



US 20090305790A1

(19) **United States**

(12) **Patent Application Publication**
LU et al.

(10) **Pub. No.: US 2009/0305790 A1**

(43) **Pub. Date: Dec. 10, 2009**

(54) **METHODS AND APPARATUSES OF GAME
APPLIANCE EXECUTION AND RENDERING
SERVICE**

(75) Inventors: **YANG LU**, Windsor (CA);
WEIDONG SHI, Windsor (CA)

Correspondence Address:
YANG LU
P.O. BOX 4617
NAPERVILLE, IL 60567 (US)

(73) Assignee: **VITIE INC.**, Naperville, IL (US)

(21) Appl. No.: **12/544,701**

(22) Filed: **Aug. 20, 2009**

Related U.S. Application Data

(63) Continuation-in-part of application No. 11/668,479, filed on Jan. 30, 2007, Continuation-in-part of application No. 11/670,968, filed on Feb. 3, 2007, Continuation-in-part of application No. 11/965,723, filed on Dec. 27, 2007, Continuation-in-part of application No. 11/927,684, filed on Oct. 30, 2007.

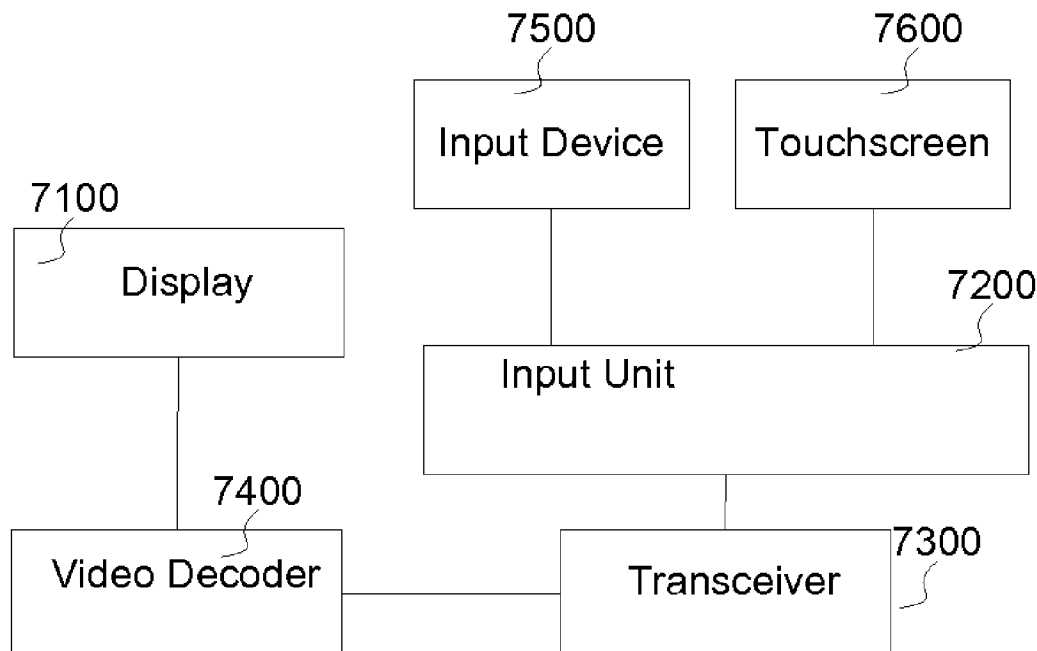
Publication Classification

(51) **Int. Cl.**
A63F 9/24 (2006.01)

(52) **U.S. Cl.** **463/42**

(57) **ABSTRACT**

The present invention presents methods and apparatuses of game appliance execution and rendering service wherein a game appliance system can start a game appliance for a client system and said client system can send input to the game appliance system via a network to control game play of the game appliance and in return receive media stream of the game appliance via a network. Said game appliance system comprising, at least one game appliance apparatus wherein said game appliance apparatus further comprises, at least one transceiver; at least one control processing element that can be programmed to, on-demand retrieve game appliance from at least one game appliance distribution system, and a plurality of control processing elements wherein each control processing element can be programmed to, execute a game appliance using emulation or virtualization; receive input from client system; capture graphics rendering command of the executed game appliance using virtual graphics driver; and transmit the captured graphics rendering command to at least one graphics rendering apparatus. Said game appliance system can further comprise, at least one graphics rendering apparatus wherein said graphics rendering apparatus can comprise, at least one transceiver wherein said transceiver connects said graphics rendering apparatus with one or a plurality of game appliance apparatuses; a plurality of graphics accelerators; and at least one control processing element that can be programmed to, receive graphics rendering command from at least one game appliance apparatus over network; schedule the received graphics rendering command to the graphics accelerators; direct the graphics accelerators to compress the rendered frame into compressed frame or video; transcode the rendered frames into streaming video; and stream said video over a network to client system.



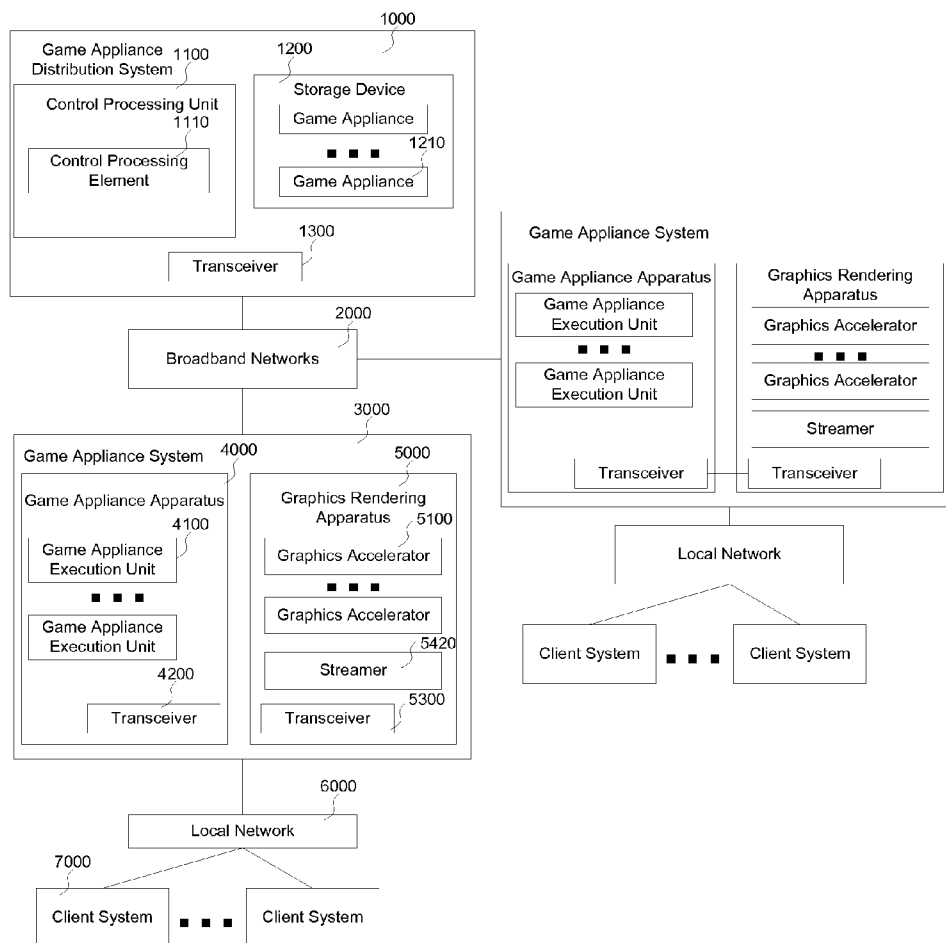


Fig.1

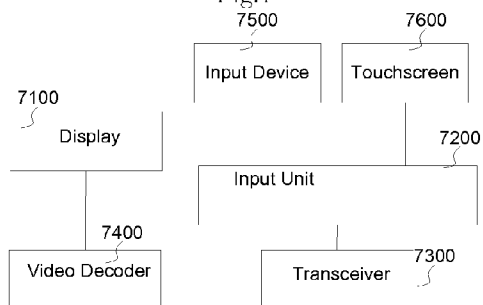


Fig.2

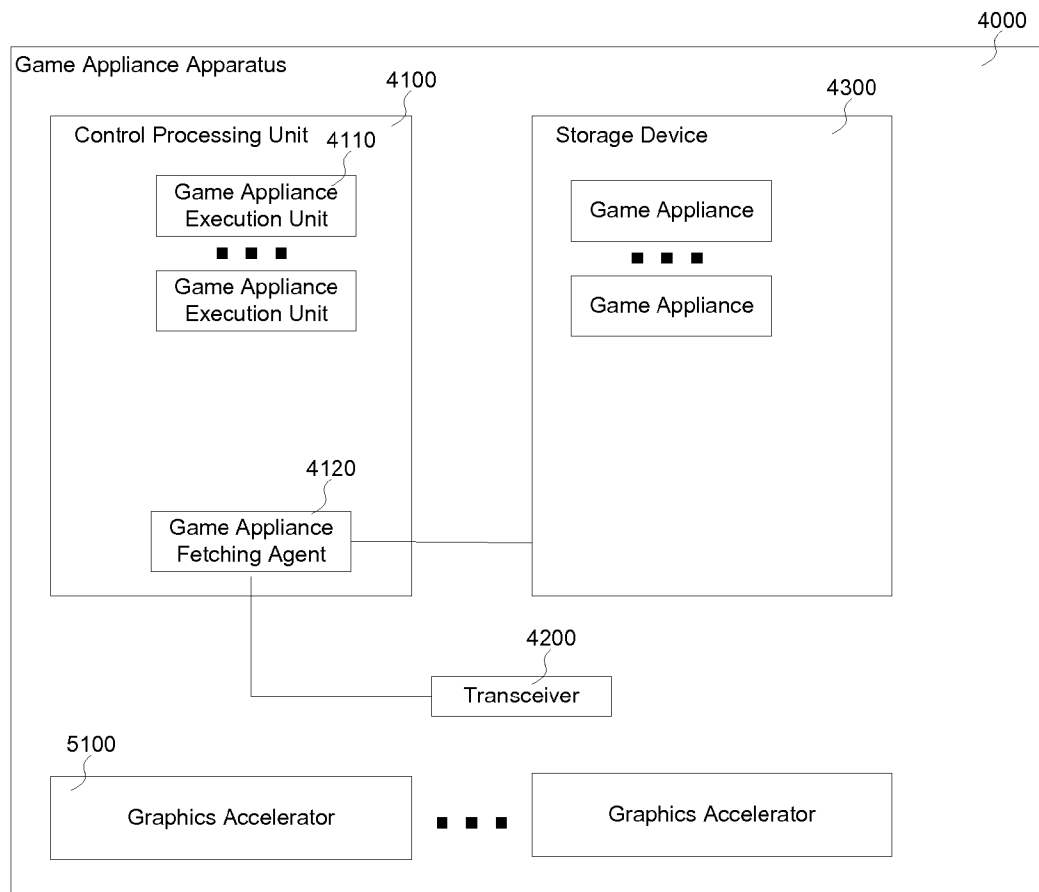


Fig.3

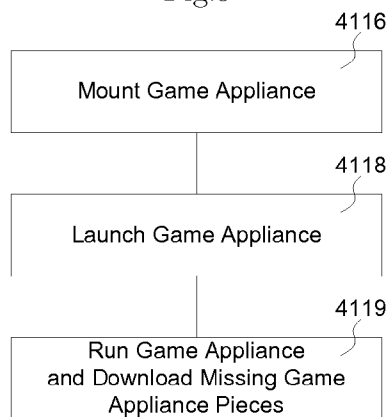


Fig.4

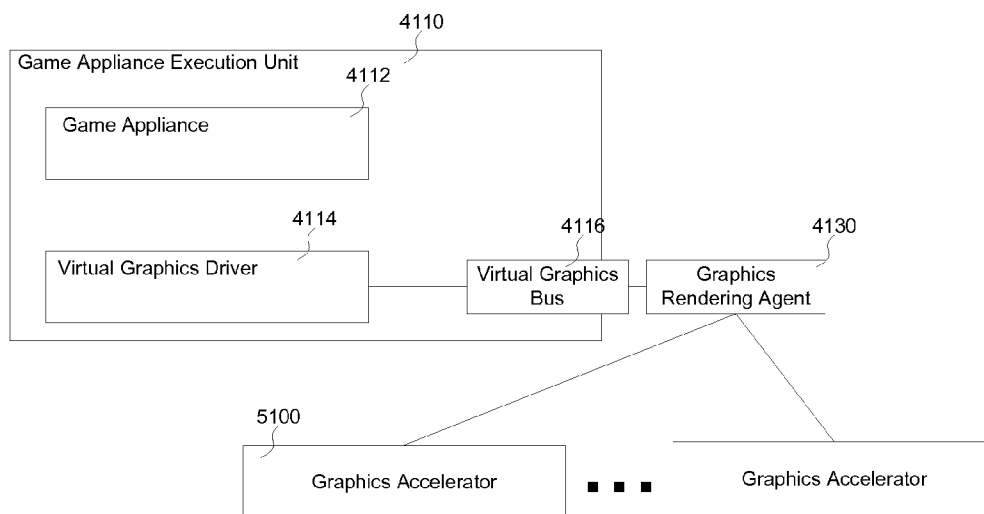


Fig.5

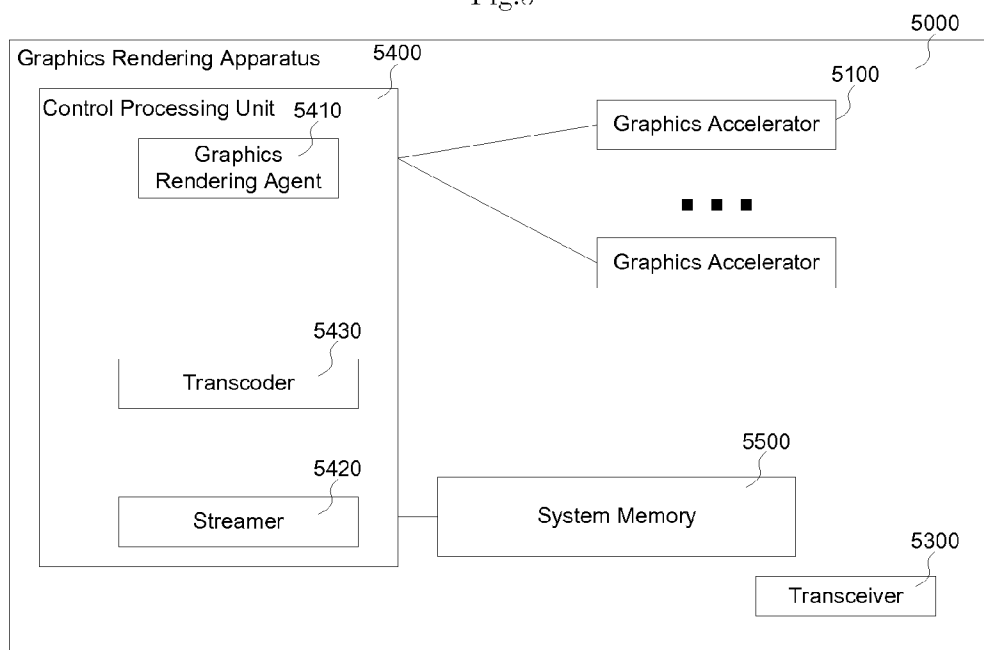


Fig.6

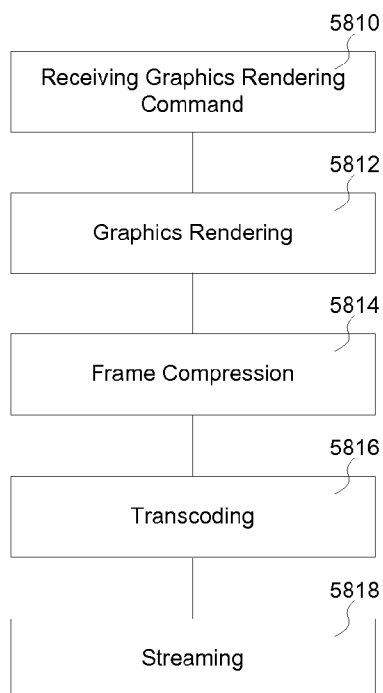


Fig.7

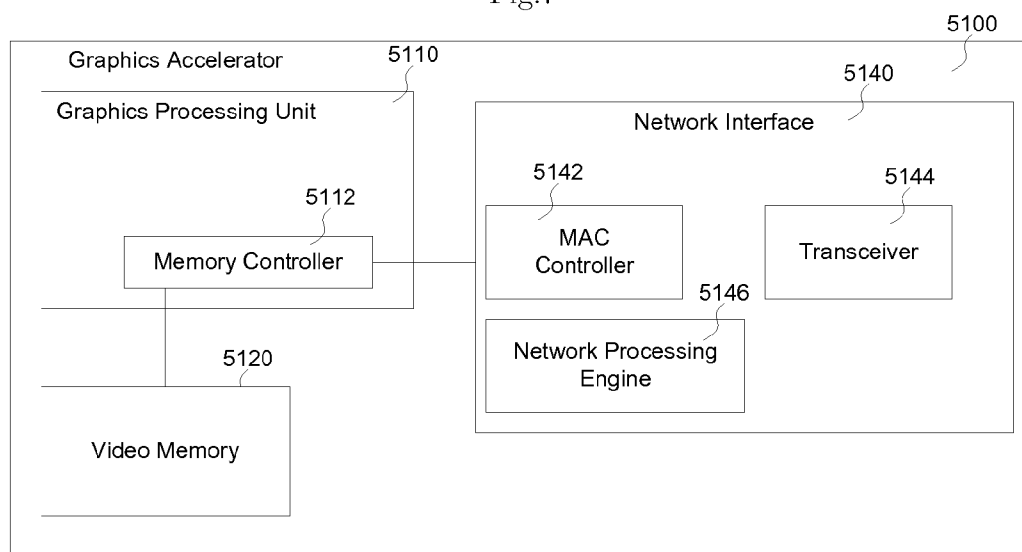


Fig.8

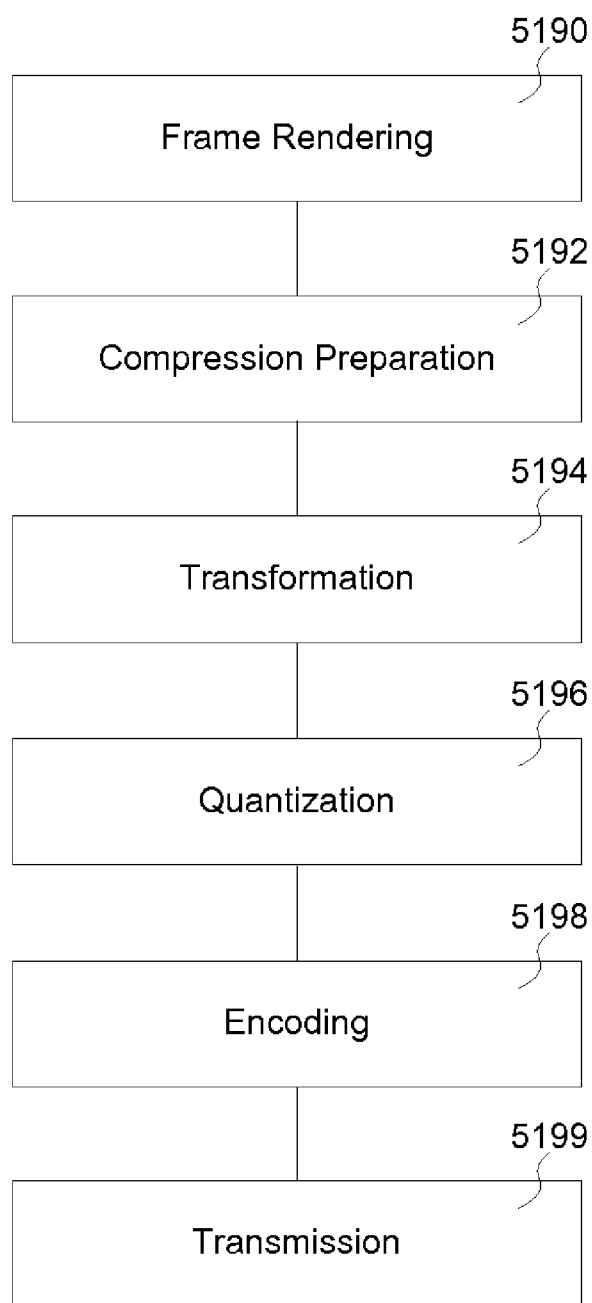


Fig.9

METHODS AND APPARATUSES OF GAME APPLIANCE EXECUTION AND RENDERING SERVICE

[0001] The present application is a continuation-in-part of U.S. application Ser. No. 11/668,479, entitled “Method and Apparatus of Running and Rendering Games Remotely Over Broadband Network”, filed Jan. 30, 2007, which, in turn, relates to and claims priority from U.S. Provisional App. No. 60/763,856, entitled “Method and Apparatus of Playing and Displaying Video Game Remotely Over Broadband Network”, filed Jan. 31, 2006; and U.S. Provisional App. No. 60/765,930, entitled “Method and Apparatus of Hosting and Rendering Video Games Remotely Over Broadband Network”, filed Feb. 7, 2006; The present application is also a continuation-in-part of U.S. application Ser. No. 11/670,968, entitled “Methods and Apparatus of Sharing Graphics Data of Multiple Instances of Interactive Application”, filed Feb. 3, 2007, which, in turn, relates to and claims priority from U.S. Provisional App. No. 60/765,930, entitled “Method and Apparatus of Playing and Displaying Video Game Remotely Over Broadband Network”, filed Jan. 31, 2006; and U.S. Provisional App. No. 60/780,020, entitled “Method and Apparatus of Running Low Cost Interactive Client Over Broadband or Cable Network”, filed Mar. 7, 2006; The present application is also a continuation-in-part of U.S. application Ser. No. 11/965,723, entitled “Methods and Apparatuses of Scalable Pipelined Execution of Graphical Applications”, filed Dec. 27, 2007, which, in turn, relates to and claims priority from U.S. Provisional App. No. 60/877,287, entitled “Methods and Apparatuses of Scalable Pipelined Execution of Graphical Applications”, filed Dec. 28, 2006; The present application is also a continuation-in-part of U.S. application Ser. No. 11/927,684, entitled “Methods and Apparatus of Separating GUI and Graphics Frame in Remote Interactive Graphical Application Execution and Rendering”, filed Oct. 30, 2007, which, in turn, relates to and claims priority from U.S. Provisional App. No. 60/856,190, entitled “Methods and Apparatus of Separating GUI and Graphics Frame in Remote Interactive Graphical Application Execution and Rendering”, filed Nov. 2, 2006; and the present application also claims the benefit of U.S. Provisional App. No. 61/165,789, entitled “Methods and Apparatuses of Software-Image Streaming”, filed Apr. 1, 2009. All of which are hereby incorporated by reference in their entireties.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] This invention relates to a game appliance system that can offer game appliance execution and graphics rendering service wherein said game appliance system can, on-demand fetch game appliance from a game appliance distribution system, execute the game appliance on a game appliance apparatus using emulation or virtualization, capture graphics rendering command of the executed game appliance using virtual graphics driver, render frame for the executed game appliance by a graphics rendering apparatus, compress and transcode the rendered frame into streaming video, and stream the video to at least one client system over a network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The invention may be better understood, and further advantages and uses thereof more readily apparent, when

considered in view of the following detailed description of exemplary embodiments and examples, taken with the accompanying diagrams, in which:

[0005] FIG. 1 is a block diagram showing, in one exemplary embodiment, components of game appliance system and a game appliance distribution system;

[0006] FIG. 2 is a block diagram showing, in one exemplary embodiment, components of a client system;

[0007] FIG. 3 is a block diagram showing, in one exemplary embodiment, components of a game appliance apparatus;

[0008] FIG. 4 is a flowchart showing, in one exemplary embodiment, process used by a game appliance apparatus to launch a game appliance;

[0009] FIG. 5 is a block diagram showing, in one exemplary embodiment, components of game appliance execution unit and additional components of a game appliance apparatus;

[0010] FIG. 6 is a block diagram showing, in one exemplary embodiment, components of a graphics rendering apparatus;

[0011] FIG. 7 is a flowchart showing, in another exemplary embodiment, process of graphics rendering used by a graphics rendering apparatus;

[0012] FIG. 8 is a block diagram showing, in one exemplary embodiment, components of a graphics accelerator; and

[0013] FIG. 9 is a flowchart showing, in one exemplary embodiment, process of frame compression used by a graphics accelerator.

[0014] While the patent invention shall now be described with reference to the embodiments shown in the drawings, it should be understood that the intention is not to limit the invention only to the particular embodiments shown but rather to cover alterations, modifications and equivalent arrangements possible within the scope of appended claims.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0015] Discussion in this section is intended to provide a brief description of some exemplary embodiments of the present invention. The term “game appliance apparatus” in the present application is equivalent to the term “video game host” in U.S. application Ser. No. 11/668,479. The term “game appliance apparatus” in the present application is equivalent to the term “interactive application host” in U.S. application Ser. No. 11/670,968. The term “game appliance apparatus” in the present application is equivalent to the term “graphical application execution host” in U.S. application Ser. No. 11/965,723. The term “game appliance apparatus” in the present application is also equivalent to the term “interactive graphical application execution host” in U.S. application Ser. No. 11/927,684. The term “game appliance apparatus” is also equivalent to the term “software-image streaming client” in U.S. Provisional App. No. 61/165789. The term “game appliance distribution server” in the present application is equivalent to the term “software-image streaming system” in U.S. Provisional App. No. 61/165789. The term “game appliance” in the present application is equivalent to the term “software-image” in U.S. Provisional App. No. 61/165,789.

[0016] FIG. 1 is a block diagram showing, in one exemplary embodiment, components of game appliance system (3000) and a game appliance distribution system.

[0017] A game appliance distribution system (1000) comprises, at least one transceiver (1300), and at least one control processing unit (1100) that can be programmed to deliver pieces of a game appliance (1210) stored in a storage device (1200) (e.g., hard drive, CD, DVD, flash memory, networked file system, etc) to at least one game appliance apparatus (4000) through a broadband network (2000) (e.g., Internet, wide-area network, wired network, wireless network, mobile network, cellular network, etc).

[0018] In an embodiment, a game appliance distribution system (1000) can comprise one or multiple control processing units (1100). A control processing unit can comprise one or multiple processing cores (1110).

[0019] A game appliance (1210) comprises a package of at least one game software application and/or game software appliance. Examples of game appliance include but not limited to, ISO CD/DVD image containing one or multiple game applications, a virtual file system package containing one or multiple game applications, a virtual disk volume image containing one or multiple game applications, a virtual machine image containing one or multiple game applications, etc.

[0020] In a further embodiment, a game appliance can comprise a bundle of operating system together with one or multiple game applications. For example, in an embodiment, a game appliance can comprise, one or multiple game applications and an operating system (e.g., operating system, runtime time environment, just enough operating system, etc) required for running the game applications.

[0021] In an additional embodiment, a game appliance can also comprise emulation environment required by a game application (e.g., console emulation environment, handset emulation environment, portal gaming device emulation environment, etc). For example, in an embodiment, a game appliance can comprise, an operating system, an emulation environment, and one or multiple game applications.

[0022] In an embodiment, the kind of game application contained in a game appliance includes but not limited to, computer game, mobile game, console game, online game client, virtual world client, video game, etc.

[0023] A game appliance can be split into multiple pieces. Depending on embodiment, each piece can have equal size or different size.

[0024] Alternatively, each piece of a game appliance can be compressed using any data compression approach (e.g., zip, tar, bzip, lzw, LZ0, LZX, LZMA, prediction by partial matching, dynamic Markov compression, entropy encoding, etc).

[0025] In an additional embodiment, each piece of a game appliance can be encrypted using any data encryption approach (e.g., AES, Blowfish, DES, Triple DES, Serpent, Twofish, etc).

[0026] In one embodiment, a game appliance system (3000) comprises one or a collection of game appliance apparatuses (4000) that can retrieve game appliance and execute game appliance for one or a collection of client systems (7000).

[0027] In one embodiment, a game appliance apparatus can comprise one or multiple control processing units (e.g., computer processor) featuring several control processing elements (e.g., processor core) in a single chip or in a single IC (integrated circuit) package, so called many-core processors.

[0028] In an embodiment, a game appliance apparatus can comprise one or multiple transceivers (4200)(e.g., wired net-

work transceiver, wireless network transceiver, metropolitan area network transceiver, mobile network transceiver, etc).

[0029] Furthermore, when a game appliance apparatus comprises multiple transceivers, the multiple transceivers can be of the same type or different type.

[0030] A game appliance apparatus (4000) can start to execute a game appliance before all pieces of the game appliance have been downloaded to the game appliance apparatus and/or all pieces of the game appliance have been cached in the game appliance apparatus's storage device. A game appliance apparatus (4000) can start executing a game application contained in a game appliance at the same time when pieces of the game appliance are streamed from one or multiple game appliance distribution systems to the game appliance apparatus. The pieces of a game appliance can be downloaded to a game appliance apparatus (4000) in on-demand fashion wherein a piece is downloaded to a game appliance apparatus if the piece is required by the game appliance apparatus or predicted to be needed in future by the game appliance apparatus.

[0031] In an embodiment, depending on implementation, a game appliance apparatus (4000) can choose any transport for downloading pieces of a game appliance (e.g., HTTP, SSH, UDP, TCP, peer-to-peer link, FTP, etc). An embodiment can choose one transport or alternatively support multiple transports. When multiple transports are supported, an embodiment can either allow user to customize which transport should be used or automatically choose a transport.

[0032] Either a game appliance apparatus itself, or a game appliance distribution system, or both of them can make predictions on which pieces of a game appliance may be needed in future by a game appliance apparatus. Based on the prediction, a game appliance distribution system can speculatively push pieces of a game appliance to a game appliance apparatus and a game appliance apparatus can predictively fetch pieces of a game appliance.

[0033] In an exemplary embodiment, a game appliance apparatus can execute a game appliance using the following process, download one or several pieces of a game appliance from a game appliance distribution system, and start executing the game appliance. When the game appliance apparatus encounters a missing piece of the game appliance, it will search the missing piece in a local storage device where the local storage device stores previously downloaded pieces of one or multiple game appliances as cache. If the missing piece can be found in the cache, the game appliance apparatus will use it. Otherwise it will download the missing piece from one or multiple game appliance distribution systems.

[0034] In an embodiment, a game appliance distribution system (1000) can comprise a game appliance distribution predictor. A game appliance distribution predictor can predict which pieces of a game appliance will be needed by a game appliance apparatus based on the current and/or historic download requests from the game appliance apparatus that is running the game appliance.

[0035] In an embodiment, a game appliance distribution predictor (4000) can make predictions using the following process, access the current and/or historic download requests of pieces of a game appliance made by a game appliance apparatus and make predictions on which pieces of the game appliance that the game appliance apparatus may need to download in future.

[0036] In an embodiment, a game appliance distribution system (1000) can store any program of game appliance

distribution predictor in a storage device (e.g., random access memory, persistent storage unit, etc).

[0037] In a further embodiment, a game appliance distribution system (1000) can comprise one or multiple encrypters. The encrypter can provide encryption service to encrypt pieces of a game appliance. The encrypter can use any encryption approach (e.g., AES, Blowfish, DES, Triple DES, Serpent, Twofish, etc).

[0038] In one embodiment, a game appliance distribution system can encrypt pieces of a game appliance for a game appliance apparatus before those pieces are transmitted to a game appliance apparatus.

[0039] In an alternative embodiment, a game appliance distribution system can encrypt pieces of a game appliance after those pieces of the game appliance are uploaded or encrypt pieces of the game appliance in off-line batch processing.

[0040] In an embodiment, a game appliance distribution system (1000) can store any program of game appliance encrypter in a storage device (e.g., random access memory, persistent storage unit, etc).

[0041] In an additional embodiment, a game appliance distribution system (1000) can store pieces of a game appliance in a content distribution network. The content distribution network can act as a proxy or cache so that a game appliance apparatus can download pieces of a game appliance from the content distribution network.

[0042] Furthermore, a content distribution network can comprise multiple servers distributed over many geographical places. A game appliance distribution system can distribute pieces of a game appliance to all or a subset of those distributed servers of a content distribution network. A game appliance apparatus can fetch pieces of the game appliance from the content distribution network's servers.

[0043] In accordance with the present invention, in one embodiment, a game appliance can use virtualization (e.g., platform emulation, machine emulator such as console emulator, virtual machine, hypervisor, operating system emulation, etc) to execute a game appliance. A game appliance may be developed for hardware or operating system different from the hardware or system of a game appliance apparatus (4000). A game appliance apparatus can use any form of virtualization (e.g., operating system simulation, operating system emulation, hardware assisted virtualization, etc) in order to execute a game appliance originally developed for a different operating system.

[0044] One way to execute a game appliance is to provide a compatible software implementation of a target operating system's kernel services. For example, using OS emulation or simulation, Windows game can be executed on Linux system. Example of such OS emulation includes but not limited to WineX and other similar operating system simulation or emulation approach.

[0045] Furthermore, when emulation is used, the host operating system running on a game appliance apparatus can create emulation tasks that emulate a target platform or a target operating system along with the game appliance running over the target platform or the target operating system.

[0046] Another way to execute one or multiple game appliances by a game appliance apparatus is to run multiple emulation tasks with each task emulating a target platform (e.g., console, a computer platform, an operating system, etc). For example, in one embodiment, a game appliance apparatus (4000) can run multiple emulators (e.g., DOS emulator,

Gameboy emulator, Smartphone emulator, Gamecube emulator, Playstation emulator, Xbox emulator, x86 emulator, Arcade machine emulator, Virtual Box virtual machine guest, Vmware guest, etc).

[0047] A third way to execute a game appliance by a game appliance apparatus is to run game appliance using hypervisor based virtualization. Example of such virtualization includes but not limited to, Xen virtual machine, HyperV virtual machine, Virtual Iron, etc.

[0048] In addition, a game appliance system (3000) can comprise one or a collection of graphics rendering apparatuses (5000) that can render graphics frames for a game appliance executed by a game appliance apparatus (4000). A game appliance apparatus can capture the graphics rendering command (e.g., graphics processing API(application programming interface) call, graphics rendering command, etc) alone with the graphics data of a game appliance executed by the game appliance apparatus. The game appliance apparatus (4000) can send the graphics rendering command including the associated graphics data to a graphics rendering apparatus (5000) that performs graphics processing. A graphics rendering apparatus (5000) can comprise, one or multiple transceivers (5300) for receiving graphics rendering command and graphics data sent from one or multiple game appliance apparatuses; and multiple graphics accelerators for conducting graphics processing and frame rendering.

[0049] In a further embodiment, a graphics rendering apparatus (5000) can compress rendered frames into media stream, and transmit the media stream over a local network (e.g., local area network, local wireless network, metropolitan area network, or mobile network, etc) (6000) to one or a collection of client systems (7000) using a media streamer (5420).

[0050] In one embodiment, the local network (6000) connecting a game appliance system (3000) and a client system (7000) is a local wired network (e.g., Ethernet(e.g., Fast Ethernet, Gigabit Ethernet, multi-Gigabit Ethernet, 100-Gigabit Ethernet), etc)).

[0051] In a further embodiment, the local wired network (6000) can also comprise, one or multiple bridges, routers, switches, or network repeaters.

[0052] In another embodiment, the local network (6000) connecting a game appliance system (3000) and a client system is a local wireless network (e.g., all versions of WiFi(e.g., 802.11a, 802.11b, 802.11g, 802.11n, 802.11i, 802.11s, 802.11aa, 802.11ac, 802.11ad, WAPI, or other future WiFi standard), WiMax, etc). Furthermore, when an embodiment uses WiFi, one or multiple WiFi repeaters can be used to extend the range of a game appliance system.

[0053] In another embodiment, the local network (6000) connecting a game appliance system (3000) and a client system is a metropolitan area network (e.g., metro Ethernet, DQDB, ATM, HiperMAN, iburst, etc).

[0054] In another embodiment, the local network (6000) connecting a game appliance system and a client system is a mobile network (e.g., 4G, 3GPP Long Term Evolution, E-UTRA, etc).

[0055] In an embodiment, game appliance systems (3000) can be deployed at hotspot (e.g., train/bus stations, airports, military bases, restaurants, libraries, coffee shops, bookstores, hotels, hospitals, Internet-cafe, department stores, supermarkets, parks, campgrounds, student dormitories and

other similar places) and client system can connect to the game appliance system via a local network provided by the hotspot.

[0056] In an embodiment, the compression processing used by a game appliance system can comprise any video (e.g., WMV, H263, H264, H265, or any existing or future video encoding techniques), or image compression approach (e.g., JPEG, lossless JPEG, JPEG2000, PNG, TIFF, GIF, or any other standard or customized image compression approach).

[0057] In an embodiment, a game appliance apparatus and a graphics rendering apparatus can communicate with each other via transceiver. The connection that connects a graphics rendering apparatus with a game appliance apparatus can include but not limited to, Gigabit Ethernet, multi-Gigabit Ethernet, 100-Gigabit Ethernet, PCI Express extension, advanced switching fabric built over PCI Express, Infiniband, Scalable Coherent Interface, or other system-area-network or local area network mechanism.

[0058] In an additional embodiment, an inter-connect mechanism can connect multiple game appliance apparatuses with one or multiple graphics rendering apparatuses.

[0059] In a further embodiment, a graphics rendering apparatus (**5000**) can be shared by multiple game appliance apparatuses (**4000**). A game appliance apparatus can also access multiple graphics rendering apparatuses.

[0060] In an additional embodiment, because the rendered frames are not displayed locally by a graphics rendering apparatus or game appliance apparatus, for each game appliance, its graphics can be rendered on some “virtual frame buffer” (e.g., emulated display device, virtual frame buffer, non-display buffer, or “memory surface”, etc). When a graphics rendering apparatus processes graphics rendering commands from multiple game appliance apparatuses, it can create different virtual frame buffer for different game appliance.

[0061] In an embodiment, a client system (e.g., computer, mobile device such as PDA or cellphone or handset, netbook, laptop computer, etc) supports rendering of the media streams transmitted to it from a game appliance apparatus or a graphics rendering apparatus over a local network. It can decompress the received media stream and display the frames on an attached display device (e.g., television, monitor, LCD panel, TFT-LCD, ECD, etc).

[0062] In an embodiment, for interactive game play or control, a client system can comprise one or multiple input devices such as game pad, video game controller, touchscreen, buttons, keypad, or other similar game input device. Optionally, a client system itself (e.g., cellphone, PDA) can be used as an input device. An input device can connect to a client system via wired or wireless connection.

[0063] FIG. 2 is a block diagram showing, in one exemplary embodiment, components of a client system (**7000**) (e.g., computer, portable computer, netbook, laptop computer, mobile device (e.g., PDA, handset, portal device), etc).

[0064] A client system (**7000**) supports rendering of the video streams transmitted to it from a game appliance system (**3000**) over a local network (e.g., local area network, local wireless network, metropolitan area network, or mobile network, etc). It can decompress the received media stream using a video decoder (**7400**), display the media stream on an attached display device (**7100**) (e.g., monitor, LCD panel, TFT-LCD, ECD, etc).

[0065] In an embodiment, a client system (**7000**) can implement video decoder (**7400**) as a piece of software executed by at least one control processing unit. An embodiment can

choose any programming environment for implementing video decoder. Furthermore, a video decoder can be a stand-alone application, or a browser plugin, or a dynamically loadable library, etc.

[0066] In addition, in an embodiment, the decoded media stream can be displayed either in fullscreen display mode, or in windows display mode, or as an overlay display, or as embedded display where the media stream is embedded into another application (e.g., a browser).

[0067] For interactive game play, a client system can comprise one or multiple input devices (e.g., game pad, video game controller, touchscreen, keypad, or other similar game input device, etc) (**7500, 7600**). An input device (**7500, 7600**) can connect to the client system (**7000**) via wired or wireless connection.

[0068] Furthermore, there can be multiple input devices connected to a client system (**7000**).

[0069] In an embodiment, a client system (**7000**) can comprise at least one input unit (**7200**) where said input unit receives input event from one or multiple input devices connecting with the client system and sends the received input to a game appliance apparatus.

[0070] In an embodiment, a game appliance apparatus can use device emulation to emulate one or multiple input devices (e.g., game controller, keyboard, game pad, pointing device, etc). A game appliance executed by a game appliance apparatus can receive input data from the emulated input device. A game appliance apparatus can use the input data sent from a client system to drive the emulated input device.

[0071] For example, in one embodiment, buttons of a handset can be mapped to the buttons of a game controller. Then interactive input of the handset can be translated into game controller input.

[0072] In another example, in one embodiment, touchscreen of a handset can be mapped to a pointing device (e.g., mouse). Then input of the handset can be translated into pointing device input.

[0073] In addition, because a game appliance apparatus emulates input devices, the emulated devices can be bound with input data from any client system.

[0074] In one exemplary embodiment, inputs received by a client system can be transmitted as one input event or multiple input events depending on the interval between them. Said input can be generated by any input device (e.g., keyboard, keypad, game controller, projector controller, joypad, joystick, touchscreen, mouse, accelerometer, etc).

[0075] In one embodiment, when one input (e.g., key/button release, mouse button release, touchscreen untouch, etc) occurs quickly after another input (e.g., key/button press, mouse button press, touchscreen touch, etc) with interval less than a threshold, a client system can send a message that comprises both inputs to a game appliance apparatus.

[0076] In an additional embodiment, a client system may first translate an input event to a different input event that can be processed by a destination game appliance apparatus, then transmit the translated input event to the game appliance apparatus.

[0077] On an alternative embodiment, a client system can transmit the original input event. Then, the game appliance apparatus that receives the event does any required translation.

[0078] In an embodiment, upon receipt of an input event by a game appliance apparatus, it can append the input event to a game appliance execution unit after optional input event

translation. There can be one or multiple input event queues (e.g., fifo, queue, stack, ring, etc) associated with each instance of game appliance executed by a game appliance apparatus. A game appliance apparatus appends the received input events to the corresponding queue of the targeted game appliance instance.

[0079] On a client system, after first input (e.g., key/button press, mouse button press, touchscreen touch, etc) is received, if the client system does not receive the second input (e.g., key/button release, mouse button release, touchscreen untouch, etc) within a pre-determined time window, the client system will treat them as two separate events. In one case, a client system can send the first input event (e.g., key/button press, mouse button press, touchscreen touch, etc), followed by the second input (e.g., key/button release, mouse button release, touchscreen untouch, etc) after it is captured by the input unit.

[0080] In an embodiment, input translation can be performed either by a client system or a game appliance apparatus.

[0081] In an embodiment, a client system (7000) can comprise one or multiple transceivers (7300) for receiving media stream and/or transmitting input.

[0082] In one embodiment, a transceiver of a client system (7300) connects said client system to a local wired network (e.g., Ethernet(e.g., Fast Ethernet, Gigabit Ethernet, multi-Gigabit Ethernet, 100-Gigabit Ethernet), etc)).

[0083] In another embodiment, a transceiver of a client system (7300) connects said client system to a local wireless network (e.g., all versions of WiFi(e.g., 802.11a, 802.11b, 802.11g, 802.11n, 802.11i, 802.11s, 802.11aa, 802.11ac, 802.11ad, WAPI, or other future WiFi standard), WiMax, etc).

[0084] In another embodiment, a transceiver of a client system (7300) connects said client system to a metropolitan area network (e.g., metro Ethernet, DQDB, ATM, HiperMAN, iburst, etc).

[0085] In another embodiment, a transceiver of a client system (7300) connects said client system to a mobile network (e.g., 4G, 3GPP Long Term Evolution, E-UTRA, etc).

[0086] FIG. 3 is a block diagram showing, in one exemplary embodiment, components of a game appliance apparatus.

[0087] A game appliance apparatus (4000) can offer game appliance execution service for at least one client system wherein a game appliance apparatus can start a game appliance for a client system and said client system can send input to the game appliance apparatus via network to control game play of the game appliance and in return receive media stream of the game appliance via a local network.

[0088] In an embodiment, a game appliance apparatus (4000) can comprise, one or multiple transceivers (4200)(e.g., wired network transceiver, wireless network transceiver, metropolitan area network transceiver, mobile network transceiver, etc). When a game appliance apparatus comprises multiple transceivers, the multiple transceivers can be of the same type or different type.

[0089] In one embodiment, a transceiver (4200) of a game appliance apparatus (4000) is a wired network transceiver (e.g., Ethernet transceiver (e.g., Fast Ethernet transceiver, Gigabit Ethernet transceiver, multi-Gigabit Ethernet transceiver, 100-Gigabit Ethernet transceiver), DSL modem transceiver, cable modem transceiver, Infiniband transceiver, PCI Express transceiver, etc).

[0090] In another embodiment, a transceiver (4200) of a game appliance apparatus (4000) is a wireless network transceiver (e.g., WiFi transceiver(e.g., 802.11a, 802.11b, 802.11g, 802.11n, 802.11i, 802.11s, 802.11aa, 802.11ac, 802.11ad, WAPI, or other future WiFi standard), WiMax transceiver, etc).

[0091] In another embodiment, a transceiver (4200) of a game appliance apparatus (4000) is a metropolitan area network transceiver (e.g., metro Ethernet transceiver, DQDB transceiver, ATM transceiver, HiperMAN transceiver, iburst transceiver, etc).

[0092] In another embodiment, a transceiver (4200) of a game appliance apparatus (4000) is a mobile network transceiver (e.g., 4G transceiver, 3GPP Long Term Evolution transceiver, E-UTRA transceiver, etc).

[0093] In an embodiment, a game appliance apparatus (4000) can use the same transceiver for connecting to both game appliance distribution system and graphics rendering apparatus.

[0094] In another embodiment, a game appliance apparatus (4000) can use different transceivers for connecting to game appliance distribution system and graphics rendering apparatus.

[0095] In an exemplary embodiment where a game appliance apparatus (4000) uses different transceivers for connecting to game appliance distribution system and graphics rendering apparatus, the game appliance apparatus can use one or multiple wired network transceivers (e.g., Ethernet transceiver (e.g., Fast Ethernet transceiver, Gigabit Ethernet transceiver, multi-Gigabit Ethernet transceiver, 100-Gigabit Ethernet transceiver), Infiniband transceiver, PCI Express transceiver, etc) for connecting with one or multiple graphics rendering apparatuses and uses another transceiver or another set of transceivers (e.g., wired network transceiver, wireless network transceiver, metropolitan area network transceiver, mobile network transceiver, etc) for connecting with one or multiple game appliance distribution systems.

[0096] Furthermore, a game appliance apparatus (4000) can comprise one or multiple many-core control processing units (4100). A control processing unit can further comprise multiple control processing elements (e.g., processor core of a many-core control processing unit, etc). A game appliance can assign different workload to different control processing elements.

[0097] In an embodiment, a game appliance apparatus (4000) can program one control processing element to, on-demand retrieve a game appliance from one or multiple game appliance distribution systems or retrieve a game appliance from a local game appliance cache. The part of a game appliance apparatus that completes said functionality is called, game appliance fetching agent (4120).

[0098] In an embodiment, a game appliance apparatus (4000) can execute or can be programmed to execute a game appliance in on-demand fashion before all pieces of the game appliance have been downloaded to the game appliance apparatus and/or all pieces of the game appliance have been cached by the game appliance apparatus.

[0099] In a further embodiment, a game appliance can allocate and program a plurality of control processing elements as game appliance execution units (4110). A game appliance execution unit (4110) can execute a game appliance using emulation or virtualization (e.g., platform emulation, machine emulator such as console emulator, virtual machine, hypervisor, operating system emulation, hardware assisted

virtualization, etc), receive input from one or multiple client systems, capture graphics rendering command of the executed game appliance using virtual graphics driver, and transmit the captured graphics rendering command to one or multiple graphics rendering agents.

[0100] In an embodiment of the present invention, multiple game appliances can be executed concurrently by multiple game appliance execution units of a game appliance apparatus using emulation or virtualization (e.g., platform emulation, machine emulator such as console emulator, virtual machine, hypervisor, operating system emulation, etc).

[0101] When a game appliance apparatus uses emulation to emulate a console or a computer platform or an operating system (e.g., VMware, Virtual box, Parallel Workstation, Bochs, QEMU, etc), a game appliance execution unit comprises the emulation process or emulation task.

[0102] When a game appliance apparatus uses virtualization (e.g., Xen, HyperV, Virtual Iron, KVM, hardware assisted virtualization, etc), a game appliance execution unit comprises the virtual machine guest. For example, when Xen virtualization is used by a game appliance apparatus, a game appliance execution unit comprises DomU of a virtualization guest.

[0103] In an additional embodiment, a game appliance apparatus can also use existing or future hardware-assisted virtualization (e.g., Intel VT, AMD-V).

[0104] Furthermore, a game appliance can execute multiple game appliances using multiple game appliance execution units concurrently and also perform graphics rendering for the multiple game appliances concurrently. This allows a game appliance system (3000) to respond to multiple client systems.

[0105] The actual graphics rendering can be carried out by a game appliance apparatus or a graphics rendering apparatus (5000).

[0106] In an embodiment, a game appliance apparatus (4000) can comprise at least one persistent storage device (4300) wherein said persistent storage device can store one or multiple downloaded pieces of a game appliance that can be used as game appliance cache.

[0107] In a further embodiment, when game appliance is encrypted, a game appliance apparatus (4000) can comprise at least one game appliance decrypter. The decrypter can decrypt pieces of a game appliance received by the game appliance apparatus. The decrypter can use any decryption approach compatible with the encryption used by a game appliance distribution system (e.g., AES, Blowfish, DES, Triple DES, Serpent, Twofish, etc).

[0108] In an embodiment, a game appliance apparatus can store any program of game appliance decrypter in a storage device (e.g., random access memory, persistent storage unit, etc).

[0109] In an additional embodiment, one or multiple decryption keys can be exchanged between a game appliance apparatus (4000) and a game appliance distribution system.

[0110] In a further embodiment, a game appliance distribution system can authenticate a game appliance apparatus before providing service to it using any authentication approach. Furthermore, a game appliance apparatus can also authenticate a game appliance distribution system using any authentication approach. An embodiment can alternatively use mutual authentication to ensure the identity of a game appliance distribution system or game appliance apparatus.

[0111] FIG. 4 is a flowchart showing, in one exemplary embodiment, process used by a game appliance apparatus to launch a game appliance.

[0112] In one embodiment, a game appliance apparatus can mount a game appliance (4116) as a virtual file system or a virtual disk volume. It can execute software application contained in a game appliance before all pieces of the game appliance are downloaded.

[0113] In an embodiment, a game appliance apparatus can execute a game appliance using emulation or virtualization (4118) (e.g., virtual machine, emulator, application virtualization, etc). The kind of emulation or virtualization includes but not limited to, full machine emulation, OS stack emulation, virtualized operating system, partially virtualized operating system services, application virtualization, hardware assisted virtualization, etc. Examples of virtualization solutions that can be used by a game appliance apparatus include but not limited to, Microsoft Hyper-V, virtual PC, virtual box, VMWare, VMWare thinapp, Xen, KVM, QEMU, BoxedApp, InstallFreen, Xencode, App-V, Virtualbox, Virtual Iron, etc. It should be kept in mind, the above list is for illustration purpose only. It is not intended to limit the scope of the present invention to only those virtualization solutions.

[0114] In an embodiment, a game appliance apparatus can execute a game appliance and download pieces of the game appliance concurrently. When a game appliance apparatus encounters a missing piece of the game appliance, it will search the missing piece in a local storage device where the local storage device stores previously downloaded pieces of one or multiple game appliances as cache. If the missing piece can be found in the cache, the game appliance apparatus will use it. Otherwise it will download the missing piece from one or multiple game appliance distribution systems (4119).

[0115] In an embodiment, a game appliance apparatus can mount a game appliance as a virtual file system or a virtual disk volume (4116). In one embodiment, the mounted file system or disk volume may be in read-only mode. In this case, the game appliance contained in the software-image may use union file system (e.g., ram file system, union file system, AuFS, or other union file system or disk union system) or stacked file system to prevent writing to the mounted file system or disk volume.

[0116] In an alternative embodiment, a game appliance apparatus can mount a game appliance as a virtual file system or a virtual disk volume in such a way that all writes are kept local.

[0117] In an additional embodiment, local changes to a game appliance can be separated from the original downloaded pieces of a game appliance.

[0118] In an embodiment, a game appliance apparatus (4000) can comprise a storage device (e.g., hard drive, CD, DVD, flash memory, etc) which stores programs required for running a game appliance.

[0119] FIG. 5 is a block diagram showing, in one exemplary embodiment, components of a game appliance execution unit and additional components of a game appliance apparatus (4000).

[0120] In an embodiment of the present invention, a game appliance (4112) can be executed by a game appliance execution unit (4110) of a game appliance apparatus (4000) using emulation or virtualization (e.g., platform emulation, machine emulator such as console emulator, virtual machine, hypervisor, operating system emulation, etc). A game appliance execution unit (4110) can capture graphics rendering

commands and graphics data of an executed game appliance. For example, two most popular graphics APIs used for developing video games are OpenGL and Direct3D. Using virtual graphics driver (4114), a game appliance execution unit can capture the graphics processing calls made by a game appliance (4112) and the associated graphics data sent by the game appliance (4112) for graphics rendering.

[0121] In an embodiment of the present invention, a virtual graphics driver (4114) can be used by a game appliance execution unit (4110) to intercept graphics processing commands. In this embodiment, a game appliance can register a virtual graphics driver. A game application contained in a game appliance (4112) can use the virtual graphics driver for graphics processing. Said virtual graphics driver (4114) can capture the graphics commands made by the game application and send them to a graphics rendering agent (4130) executed by either the game appliance apparatus or a graphics rendering apparatus.

[0122] After graphics commands or graphics data are captured by a virtual graphics driver (4114), they are sent to one or multiple graphics rendering agents via a virtual graphics bus (4116). A virtual graphics bus (4116) comprises at least one communication channel (e.g., network communication channel, local socket communication, shared memory, memory mapped IO, pipe, etc). A virtual graphics driver sends graphics rendering command and/or graphics data to a graphics rendering agent residing either on the same game appliance apparatus or on a graphics rendering apparatus.

[0123] In one embodiment, a virtual graphics bus (4116) comprises one or multiple network communication channels (e.g., TCP, IP, UDP, Ethernet communication channel, Infiniband communication channel, etc).

[0124] In another embodiment, a virtual graphics bus (4116) comprises one or multiple socket communication channels.

[0125] In another embodiment, a virtual graphics bus (4116) comprises one or multiple shared memories.

[0126] In another embodiment, a virtual graphics bus (4116) comprises memory mapped IO.

[0127] In another embodiment, a virtual graphics bus (4116) comprises one or multiple pipes.

[0128] In an embodiment, a game appliance apparatus can further comprise, one or multiple graphics accelerators (5100). A graphics accelerator can be a discrete device or an integrated device. For example, a graphics accelerator can connect to a game appliance apparatus using PCI Express bus or PCI Express extension connection, or other inter-connect or bus mechanism.

[0129] Furthermore, a game appliance apparatus can comprise, one or multiple graphics rendering agents (4130) wherein said graphics rendering agent is executed by at least one control processing element and the graphics rendering agent schedules graphics rendering commands to one or multiple graphics accelerators. In this embodiment, a game appliance execution unit can transmit graphics command and the associated graphics data via a virtual graphics bus (e.g., network communication channel, local socket communication, shared memory, memory mapped IO, pipe, etc).

[0130] In an additional embodiment, a game appliance apparatus can comprise one or multiple frame compressors wherein said frame compressor comprises frame compressing program executed by one or multiple graphics accelerators and said frame compressor can compress rendered frame into compressed frame or video.

[0131] In an alternative embodiment, a graphics rendering agent can be executed by a stand-alone graphics rendering apparatus. A game appliance system can comprise, one or multiple graphics rendering apparatuses that offer graphics rendering for one or multiple game appliance apparatuses.

[0132] Furthermore, when stand-alone graphics rendering apparatus is used, a game appliance apparatus can connect with one or multiple graphics rendering apparatuses via a local area network (e.g., Ethernet (e.g, Fast Ethernet, Gigabit Ethernet, multi-Gigabit Ethernet, 100-Gigabit Ethernet, etc), wireless LAN, etc), system-area-network (PCI Express extension, advanced switching fabric built over PCI Express, Infiniband, peer-to-peer direct connection, etc) or other similar network communication mechanism using one or multiple transceivers. A game appliance apparatus can comprise one or multiple transceivers for transmitting graphics commands and the associated graphics data to one or multiple graphics rendering apparatuses.

[0133] FIG. 6 is a block diagram showing, in one exemplary embodiment, components of a graphics rendering apparatus. A graphics rendering apparatus (5000) can comprise, one or multiple graphics rendering agents (5410), and one or multiple graphics accelerators (5100), and system memory (5500). A graphics accelerator (5100) can further comprise, one or multiple graphics processing units and graphics memory.

[0134] In one embodiment, a graphics rendering agent (5410) receives graphics rendering commands and the associated data (e.g., texture maps, geometry data, 3D mesh, vertex attributes, shader programs, etc).

[0135] In a further embodiment, a graphics rendering agent can create a graphics rendering context for a game appliance and store the corresponding graphics rendering commands and graphics data as context of the game appliance. A graphics rendering agent can also perform graphics rendering by scheduling graphics rendering commands to one or multiple graphics accelerators.

[0136] In an embodiment, a graphics rendering apparatus can comprise one or multiple control processing units (5400). Furthermore, a control processing unit can comprise one or multiple control processing elements. A graphics rendering agent (5410) can be executed by one or multiple control processing elements of a graphics rendering apparatus.

[0137] In an additional embodiment, graphics data can be compressed and then transmitted to a graphics rendering apparatus for reducing bandwidth demand. Examples of compressible graphics data include but not limited to, textures, or 3D geometry attributes, or vertex attributes, or vertex buffer data, or vertex arrays, or 3D mesh, or other similar graphics data.

[0138] When graphics commands or graphics data are compressed, a graphics rendering agent can decompress the compressed graphics command and data before scheduling them to a graphics accelerator.

[0139] In an additional embodiment, programs capable of generating or synthesizing graphics data can be transmitted to a graphics rendering apparatus in replacing of the graphics data themselves. For example, many games use some sorts of particle simulation to simulate natural phenomena or to achieve many visual effects. Instead of sending large amount of particle data, a game appliance can transmit or upload program of particle simulation to a graphics rendering apparatus. During graphics rendering, the program can generate or simulate particle data based on certain control parameters.

Those control parameters can be sent by a game appliance in realtime without consuming large amount of bandwidth.

[0140] In another example, a game appliance apparatus can send program that can synthesize or generate textures to a graphics rendering apparatus instead of transmitting the textures themselves. Similarly, 3D mesh can be also generated or synthesized using programs. Those mesh generating or synthesizing or tessellation programs can be transmitted to a graphics rendering apparatus for producing mesh data instead of transmitting the mesh data themselves.

[0141] Furthermore, mesh generating or synthesizing programs can export control parameters that can be used for controlling mesh generation. A game appliance can at runtime send different configurations of control parameters to generate the kind of mesh that it wants without transmitting large number of vertices or geometry data.

[0142] In addition, vertex attributes such as texture coordinates can also be generated or synthesized using programs executed by a graphics rendering apparatus.

[0143] A graphics rendering agent (**5410**) of a graphics rendering apparatus can schedule execution of graphics commands of a graphics rendering context or send graphics commands of a graphics rendering context to a graphics accelerator (**5100**) of the graphics rendering apparatus.

[0144] Furthermore, a graphics accelerator (**5100**) of a graphics rendering apparatus can render graphics frames for multiple graphics rendering contexts.

[0145] In one embodiment, a graphics accelerator (**5100**) can be time-shared by a collection of graphics rendering contexts.

[0146] In one embodiment, for a graphics rendering context, a graphics rendering agent (**5410**) can reserve a processing time window or a set of graphics command queue entries of one or multiple graphics accelerators (**5100**). In one embodiment, a graphics accelerator (**5100**) can read graphics commands from an input command queue.

[0147] In another embodiment, when a graphics accelerator (**5100**) is able to process graphics commands from multiple graphics rendering contexts concurrently at the same clock cycle or is able to interleave graphics commands from multiple graphics rendering contexts at either clock cycle level or graphics command level, then a graphics rendering apparatus (**5000**) can schedule multiple graphics rendering contexts to a graphics accelerator (**5100**).

[0148] In this case, the number of graphics rendering contexts that can be scheduled to a graphics accelerator (**5100**) depends on how many different graphics rendering contexts the graphics accelerator (**5100**) can process either concurrently or at cycle interleaving level.

[0149] Furthermore, a graphics rendering apparatus (**5000**) can comprise a plurality of graphics accelerators (**5100**). This allows a graphics rendering apparatus (**5000**) to schedule graphics commands from multiple game appliances to different graphics accelerators. In addition, a graphics rendering apparatus (**5000**) can dynamically bind graphics accelerator (**5100**) with graphics rendering contexts. After a graphics context is bound with a graphics accelerator, a graphics rendering agent (**5410**) can route all the received graphics rendering commands of the graphics rendering context to the associated graphics accelerator.

[0150] When a graphics rendering apparatus (**5000**) receives graphics command of a graphics rendering context, it may not send them directly to a graphics accelerator. Instead, in an embodiment, a graphics rendering apparatus (**5000**) can

append the received graphics command of a graphics rendering context to one or multiple virtual graphics command queues associated with the graphics rendering context. Said virtual graphics command queue or queues of a graphics rendering context store the graphics commands of the graphics rendering context. When a graphics rendering apparatus (**5000**) assigns a graphics rendering context to one or multiple graphics accelerators (**5100**), the stored graphics commands of said graphics rendering context can be retrieved and sent to the graphics accelerator.

[0151] In one embodiment, when a graphics rendering apparatus (**5000**) schedules graphics commands, the graphics rendering apparatus (**5000**) selects a graphics rendering context that has pending graphics commands out of a collection of graphics rendering contexts. Then it selects one or multiple graphics accelerators (**5100**) of the graphics rendering apparatus for processing the pending graphics commands of the selected graphics rendering context. After that it issues context switch request to the selected graphics accelerator. Then, it sends pending graphics commands of the selected graphics rendering context to the graphics accelerator.

[0152] In one embodiment, a graphics rendering apparatus (**5000**) can add graphics commands of the selected graphics rendering contexts to the command queue or commands queues of the selected graphics accelerator.

[0153] Still referring to FIG. 6, in an embodiment, graphics commands sent by a graphics rendering apparatus (**5000**) to a graphics accelerator can be translated or converted graphics commands from the original graphics commands received by the graphics rendering apparatus.

[0154] In an embodiment, a graphics accelerator of a graphics rendering apparatus can accept or handle graphics command in a format different from the format of graphics command received by the graphics rendering apparatus from a game appliance apparatus. For example, in an embodiment, a graphics rendering apparatus can receive graphics API calls from a game appliance apparatus. Then the graphics API calls are converted or transformed into graphics commands that can be processed by a graphics accelerator.

[0155] In a further embodiment, graphics command and/or graphics data can be translated from one API to another API. For instance, in an embodiment, DirectX based graphics rendering commands can be translated to OpenGL rendering commands. Similarly, OpenGL rendering commands can be translated to DirectX based graphics rendering commands.

[0156] Moreover, graphics rendering commands and/or graphics data of different API versions can also be translated. For instance, in an embodiment, graphics rendering commands of one version of DirectX or OpenGL API can be translated to graphics rendering commands of a different version of DirectX or OpenGL API.

[0157] In another exemplary embodiment, graphics rendering commands and/or graphics data of OpenGL ES can be translated to OpenGL rendering commands.

[0158] Furthermore, during API translation, shaders and/or any program designated to be executed by a graphics rendering unit can also be translated from one API to shaders of another API. For example, shaders of DirectX can be translated to OpenGL shaders.

[0159] Still referring to FIG. 6, there can be multiple embodiments of how a graphics rendering apparatus (**5000**) selects the next graphics rendering context when it schedules graphics commands to a graphics accelerator (**5100**). In one embodiment, a graphics rendering apparatus can select the

next graphics rendering context out of a collection of graphics rendering contexts in round-robin manner.

[0160] In another embodiment, a graphics rendering apparatus (5000) can select the next graphics rendering context out of a collection of graphics rendering contexts based on pre-determined frame rate constraints or frame rendering deadlines. For example, in an embodiment, a graphics rendering context can have a quality of service requirement that specifies a desired rendering frame rate or frame rendering period.

[0161] Furthermore, a graphics rendering apparatus (5000) can record or keep track of the number of frames that have already been rendered for each graphics rendering context within a time window. Then the disparity or difference between the desired and the actual frame rate or frame count can be calculated. Then a graphics rendering apparatus can assign priority to graphics rendering contexts based on the extent how the desired and the actual frame rate differ. For example, a higher priority can be assigned to a graphics rendering context whose real frame rate is significantly behind a pre-determined desired frame rate.

[0162] In one embodiment, a graphics rendering apparatus (5000) can schedule graphics rendering contexts using any real time scheduling approaches such as rate monotonic scheduling, or fair queueing, or earliest deadline first scheduling, or dynamic priority scheduling, or other similar scheduling approach.

[0163] In an embodiment, a graphics rendering apparatus (5000) can further comprise, one or multiple transcoders (5430) executed by one or multiple control processing elements. Said transcoders (5430) can transcode rendered frames according to the condition of a destination client system (e.g., screen resolution, available bandwidth, required frame rate, etc). For example, frames can be transcoded to fit with the screen size of a destination client system. Furthermore, transcoding can also be performed according to achievable or pre-determined bitrate. In an embodiment, bitrate can be determined based on the available or assigned bandwidth between a client system and a game appliance system.

[0164] Furthermore, a graphics rendering apparatus (5000) can comprise, one or multiple media streamers (5420) wherein said media streamer can deliver media stream of a game appliance to one or multiple client systems via a local network (e.g., local area network, local wireless network, metropolitan area network, or mobile network, etc). An embodiment can use any existing, customized, or future media streaming approach (e.g., RTP, RTSP, MMS, MPEG-TS, RTMP, etc).

[0165] In an embodiment, a graphics rendering apparatus (5000) can comprise one or multiple networking interfaces. Said networking interface can comprise, one or multiple transceivers (5300)(e.g., wired network transceiver, wireless network transceiver, metropolitan area network transceiver, mobile network transceiver, etc).

[0166] In one embodiment, a transceiver (5300) of a graphics rendering apparatus is a wired network transceiver (e.g., Ethernet transceiver (e.g. Fast Ethernet transceiver, Gigabit Ethernet transceiver, multi-Gigabit Ethernet transceiver, 100-Gigabit Ethernet transceiver), DSL modem transceiver, cable modem transceiver, Infiniband transceiver, PCI Express transceiver, etc).

[0167] In another embodiment, a transceiver (5300) of a graphics rendering apparatus is a wireless network transceiver (e.g., WiFi transceiver(e.g., 802.11a, 802.11b, 802.

11g, 802.11n, 802.11i, 802.11s, 802.11aa, 802.11ac, 802.11ad, WAPI, or other future WiFi standard), WiMax transceiver, etc).

[0168] In another embodiment, a transceiver (5300) of a graphics rendering apparatus is a metropolitan area network transceiver (e.g., metro Ethernet transceiver, DQDB transceiver, ATM transceiver, HiperMAN transceiver, iburst transceiver, etc).

[0169] In another embodiment, a transceiver (5300) of a graphics rendering apparatus is a mobile network transceiver (e.g., 4G transceiver, 3GPP Long Term Evolution transceiver, E-UTRA transceiver, etc).

[0170] Furthermore, when a graphics rendering apparatus (5000) comprises multiple transceivers, the multiple transceivers can be of the same type or different type.

[0171] In an embodiment, a graphics rendering apparatus (5000) can use the same transceiver for connecting with both game appliance apparatus and client system.

[0172] In another embodiment, a graphics rendering apparatus (5000) can use different transceivers for connecting with game appliance apparatus and client system.

[0173] In an exemplary embodiment where a graphics rendering apparatus (5000) uses different transceivers for connecting with game appliance apparatus and client system, the graphics rendering apparatus can use one or multiple wired network transceivers (e.g., Ethernet transceiver (e.g. Fast Ethernet transceiver, Gigabit Ethernet transceiver, multi-Gigabit Ethernet transceiver, 100-Gigabit Ethernet transceiver), Infiniband transceiver, PCI Express transceiver, etc) connecting with one or multiple game appliance apparatuses and use another transceiver or another set of transceivers (e.g., wired network transceiver, wireless network transceiver, metropolitan area network transceiver, mobile network transceiver, etc) connecting with one or multiple client systems. For instance, in one exemplary embodiment, a graphics rendering apparatus (5000) can use one or multiple wired transceivers for connecting with one or multiple game appliance apparatuses and use one or multiple wireless network transceivers or mobile network transceivers for connecting with one or multiple client systems.

[0174] In one embodiment, a networking interface can comprise one or multiple programmable network processing engines (e.g., network processing ASIC (application specific integrated circuit), network processor, etc). The networking interface can be either a stand-alone device or an integrated device (e.g., integrated with a graphics accelerator, integrated with a graphics processing unit, etc).

[0175] Furthermore, a networking interface can have its own device memory (e.g., SRAM, RAM, etc) or use the video memory of the graphics accelerator for its operation and data storage.

[0176] In advanced embodiment, a networking interface can comprise additional functionality of data processing.

[0177] For example, the network processing engine of a networking interface can have enough processing power for data compression, data packaging, or data segmentation/packetizing, etc. The additional functionality can be either implemented as fixed logic or software.

[0178] In an embodiment, the network processing engine of a networking interface can first read the rendered frame from either host memory or from a graphics accelerator's video memory, then it can compress the rendered frame, packetize the compressed frame, and transmit the frame out.

[0179] The frame compression or packetizing functionality can be implemented either as hardware logic or software or firmware executed by a networking processing engine of a networking interface.

[0180] FIG. 7 is a flowchart showing, in another exemplary embodiment, process of graphics rendering used by a graphics rendering apparatus, receiving graphics rendering command (5810), carrying out graphics rendering (5812), compressing the rendered frames (5814), transcoding (5816), and streaming the results over a local network to one or multiple client systems(5818).

[0181] The steps involved on a graphics rendering apparatus comprise (the steps can be performed concurrently, or in any sequence or combination), receiving graphics rendering commands and the associated data by a graphics rendering agent (5810), rendering frames by one or multiple graphics accelerators(5812), compressing or encoding the rendered frames (5814), transcoding the frames (5816), streaming (5818) the result packets over a network. The term “transcoding” in the present application includes both video encoding and/or video transcoding.

[0182] Graphics rendering can use any video or image compression approach for compressing rendered frames (e.g., WMV, H264, H263, H265, or any standard, customized or future video or frame compression approach).

[0183] After compression, frames can be further transcoded according to the condition of the destination client system (e.g., screen resolution, available bandwidth, required frame rate, etc) by a graphics rendering apparatus via one or multiple transcoders executed by one or multiple control processing elements. For example, frames can be transcoded to fit with the screen size of a destination client system. Furthermore, transcoding can also be performed according to achievable or pre-determined bitrate. In an embodiment, bitrate can be determined based on the available or assigned bandwidth between a client system and a game appliance system.

[0184] In a further embodiment, a graphics rendering apparatus can comprise, one or multiple video encoding accelerators. A video encoding accelerator can offload whole or part of transcoding process (5816). In one embodiment, a graphics rendering apparatus can use one or multiple add-on cards where said add-on card comprises one or multiple processing elements that can accelerate video transcoding. For instance, a graphics accelerator comprising multiple SIMD(Single Instruction stream, Multiple Data stream) or MIMD(Multiple Instruction stream, Multiple Data stream) engines can also be used as video transcoding accelerator where certain transcoding processing can be offloaded to the graphics accelerator (e.g., motion estimation, transformation, etc).

[0185] Alternatively, a video encoding accelerator can comprise one or multiple FPGA (Field-programmable gate array) devices where certain transcoding operations are offloaded to the FPGA device.

[0186] Furthermore, a video encoding accelerator can comprise, one or multiple DSP (digital signal processing) devices where certain transcoding operations are offloaded to the DSP device.

[0187] After transcoding, media stream of an executed game appliance is delivered to one or multiple client systems via a local network (e.g., local area network, metropolitan area network, local wireless network, or mobile network, etc). An embodiment can use any existing, customized, or future media streaming approach (e.g., RTP, RTSP, MMS, MPEG-TS, RTMP, etc).

[0188] In addition, in an embodiment, a graphics rendering apparatus can stream to multiple client systems.

[0189] FIG. 8 is a block diagram showing, in one exemplary embodiment, components of a graphics accelerator (5100).

[0190] In an embodiment, a graphics accelerator (5100) can comprise, one or multiple graphics processing units (5110).

[0191] In an embodiment, a graphics processing unit (5110) can be a GPU comprising multiple SIMD (Single Instruction stream, Multiple Data stream) engines or MIMD (Multiple Instruction stream, Multiple Data stream) engines.

[0192] Furthermore, in an alternative embodiment, a graphics processing unit can be any processing element that can execute graphics processing pipeline. A graphics processing unit (5110) can comprise, one or multiple DSP processors, or micro processors for running graphics rendering (e.g., Imagine Stream Processor, Intel Lagrande, etc).

[0193] In an additional embodiment, a graphics accelerator (5100) can comprise, one or multiple networking interfaces (5140) coupled with said graphics processing unit(5110). A networking interface can comprise, one or multiple transceivers (5144)(e.g., Ethernet transceiver, WiFi transceiver, Infini-band transceiver, etc) and media access controller(5142).

[0194] In one embodiment, one or multiple networking interfaces can be integrated with a graphics accelerator either on the same printed circuit board, or integrated on the same chip using system-in-a-package or die stacking, or integrated on the same die.

[0195] In a further embodiment, both a graphics processing unit (5110) and a networking interface (5140) can share memory interface (5112) of video memory (5120) where the rendered frames are stored. In this embodiment, a networking interface (5140) can read or fetch the rendered frames directly from the shared memory and transmit the rendered frames out.

[0196] Furthermore, a networking interface (5140) can comprise one or multiple networking processing engines (5146)(e.g., network processing ASIC (application specific integrated circuit), network processor, etc). A networking interface can be either a stand-alone device or an integrated device (e.g., integrated with a graphics accelerator's main board, integrated with a graphics processing unit, etc).

[0197] Furthermore, a networking interface (5140) can have its own device memory (e.g., SRAM, RAM, etc) or use the video memory of a graphics accelerator for its operation and data storage.

[0198] In an embodiment where a networking interface is integrated with one or multiple graphics processing units, it can share memory interface with the graphics processing unit.

[0199] A networking interface (5140) can comprise additional functionality of data processing. For example, the network processing engine (5146) of a networking interface can have enough processing power for data compression, or data packaging, or data segmentation/packetizing, etc. The additional functionality can be either implemented as fixed logic or software.

[0200] In an embodiment, the network processing engine (5146) of a networking interface can first read the rendered frame from a graphics processing unit's video memory, then it can compress the rendered frame, packetize the compressed frame, and transmit the frame out.

[0201] The frame compression or packetizing functionality can be implemented either as hardware logic or software or firmware executed by a networking processing engine (5146) of a networking interface.

[0202] FIG. 9 is a flowchart showing, in one exemplary embodiment, process of frame compression used by a graphics accelerator before they are further processed by a graphics rendering apparatus for streaming. After compression, frames can be further transcoded according to the condition of a destination client system (e.g., screen resolution, available bandwidth, required frame rate, etc), and delivered to one or multiple client systems through local network (e.g., local area network, metropolitan area network, local wireless network, or mobile network).

[0203] There are several different embodiments of how the rendered frames are compressed and transmitted.

[0204] In one embodiment, graphics frames are transmitted first to a graphics rendering apparatus's host memory. Then they are transcoded and streamed to one or multiple client systems.

[0205] In another embodiment, rendered frames can be directly transmitted to one or multiple networking interfaces of a graphics rendering apparatus using peer-to-peer bus transactions. In peer-to-peer transmission mode, data from one peripheral device can be transmitted directly to another peripheral device using bus bridge or switch. In this mode, data of a rendered frame can be transmitted directly from the video memory of a graphics accelerator to the device memory of a networking interface.

[0206] In a third embodiment, one or multiple networking interfaces can be integrated with a graphics accelerator either on the same printed circuit board or integrated on the same chip (system-in-a-package or die stacking) or integrated on the same silicon die. The integrated networking interface can comprise programmable network processing engine for controlling one or multiple transceivers (e.g., WiFi transceiver, Ethernet transceiver, etc).

[0207] Furthermore, a networking interface can share memory or share memory controller with a graphic processing unit or graphics processing accelerator. In this case, a networking interface can access or read rendered frame directly from the shared memory or shared memory controller.

[0208] In an embodiment, graphics frames can be rendered by a graphics processing unit on virtual or non-display frame buffers. As a result, the number of graphics frames or the size of graphics frames or the number of graphics rendering contexts are not restricted by the number of displays or the size of actual displayable frame buffer of a graphics rendering apparatus.

[0209] In one embodiment, for a graphics rendering context, a graphics rendering apparatus can create one or multiple virtual frame buffers either in the graphics rendering apparatus's host memory or in the video memory of one or multiple graphics accelerators. After a graphics rendering context and its pending graphics commands are scheduled to a graphics accelerator, graphics processing unit of the graphics accelerator will render frames on the virtual frame buffer allocated to the graphics rendering context.

[0210] In an additional embodiment, to reduce transmission overhead, before rendered frames are transmitted, they can be compressed. A compression module can use any frame or image compression approach (e.g., JPEG, lossless JPEG, PNG, TIFF, GIF, motion JPEG, JPEG 2000, Progressive

Graphics File, Enhanced Compression Wavelet, TGA, etc) for compressing rendered frames with the compression approach implemented as hardware logic or software or firmware or as software hardware hybrid (programmable compression hardware logic or programmable compression engine). After the frames stored in video memory are compressed, they can be transmitted to the host memory of a graphics rendering apparatus, or to a networking interface that acts either as a separate device from a graphics accelerator or as an integrated device with a graphics accelerator, then further processed (e.g., transcoded or sent out).

[0211] In one embodiment, a frame compressor can be implemented as one or multiple GPU shader programs (e.g., program executed by a graphics processing unit, etc). Depending on the implementation, shader based frame compression can be carried out by multiple shader programs through multiple shading passes. To compress a rendered frame, the rendered frame and other reference frames will be used as inputs to a frame compression shader program.

[0212] A compression shader program can read frame data via memory fetch instructions and write the compressed data to an output buffer.

[0213] In an embodiment, to obtain compressed frame, a graphics accelerator can be programmed to use multiple passes of processing where in each pass, a shader program performing certain operations can read result data from the previous pass and output processed data that will be used by shader program of the next pass to video memory.

[0214] In a further embodiment, when a graphics processing unit comprises MIMD (Multiple Instruction stream, Multiple Data stream) engines, shader programs assigned to multiple passes can be assigned to different engines where they can be executed in parallel. Engines running different shader programs can be chained using shared memory or shared memory queue where an engine running a shader program can read output data from other engine which is running a shader program corresponding to a previous step of frame processing, process the data, and output the result to a shared memory or memory queue that can be accessed by an engine which is running a shader program corresponding to the next step of frame processing.

[0215] In one exemplary embodiment where rendered frame is compressed by a graphics processing unit using JPEG compression (e.g., JPEG, JPEG2000, JPEG XR, etc) or other JPEG alike frame compression approach, a graphics processing unit can perform the JPEG compression in the following steps, frame rendering (5190), compression preparation (5192), transformation (5194), quantization (5196), encoding (5198), and transmission (5199).

[0216] Compression preparation (5192) comprises required preprocessing on a rendered frame before it is transformed using DCT (discrete cosine transform) or wavelet transformation. Example of such preprocessing includes color space transformation, or downsampling in JPEG, or tiling in JPEG 2000, etc.

[0217] Transformation (5194) comprises, DCT (discrete cosine transform) or wavelet transformation or other similar transformation performed by one or multiple shaders programs.

[0218] Quantization (5196) comprises the process that after transformation (5194), the coefficients are scalar-quantized to reduce the amount of bits to represent them.

[0219] In an embodiment, quantization can be implemented as a stand-alone shader program or implemented by a shader program that perform both transformation (5194) and quantization (5196).

[0220] In encoding (5198) process, quantized results of the previous step are encoded. In an embodiment, this step depends on the type of applied frame compression technique. For example, in standard JPEG, encoding can comprise some form of entropy coding or customized entropy coding (e.g., run-length encoding, etc).

[0221] In transmission (5199) process, the encoded frames of the previous step are transmitted out. In one embodiment, the encoded frames are transmitted to the system memory of a rendering apparatus where they can be further transcoded.

[0222] In another embodiment, the encoded frames are transmitted to a network interface where they are packetized and sent out via a transceiver.

[0223] In a further embodiment, when a graphics processing unit (GPU) uses frame compression, it can perform JPEG compression on rendered frame using multiple passes of processing by assigning the steps of FIG. 9 to a sequence of processing passes of a graphics processing unit.

[0224] In an alternative embodiment, when a graphics processing unit (GPU) uses JPEG compression (e.g., JPEG, JPEG2000, JPEG XR, etc) and the graphics processing unit comprises MIMD engines, it can perform JPEG compression on rendered frame using a processing chain by assigning steps of FIG. 9 to different engines of a graphics processing unit to form a JPEG processing pipeline. Results from a previous processing step performed by one engine can be fed into the next processing step performed by another engine using shared memory or memory queue (e.g., queue, FIFO, circular buffer, ring buffer, etc).

[0225] As various changes can be made in the above embodiments and operating methods without departing from the spirit or scope of the following claims, it is intended that all matter contained in the above description or shown in the accompanying drawings should be interpreted as illustrative and not in a limiting sense. The invention is not limited by the specific embodiments described herein. Variations or modifications to the design and construction of this invention, within the scope of the appended claims, may occur to those skilled in the art upon reviewing the disclosure herein. Such variations or modifications, if within the spirit of this invention, are intended to be encompassed within the scope of any claims to patent protection issuing upon this invention.

What is claimed is:

1. A game appliance apparatus which can offer game appliance execution service for at least one client system wherein a game appliance apparatus can start a game appliance for a client system and said client system can send input to the game appliance apparatus via network to control game play of the game appliance and in return receive media stream of the game appliance via a network, said game appliance apparatus comprising,

at least one transceiver;

at least one control processing element that can be programmed to, on-demand retrieve game appliance from at least one game appliance distribution system or retrieve a game appliance from a local game appliance cache; and

a plurality of control processing elements wherein each control processing element can be programmed to, 1) execute a game appliance using emulation or virtualiza-

tion; 2) receive input from at least one client system; 3) capture graphics rendering command of the executed game appliance using virtual graphics driver; and 4) transmit the captured graphics rendering command to at least one graphics rendering agent over virtual graphics bus.

2. The game appliance apparatus in claim 1 further comprising at least one storage device wherein said storage device comprises cache of game appliance.

3. The game appliance apparatus in claim 1 further comprising a game appliance fetching agent wherein a game appliance is split into a plurality of pieces and said game appliance fetching agent can retrieve a piece of a game appliance when said piece of the game appliance is required during execution of the game appliance by the game appliance apparatus.

4. The game appliance apparatus in claim 1 further comprising at least one WiFi transceiver.

5. The game appliance apparatus in claim 1 further comprising a plurality of game appliance execution units wherein said game appliance execution units are assigned to a plurality of control processing elements and said game appliance execution units can execute multiple game appliances in parallel.

6. The game appliance apparatus in claim 1 further comprising, at least one graphics accelerator; and

at least one graphics rendering agent wherein said graphics rendering agent is executed by at least one control processing element and said graphics rendering agent can schedule graphics rendering command to the graphics accelerator.

7. The game appliance apparatus in claim 6 further comprising at least one compressor wherein said compressor is executed by the graphics accelerator and said compressor can compress rendered frame into compressed frame or video.

8. The game appliance apparatus in claim 6 wherein the graphics accelerator further comprising,

at least one graphics processing unit; and

at least one network interface coupled with said graphics processing unit.

9. The game appliance apparatus in claim 8 wherein the network interface further comprising at least one network processing engine.

10. A graphics rendering apparatus which can offer 3D graphics rendering and frame compression service for at least one game appliance apparatus wherein a game appliance apparatus can send graphics rendering command to the graphics rendering apparatus over network, said graphics rendering apparatus comprising,

at least one transceiver;

a plurality of graphics accelerators; and

at least one control processing element that can be programmed to, 1) receive graphics rendering command from at least one game appliance apparatus; 2) schedule graphics rendering command to the graphics accelerators; and 3) direct the graphics accelerators to compress the rendered frame,

whereby a game appliance apparatus can, execute a game appliance using emulation or virtualization; receive input from at least one client system; capture graphics rendering command of the executed game appliance using virtual graphics driver; and transmit the captured graphics rendering command to at least one graphics rendering apparatus.

11. The graphics rendering apparatus in claim 10 wherein the graphics accelerator further comprising, at least one graphics processing unit; and at least one network interface coupled with said graphics processing unit.

12. The graphics rendering apparatus in claim 11 wherein the network interface further comprising at least one network processing engine.

13. The graphics rendering apparatus in claim 11 wherein the graphics processing unit further comprising a plurality of MIMD engines wherein said MIMD engines are programmed to, render frame for game appliance or compress said rendered frame.

14. The graphics rendering apparatus in claim 10 further comprising at least one graphics rendering agent wherein said graphics rendering agent can schedule graphics rendering commands of a plurality of game appliances to the graphics accelerators.

15. The graphics rendering apparatus in claim 10 further comprising at least one WiFi transceiver.

16. The graphics rendering apparatus in claim 10 further comprising at least one video transcoder wherein said video transcoder is executed by at least one control processing element and said video transcoder transcodes rendered frames into streaming video.

17. The transcoder in claim 16 further comprising at least one video encoding accelerator.

18. The graphics rendering apparatus in claim 16 further comprising at least one video streamer wherein said video streamer is executed by at least one control processing element and said video streamer can stream transcoded video stream over network to at least one client system.

19. A game appliance system which can offer game appliance execution and rendering service for at least one client system wherein a game appliance system can start a game appliance for a client system and said client system can send input to the game appliance system via network to control

game play of the game appliance and in return receive media stream of the game appliance via a network, said game appliance system comprising,

at least one game appliance apparatus wherein said game appliance apparatus further comprising, 1) at least one transceiver; 2) at least one control processing element that can be programmed to, on-demand retrieve a game appliance from at least one game appliance distribution system or retrieve a game appliance from a local game appliance cache; and 3) a plurality of control processing elements wherein each control processing element can be programmed to, execute a game appliance using emulation or virtualization; receive input from client system; capture graphics rendering command of the executed game appliance using virtual graphics driver; and transmit the captured graphics rendering command to at least one graphics rendering apparatus; and

at least one graphics rendering apparatus wherein said graphics rendering apparatus further comprising, 1) at least one transceiver wherein said transceiver connects said graphics rendering apparatus with one or a plurality of game appliance apparatuses; 2) a plurality of graphics accelerators; and 3) at least one control processing element that can be programmed to, receive graphics rendering command from at least one game appliance apparatus; schedule the received graphics rendering command to the graphics accelerators; direct the graphics accelerators to compress the rendered frame; transcode the rendered frames into streaming video; and stream the transcoded video over network to at least one client system wherein said client system can, decode the video and display the decoded video.

20. The game appliance system in claim 19 wherein the graphics rendering apparatus further comprises a WiFi transceiver.

* * * * *