US 20080294669A1

(19) **United States**

(12) **Patent Application Publication**

Narayanan

(10) **Pub. No.: US 2008/0294669 A1**

(43) **Pub. Date: Nov. 27, 2008**

(54) **PROGRAM-DATA COMBINING SYSTEM**

(76) Inventor: **Sarukkai R. Narayanan**, Oklahoma City, OK (US)

Correspondence Address:
DUNLAP CODDING, P.C.
PO BOX 16370
OKLAHOMA CITY, OK 73113 (US)

(21) Appl. No.: **12/121,394**

(22) Filed: **May 15, 2008**

Related U.S. Application Data

(60) Provisional application No. 60/931,153, filed on May 21, 2007.

Publication Classification

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** **707/101**; 707/E17.009

(57) **ABSTRACT**

Methods for securing at least one program application and data associated with the program application are herein described. In one embodiment, the method includes compressing the program application and data associated with the program application into a program-data combine having a compressed format and storing the program-data combine on a program-data combine computer.

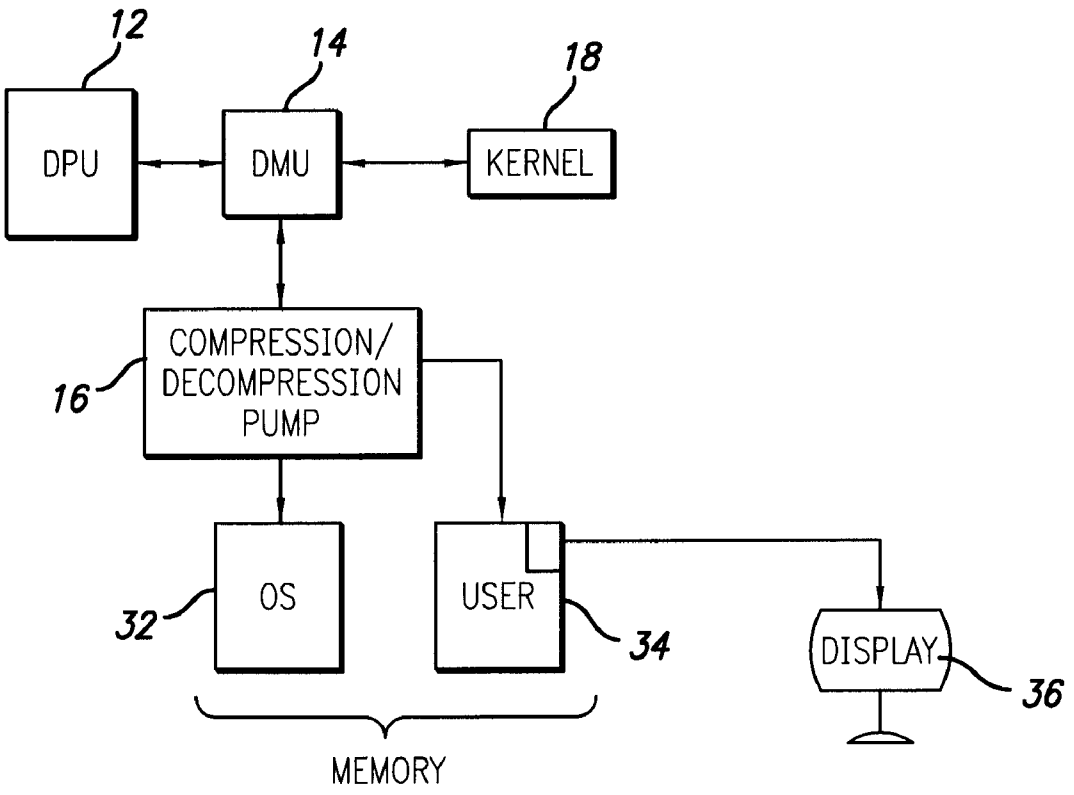


FIG. 1

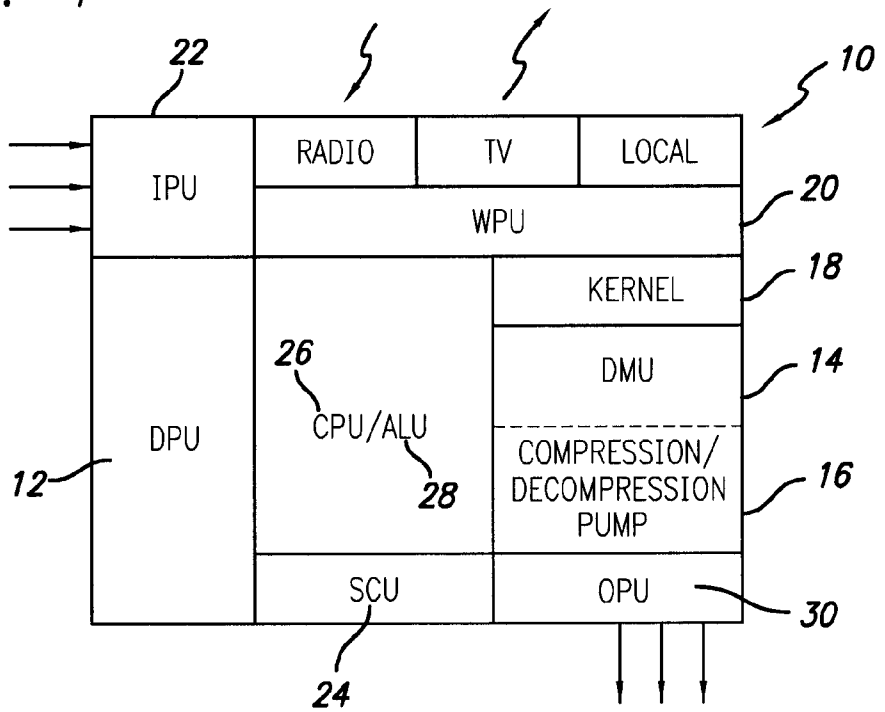
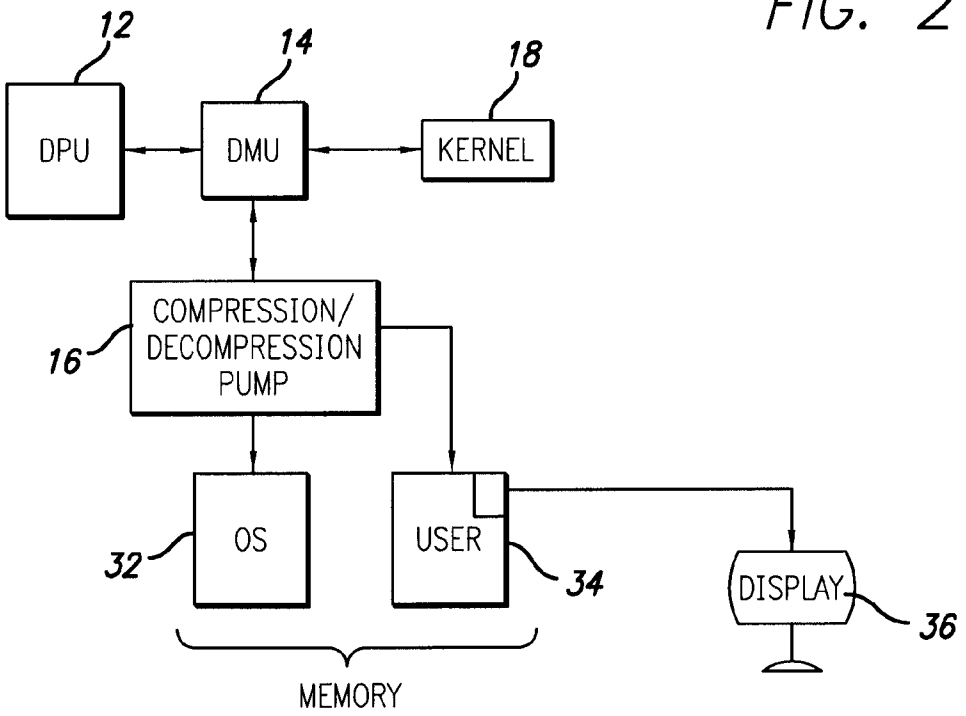

22
10
IPU
RADIO
TV
LOCAL
WPU
20
KERNEL
18
26
DMU
14
CPU/ALU
28
DPU
12
COMPRESSION/
DECOMPRESSION
PUMP
16
SCU
OPU
30
24

FIG. 2


12
14
18
DPU
DMU
KERNEL
COMPRESSION/
DECOMPRESSION
PUMP
16
OS
USER
32
34
DISPLAY
36
MEMORY

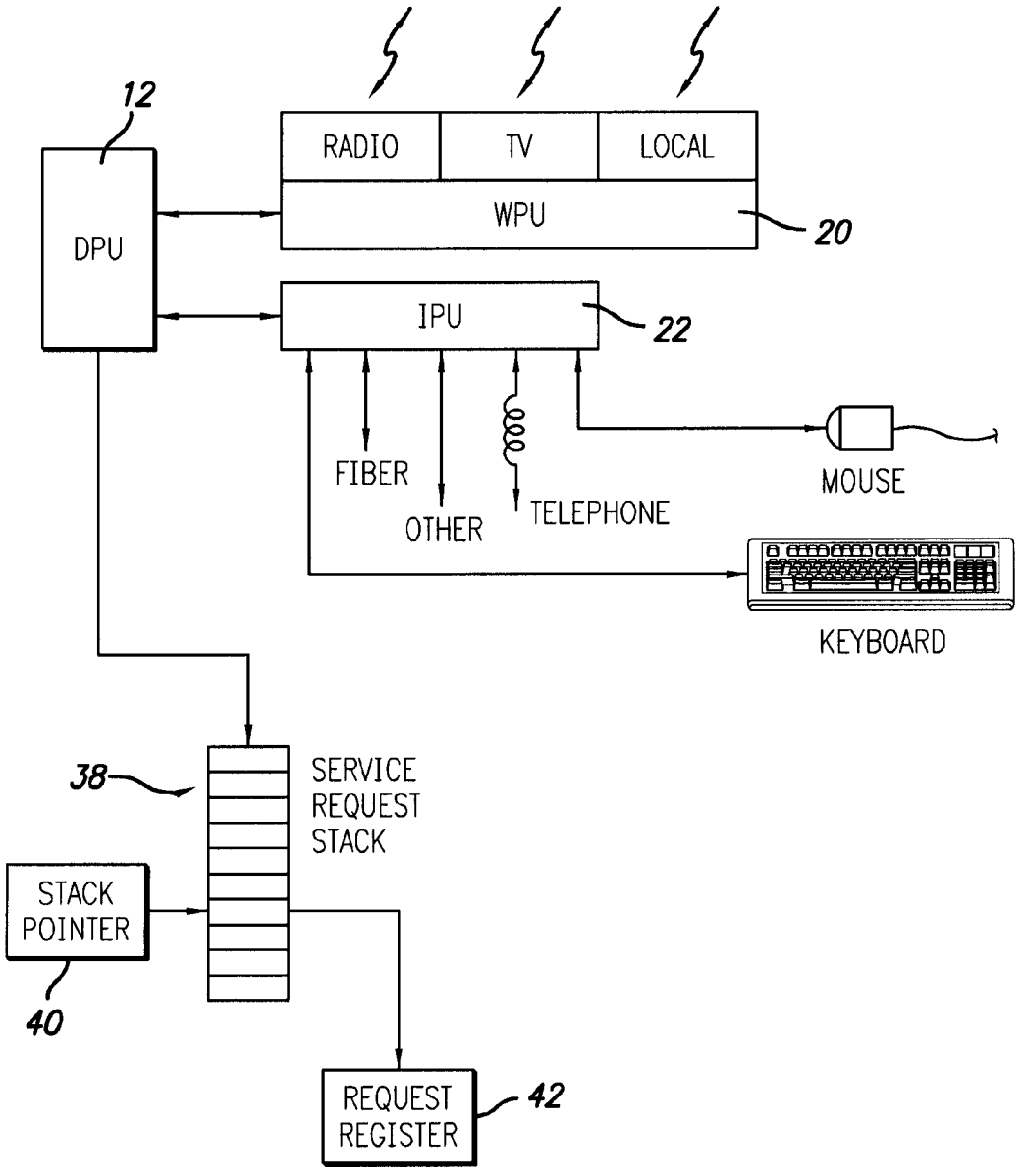

12
RADIO
TV
LOCAL
DPU
WPU
20
IPU
22
FIBER
OTHER
TELEPHONE
MOUSE
KEYBOARD
38
SERVICE
REQUEST
STACK
STACK
POINTER
40
REQUEST
REGISTER
42

FIG. 3

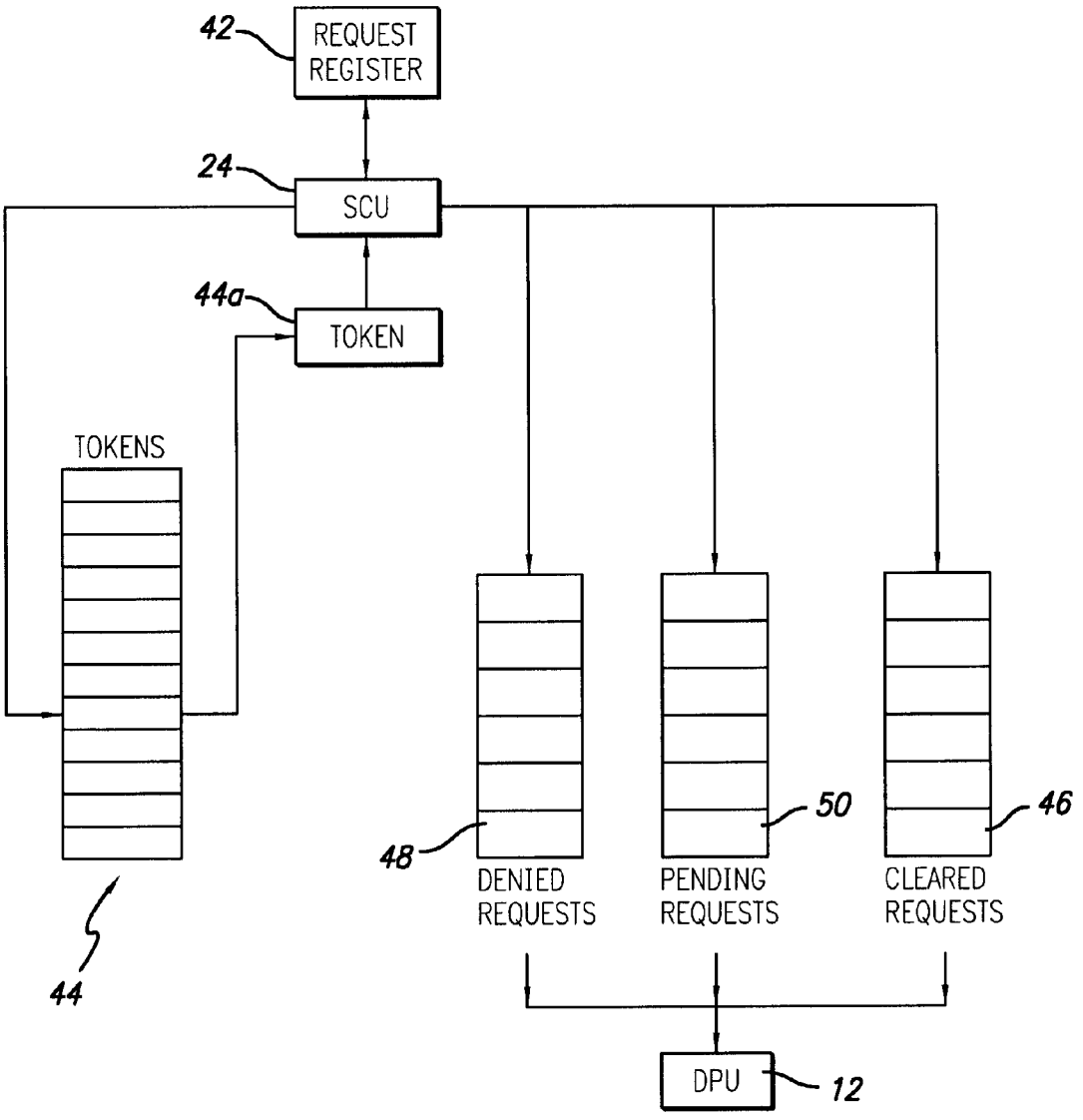

42
REQUEST REGISTER
24
SCU
44a
TOKEN
TOKENS
44
48
DENIED REQUESTS
50
PENDING REQUESTS
46
CLEARED REQUESTS
DPU
12

FIG. 4

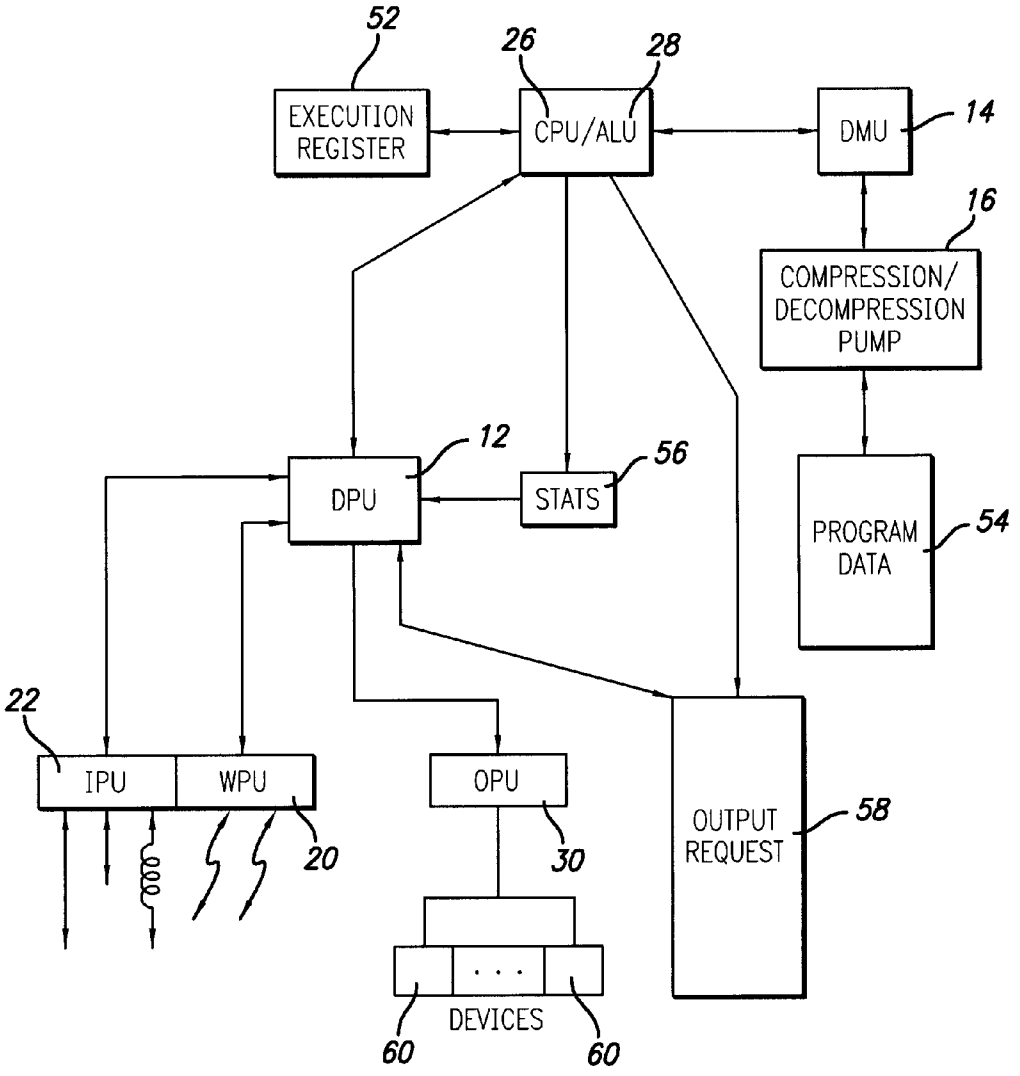

52
26
28
EXECUTION REGISTER
CPU/ALU
DMU
14
16
COMPRESSION/ DECOMPRESSION PUMP
12
56
DPU
STATS
PROGRAM DATA
54
22
IPU
WPU
OPU
OUTPUT REQUEST
58
20
30
DEVICES
60
60

FIG. 5

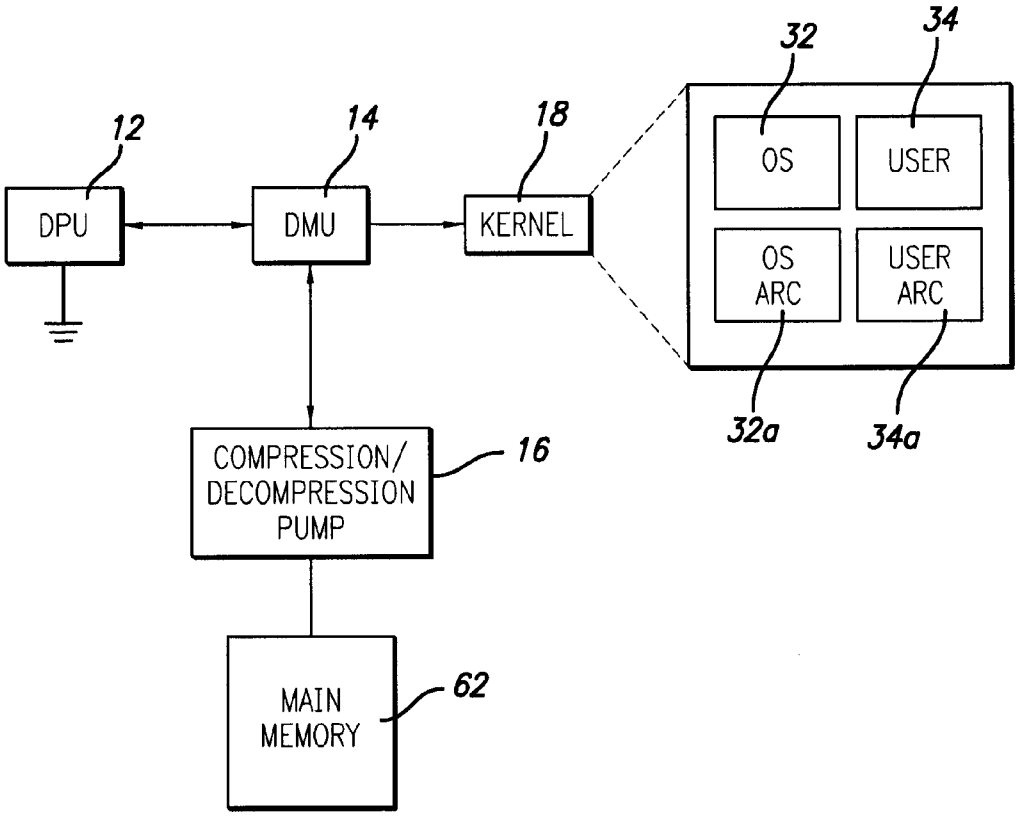

12
14
18
32
34
DPU
DMU
KERNEL
OS
USER
OS ARC
USER ARC
32a
34a
COMPRESSION/ DECOMPRESSION PUMP
16
MAIN MEMORY
62

FIG. 6

PROGRAM-DATA COMBINING SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Provisional Application Ser. No. 60/931,153, filed on May 21, 2007, the entire disclosure of which is hereby incorporated into this disclosure.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] Not applicable.

NAMES OF PARTIES TO A JOINT RESEARCH AGREEMENT

[0003] Not applicable.

REFERENCE TO A "SEQUENCE LISTING," A TABLE, OR A COMPUTER PROGRAM LISTING APPENDIX SUBMITTED ON A COMPACT DISC AND AN INCORPORATION BY REFERENCE OF THE MATERIAL ON THE COMPACT DISC

[0004] Not applicable.

BACKGROUND

[0005] 1. Field of the Invention

[0006] The present invention relates generally to a program-data combining system. More specifically, but not by way of limitation, the invention relates to storage and retrieval of programs and data as a single unit.

[0007] 2. Brief Description of Related Art

[0008] Programs are a set of instructions developed to accomplish a certain objective. Generally, programs operate on input data and stored data to accomplish the desired result. The output data produced as a result is either stored or provides instructions to external device(s) for further action.

[0009] The bifurcation between programs and data provides great flexibility. For example, one program may be able to operate on several different sets of data or several different programs may operate on the same data. With this flexibility, however, there is vulnerability.

[0010] Within bifurcated programs and data, a "rogue program" or a virus may operate on any file containing the programs and/or data in the system and corrupt the file. Such viruses may flood the computer system with spurious data, place the processor in a "loop," or other similar malfeasant acts that hinder the processor from productive work.

[0011] Currently within the field of computer technology, immunity from attacks by computer viruses is a daily threat. Security in most systems is not built in, but instead added on. For example, currently implemented security systems include firewalls, anti-virus software, and other similar mechanisms.

[0012] One of the principle reasons attacks are so prevalent is the ability to exploit the weakness in the separation between programs and data. This separation between the programs and data is an easy target to prey upon.

[0013] Within the art, separation of programs and data has been necessitated by the fact that programs and data can get arbitrarily large. As such, associating programs and data as a single entity becomes a practical improbability due to storing and transmission considerations. Great strides have been made, however, in the compression of programs and data increasing the feasibility of joining programs and data for storage and transmission.

[0014] For example, compression of data as disclosed in U.S. Pat. No. 7,298,293, entitled, "METHOD OF ENCODING DATA", produces a compressed set of data that is only a small fraction of the size of the program and/or data that it operates on. See U.S. Pat. No. 7,298,293, filed on May 18, 2006, the entire contents of which is hereby incorporated into this disclosure. Subsequent provisional applications disclose decompression of this compressed set to reproduce the original program and/or data. See U.S. Provisional Patent No. 61/016,022, filed on Dec. 21, 2007, and U.S. Provisional Patent No. 61/038,527, filed on Mar. 21, 2008, the entire contents of which are hereby incorporated by reference in their entirety.

[0015] The use of these compression techniques, and/or other similar compression techniques, provide a mechanism for combining program and data into a single entity for storage and transmission.

BRIEF SUMMARY OF THE EMBODIMENTS

[0016] The present invention advances the concepts of securing computer systems. In one embodiment, programs and data are treated as a unit. This unit is derived in a compressed format and referred to as a program-data combine. Within a program-data combine, a program may only operate and/or modify the data that is associated with the program. For example, a first program and its data are compressed to form a program-data combine. The program-data combine is stored on the program-data combine computer. A second program and/or external computer requests use of the data within the first program. The program-data combine computer provides the data to the second program, but only as a program-data combine. The second program must have the needed software and/or hardware to decompresses the program-data combine, modify the data, and recompress the program and data to provide a new version of the program-data combine to be sent back to the program-data combine computer leaving the original program-data combine unaltered. Generally, the program within the program-data combine is only operable on the data space attached to it in the program-data combine computer.

[0017] Further, the program-data combine capitalizes on the efficiencies gained from the compression and decompression of data. Generally, program-data combines are received from sources external to the program-data combine computer. Having the program-data combines in a compressed format allows for efficient transmission and storage of programs and data.

BRIEF DESCRIPTION OF SEVERAL VIEWS OF THE DRAWINGS

[0018] So that the above recited features and advantages of the present invention can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to the embodiments thereof that are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only typical embodiments of the invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

[0019] FIG. **1** is a block diagram of one embodiment of a program-data combine computer system.

[0020] FIG. **2** is a flow chart of one embodiment of a method for powering up components for use in the program-data combine computer system of FIG. **1**.

[0021] FIG. **3** is a flow chart of one embodiment of a method for service request processing for use in the program-data combine computer system of FIG. **1**.

[0022] FIG. **4** is a flow chart of one embodiment of a method for security clearance processing for use in the program-data combine computer system of FIG. **1**.

[0023] FIG. **5** is a flow chart of one embodiment of a method for providing data and output processing for use in the program-data combine computer system of FIG. **1**.

[0024] FIG. **6** is a flow chart of one embodiment of a method for providing power down processing for use in the program-data combine computer system of FIG. **1**.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0025] Present embodiments of the invention are shown in the above-identified figures and described in detail below. In describing the embodiments, like or identical reference numerals are used to identify common or similar elements. The figures are not necessarily to scale and certain features in certain views of the figures may be shown exaggerated in scale or in schematic in the interest of clarity and conciseness.

[0026] Referring now to the figures, and in particular to FIG. **1**, shown therein and designated by reference numeral **10** is a program-data combine computer for interrelating program and data information. The program-data combine computer **10** provides a mechanism for removing the bifurcation of programs and data in the current art. The program-data combine computer may also herein be referred to as a spinner **10**.

[0027] In the spinner **10**, programs and data are treated as one entity known as the program data compact (PDC). Only programs within the PDC may manipulate the data within the PDC. Generally, the program-data compact is received, acted upon, stored, and/or transmitted in a compressed format.

[0028] Generally, the program-data combine computer **10** includes a director processing unit (DPU) **12** that controls the functions of spinner **10**. The DPU **12** is capable of activating and/or deactivating any of the components within the spinner **10**. The DPU **12** may be any computational device capable of receiving and providing instructions to other computational devices.

[0029] The DPU **12** communicates with a data management unit (DMU) **14**. The DMU **14** is capable of performing memory management functions required within the spinner **10**. The DMU **14** may be any memory storage device capable of storing PDCs.

[0030] The DPU **12** within the spinner **10** oversees a variety of processes such as the following six separate processes to provide and utilize the PDC: the power up sequence, a service request process, security clearance process, data processing, output processing, and a power down sequence and other functions as may be warranted.

[0031] The power up sequence provides the operating system and memory at the time of the last power down sequence. During this process, the DPU **12** communicates directly with a compression/decompression pump (C/D/pump) **16**. Generally, the C/D pump **16** performs the process of compressing programs and/or data and decompressing such programs and/or data on demand. In the power up process, the C/D pump **16** decompresses data from a kernel to obtain the operating system and memory at the time of the last power down of the spinner **10**.

[0032] Once the spinner **10** is powered up, the DPU **12** can coordinate service requests from external sources through the service request process. Service requests are generally provided by wireless sources and/or direct sources.

[0033] The wireless sources are managed through a wireless processing unit (WPU) **20**. The WPU **20** receives signals from wireless sources such as radio, TV, local area wireless networks, and/or the like.

[0034] The direct connect sources are managed through an input processing unit (IPU) **22**. The IPU **22** manages data input from devices in direct communication with the spinner **10**. Devices in direct communication with the spinner **10** may include a keyboard, mouse, telephone line, fiber optic line, and/or the like.

[0035] Certain service requests may require security clearance. A security clearance unit (SCU) **24** manages security clearance aspects for the service requests acted upon by various components of the spinner **10**.

[0036] A service request cleared by the SCU **24** is able to be processed by a central processing unit (CPU) **26** within the spinner **10**. The CPU generally abstracts each instruction within an active program and executes it until completion. An arithmetic logic unit (ALU) **28** provides assistance to the CPU **26** during execution of the active program by performing the arithmetic and logic functions encoded within the active program.

[0037] Output provided by execution of the program is processed by an output processing unit (OPU) **30**. The OPU **30** manages the output and provides the output to other devices external to the spinner **10**. For example, the OPU **30** may provide output from the executed program to printers, fax machines, telephone line, external spinner, and/or the like.

[0038] Interaction between components of the spinner **10** illustrated in FIG. **1** provides for the storage, modification, transmission, and retrieval of the PDC. Each of the processes designed to provide and utilize the PDC are described in further detail herein below.

1. Power-Up Sequence

[0039] FIG. **2** is a block diagram of one embodiment of power up components for use in the spinner **10** of FIG. **1**. The power-up sequence generally provides the operating system and memory from the last power-down sequence.

[0040] Initially, the DPU **12** deactivates all major components within the spinner **10**. After deactivation, the DPU **12** provides an activation signal to the DMU **14**. The activation signal provides the DMU **14** a specific memory address for the power-up sequence known as the "Initial Program Load."

[0041] The DMU **14** searches and provides the data within the kernel **18** to the C/D pump **16** and includes a request to perform an initial load. The C/D pump **16** decompresses the data from the kernel **18**. Decompressing data from the kernel **16** provides an operating system **32** and user memory **34** at the time of the last power down sequence. The operating system **32** and user memory **34** at the time of the last power down sequence is loaded by the C/D pump **16** to the memory address, provided by the DMU **14** thereby accomplishing an "Initial Program Load". The operating system **32** and user memory **34** are then capable of being displayed on an external

device **36**. For example, the operating system **32** and user memory **34** may be displayed on a monitor. Once the operating system **32** and user memory **34** are loaded starting at the designated memory location, the DMU **14** signals the DPU **12** the "Initial Program Load" and therefore the power up sequence is complete.

2. Service Request Process

[0042] FIG. **3** is a block diagram of one embodiment of a method for service request processing. Generally, a service request process for the spinner **10** may be provided by several sources. The sources may be broadly grouped into two categories: wireless and direct connect.

[0043] The wireless sources are managed through the WPU **20**. The WPU **20** receives signals from wireless sources such as radio, TV, local area wireless networks, and/or the like.

[0044] The direct connect sources are managed through the IPU **22**. The IPU **22** manages data input from devices in direct communication with the spinner **10**. Devices in direct communication with the spinner **10** may include a keyboard, mouse, telephone line, fiber optic line, and/or the like.

[0045] The DPU **12** receives at least one request for service for the spinner **10** from the WPU **20** and/or the IPU **22**. The DPU **12** catalogs the request for service on a service request stack **38**. In one embodiment, the catalog provides a mechanism for prioritizing two or more requests for service from the spinner **10**. The catalog may include one or more prioritizing criteria for the two or more requests for service. Examples of prioritizing criteria may include time, memory space, and/or the like.

[0046] A stack pointer **40** selects the request for service within the request stack **38**. The stack pointer **40** places the request for service in a request register **42**.

[0047] If the request for service requires security clearance, the DPU **12** signals the SCU **24** to request security clearance as illustrated in FIG. **4**. Specifically, the DPU **12** inserts information into the request register **42** if the request for service requires security clearance or further security clearance processing.

3. Security Clearance Processing

[0048] FIG. **4** is a block diagram of one embodiment of a method for security clearance processing. The SCU **24** obtains information from the request register **42** on the request for service through a token **44**. Generally, tokens **44** contain security clearance requirements of PDC in the form of information and/or procedures. For example, tokens **44** may contain the licensing information regarding the specific application, password authentication procedures, instructions regarding the copying and/or dissemination of data, special clearance needs required before the execution of the program is allowed and/or other similar information. Additionally, tokens **44** may contain PDCs requiring execution.

[0049] The SCU **24** receives information from a specific token **44***a*. The specific token **44***a* stores information related to the request for service. The SCU **24** determines whether all of the procedures outlined in the token **44***a* are met. For example, if the token **44***a* contains specific information regarding a password, the SCU **24** determines whether the password has been authenticated.

[0050] Once all of the procedures outlined in the token **44***a* are met, the SCU **24** provides the service request to a cleared request stack **46**. If any of the procedures as outlined in the token **44** are not satisfied, the SCU **24** provides the request for service to a denied request stack **48**. If additional procedures as outlined in the token **44** require further processing, the SCU **24** provides the request for service to a pending request stack **50**.

[0051] Once the SCU **24** provides the request for service to either the cleared request stack **46**, the denied request stack **48**, or the pending request stack **50**, the SCU **24** may provide a signal to the DPU **12** regarding the status of the request for service. The DPU **12** receives the status signal and dispatches at least one message indicating the status of the request for service to the source that initially requested service.

[0052] If the request for service is placed in the cleared request stack **46**, the DPU **12** provides the request for service into an execution register **52** and signals the CPU **26** to process the request for service as illustrated in FIG. **5**.

4. Data Processing/Output Processing

[0053] FIG. **5** is a block diagram of one embodiment of a method for providing data and output processing. Generally, the CPU **26** receives the request for service from the execution register **52** and signals the DMU **14** to retrieve the required PDC **54**. The PDC **54** may be on the cleared request stack **46** (not illustrated in FIG. **5**) or the PDC may be stored within the system memory (not illustrated in FIG. **5**).

[0054] The DMU **14** retrieves the token **44***a* and processes the token **44***a* by placing a time stamp on the token **44***a*, tagging the token **44***a* as "before copy," and/or other similar processing techniques known within the art. The DMU **14** stores the token **44** in the main memory of the PDC **54** as part of user data.

[0055] The PDC **54** is then sent by the DMU **14** to the C/D Pump **16**, or similar pump, capable of decompressing the PDC **54** from its compressed format. Decompressing the PDC **54** will expand the PDC **54** to provide the program and associated data in a decompressed format. The C/D Pump **16** places the decompressed program and data information in the main memory of the spinner **10** at a location requested by the DMU **14**. The C/D Pump **16** then signals the CPU **26** to commence execution of the decompressed program and operate on the decompressed data.

[0056] As the program execution continues, the CPU **26** and the ALU **28** generate statistics **56** on the progress of the program to monitor program execution. Such statistics **56** may include information on whether the request for memory has exceeded a certain threshold, if the program is in a loop, and/or other similar information. Based on these statistics **56**, a signal may be sent by the DPU **12** in order to disable the CPU **26** and/or ALU **28** and terminate execution of the program if needed.

[0057] At the termination of program execution, the CPU **26** signals the C/D Pump **16** to recompress the program and data into the PDC **54** format. The CPU **26** may also instruct the C/D Pump **16** to create an "after copy" of the token **44***a* with the resulting PDC attached. The "after copy" of the token **44***a* is associated with the "before copy" of the token **44***a* and stored in the main memory of the spinner **10** as part of the user data. If output was requested and cleared at the termination of the program execution, the output may be provided to an output request stack **58** in the form of the "after copy" of the token **44***a*.

[0058] The CPU **26** then signals the DPU **12** that program execution is complete. At this point, the DPU **12** directs the OPU **30** to process the output derived from the program

execution. Output may be processed by the OPU **30** according to security clearance restraints instructed within the token **44***a* and provided to external devices **60** such as a parallel port, USB port or serial port for transmission purposes.

[0059] Data processing and output processing are repeated for every request for service on the cleared request stack **46**. When all such requests for service on the cleared request stack **46** have been processed, the spinner **10** goes into a wait state until additional requests are available to be serviced.

5. Power Down Sequence

[0060] FIG. **6** is a block diagram of one embodiment of a method for providing power down processing. Generally, the DPU **12** disables various components of the spinner **10** and signals the DMU **14** to initiate a shut down of the components within the spinner **10**.

[0061] The DMU **14** sends a signal request to the C/D Pump **16** to compress the operating system **32** and user data **34** within the main memory. As previously discussed, user data consists of tokens **44** stored in the main memory **62** of the spinner system **10** may include the "before copy" and the "after copy." The operating system **32** and the user data may be compressed together or compressed separately as needed for storage within the kernel **18**.

[0062] During the power down sequence, the C/D Pump **16** consolidates information into an archived operating system **32***a* and archived user data **34***a* to provide back-up copies of the operating system **32** and user data **34**. Generally, the archives **32***a* and **34***a* would normally be stored in a non-volatile memory (e.g. hard drive or flash memory) device as back up in case the operating system and user data are damaged or lost.

[0063] From the above description, it is clear that the present invention is well adapted to carry out the objects and to attain the advantages mentioned herein, as well as those inherent in the invention. Although the foregoing invention has been described in some detail by way of illustration and example, it will be apparent to those skilled in the art that certain changes and modifications may be practiced without departing from the spirit and scope of the present invention, as described herein. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features described herein.

What is claimed is:

1. A method for securing at least one program application and data associated with the program application, comprising:

compressing the program application and data associated with the program application into a program-data combine having a compressed format;

the program application being adapted to only operate on the data associated with the program application;

storing the program-data combine on a program-data combine computer.

2. The method of claim **1**, further comprising the step of:

transmitting the program-data combine from the program-data combine computer to an external computer, the external computer storing and transmitting the program-data combine in the compressed format.

3. The method of claim **2**, further comprising the steps of:

decompressing, by the external computer, the program-data combine to enable the program application to operate on the associated data in a decompressed state; and,

recompressing, by the external computer, the program application and the associated data to provide a modified program-data combine distinct from the original program data combine for storage and transmission.

4. The method of claim **2**, further comprising the step of:

validating the program-data combine by identifying at least one security feature compressed within the associated program-data combine.

5. The method of claim **4**, wherein the security feature is a time stamp identifying the before and after execution of the program by the external computer.

6. The method of claim **5**, further comprising the step of:

storing the program-data combine within a kernel, wherein the program and the associated data are compressed and stored as separate entities within the kernel.

* * * * *