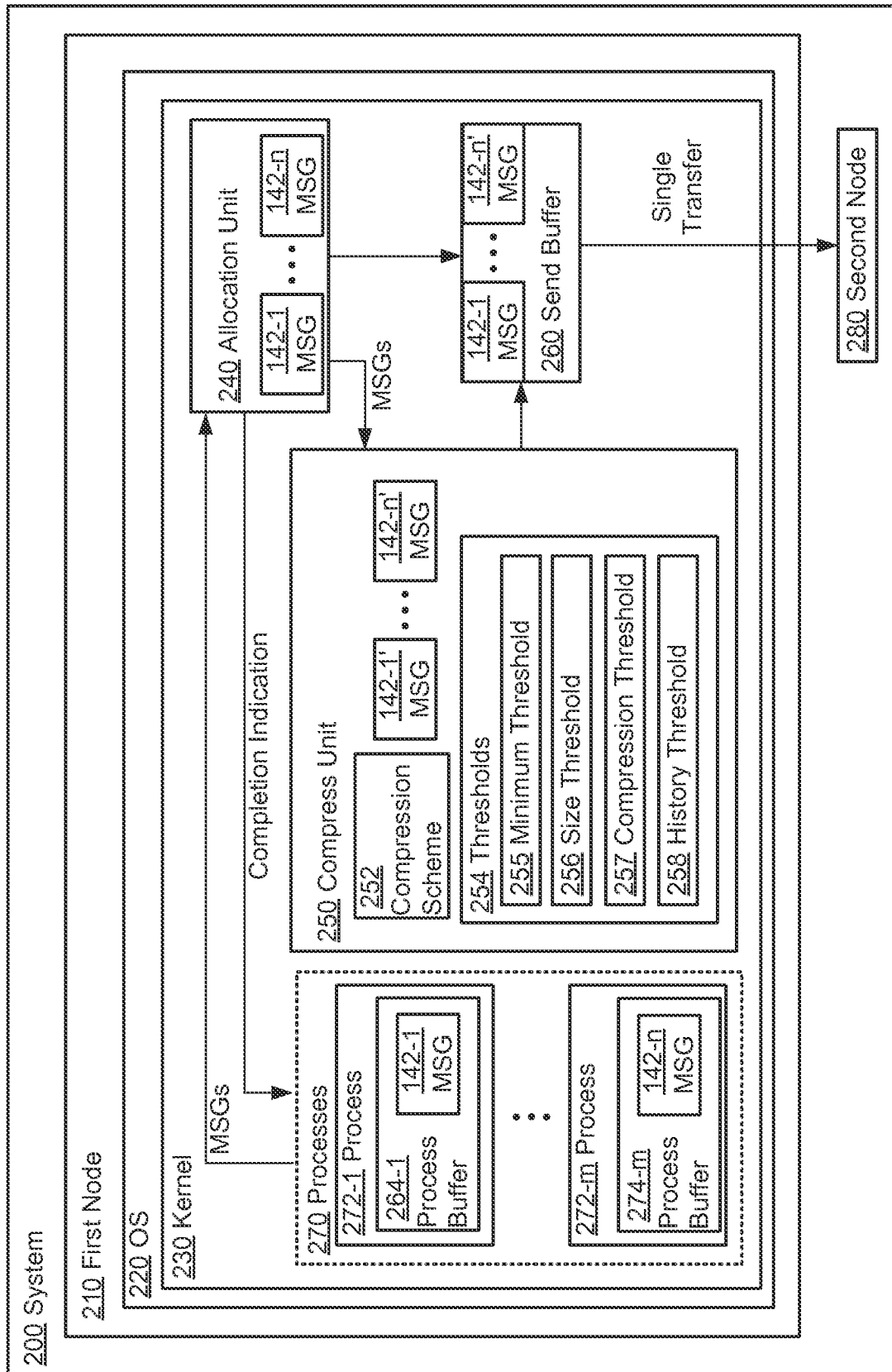


FIG. 3

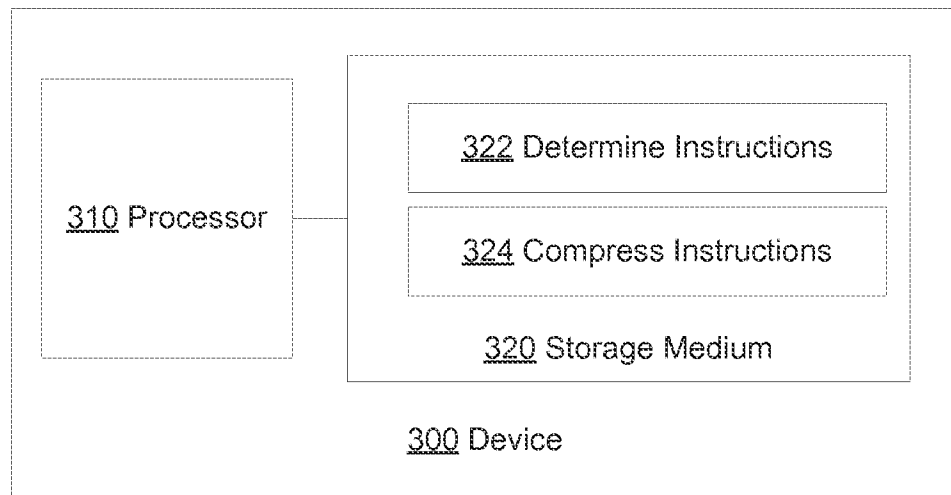
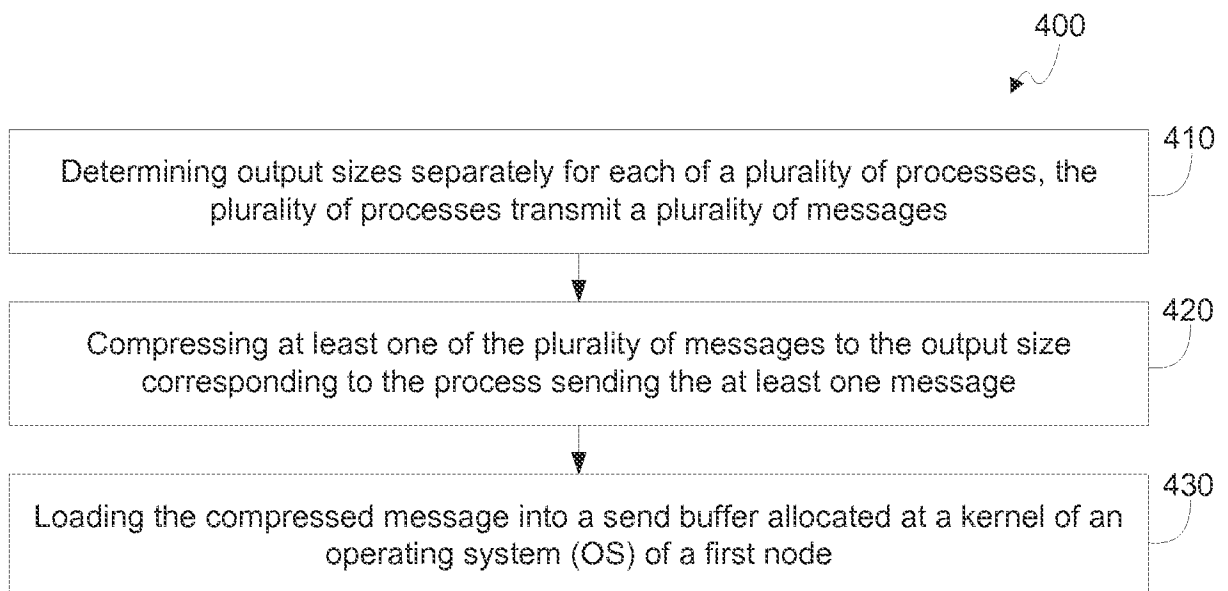


FIG. 4



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/024325**A. CLASSIFICATION OF SUBJECT MATTER****H04L 12/861(2013.01)i, H04L 12/58(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHEDMinimum documentation searched (classification system followed by classification symbols)
H04L 12/861; G06F 15/173; G06F 17/30; H04L 29/06; G06F 9/46; G06F 9/00; H04L 12/58Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility modelsElectronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: allocation, send, buffer, compression, message, load, buffer, kernel, os**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2009-0198826 A1 (YOSHIHIRO ISHIJIMA) 06 August 2009 See paragraphs [0001], [0016], [0035]; and figure 1.	1-3, 11-15
A		4-10
Y	US 2014-0281034 A1 (FUTUREWEI TECHNOLOGIES, INC.) 18 September 2014 See paragraphs [0003], [0029]; and figure 2.	1-3, 11-15
A	US 6757904 B1 (BRYAN A. WOODRUFF et al.) 29 June 2004 See column 3, line 18 - column 8, line 53; and figures 1-5.	1-15
A	US 2007-0011687 A1 (ADNAN ILIK et al.) 11 January 2007 See paragraphs [0016]-[0032]; and figures 2-4d.	1-15
A	US 2005-0192941 A1 (STEFAN BIEDENSTEIN et al.) 01 September 2005 See paragraphs [0019]-[0031]; and figures 1-2B.	1-15

☐ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

15 December 2015 (15.12.2015)

Date of mailing of the international search report

16 December 2015 (16.12.2015)

Name and mailing address of the ISA/KR

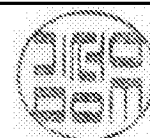
International Application Division
Korean Intellectual Property Office
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan City, 35208,
Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

KIM, Seong Woo

Telephone No. +82-42-481-3348



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/024325

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0198826 A1	06/08/2009	US 08352957 B2	08/01/2013
US 2014-0281034 A1	18/09/2014	WO 2014-139434 A1	18/09/2014
US 6757904 B1	29/06/2004	None	
US 2007-0011687 A1	11/01/2007	None	
US 2005-0192941 A1	01/09/2005	US 07263520 B2	28/08/2007
		US 07650326 B2	19/01/2010
		US 2008-0016322 A1	17/01/2008