УДК 004.92

СИНХРОНИЗАЦИЯ ГИГАПИКСЕЛЬНЫХ ВИДЕОПОТОКОВ НА ВИДЕОСТЕНАХ

Д. Д. Дрижук[1], А. А. Пойда[1], А. И. Годунов[1]

Предложен подход и описана его программная реализация для визуализации на многодисплейных видеостенах видеопотоков высокого разрешения, организованных по принципу видеопирамид (по аналогии с пирамидами изображений). Основное внимание уделяется проблемам синхронизации видеотайлов и интерактивного взаимодействия пользователей с многоузловой системой.

Ключевые слова: видеопотоки, видеопирамиды, гигапиксельные изображения, сверхвысокое разрешение, видеостены.

Введение. В современной научной среде объемы экспериментальных и моделируемых данных настолько выросли, что человек, каким бы гениальным он ни был, не в силах воспринять эти данные без предварительной обработки по причине физиологических ограничений. В связи с этим все бóльшую роль играет удобное визуальное представление данных, при этом для многих научных и технических применений требуется интерактивная визуализация изменяющихся динамических сцен в гигапиксельных разрешениях. Например, проблемой является желание продемонстрировать развитие вселенной во времени на всех уровнях детализации, визуализировать взаимодействие группы белков с точностью до атомов, предоставить человеку возможность наблюдать за процессами, проходящими внутри клетки на каждом из уровней, или с высокой точностью демонстрировать карты погодных явлений в реальном времени. Для визуализации высокого разрешения требуется не только программное обеспечение, но и соответствующие технические платформы. Все чаще в научных исследованиях используются видеостены сверхвысокого разрешения, составленные из десятков дисплеев, управляемых кластером визуализации. Каждый компьютер такого кластера обычно используется для вывода изображения на 2–4 дисплея.

Для решения задачи интерактивной визуализации изменяющихся динамических сцен на видеостенах высокого разрешения можно использовать видеопотоки высокой четкости или видеопирамиды [1]. Это — два разных подхода, но оба имеют свои недостатки.

Видеопоток высокой четкости несет в себе единый не делимый по пространству видеоконтент и отличается от обычного видеотока только высоким разрешением. Недостатком такого видеопотока является отсутствие масштабируемости по разрешению. Предположим, у нас есть видеоряд, один кадр которого представляет собой гигапиксельное изображение. Если такой видеоряд объединить в единый файл или поток, то даже попытка разобрать этот файл на кадры и вывести на экран приведет к прерываниям воспроизведения в связи с перегрузкой вычислительных ресурсов.

Второй проблемой при выводе видеопотоков высокой четкости является проблема пропускной способности сети, которой соединены элементы визуализации. Например, для демонстрации видеопотока высокой четкости без потери качества со скоростью 24 кадра в секунду на видеостене, состоящей из 16 Full-HD мониторов (8K UHD, Ultra High Definition), потребуется сеть, обеспечивающая пропускную способность между хранилищем данных или сервером видеостены с каждым из мониторов не менее 1 гигабита в секунду:

$$1920 \times 1080 \text{ px} * 24 \text{ bit/px} * 16 = 796\,262\,400 \text{ bit} \approx 0.8 \text{ Gbit}.$$

Следовательно, для передачи видеопотока потребуется пропускная способность сети выше одного гигабита [8] в секунду даже при использовании наилучших методов сжатия, потому что они не могут обеспечить сжатие в шестнадцать раз без ощутимых потерь в качестве. Конечно, производители телевизоров уже выпускают 8K UHD телевизоры [8, 9], но это не означает, что не придется решать проблему с демонстрацией на видеостене, состоящей из восьми таких телевизоров.

[1] Национальный исследовательский центр “Курчатовский институт” (НИЦ “Курчатовский институт”), пл. Академика Курчатова, д. 1г, 123182, Москва; Д. Д. Дрижук, программист, e-mail: argentum_no3@bk.ru; А. А. Пойда, нач. лаборатории, e-mail: Poyda_AA@nrcki.ru; А. И. Годунов, инженер, e-mail: alex.i.godunov@gmail.com

Технической проблемой при использовании видеопотоков высокого разрешения являются также ограничения, накладываемые современными декодерами. Например, для самого прогрессивного на текущий момент кодека HEVC/H.265 существует ограничение на максимальный размер видеопотока, равный 8K UHD [2].

Видеопирамиды [1] — это видеопоток, предварительно нарезанный на смежные участки стандартного формата, при этом нарезается не только оригинальный видеопоток, но и его уменьшенные по масштабу копии, образуя так называемую "пирамиду". В зависимости от требуемого для визуализации масштаба и области видео на экран выводится только часть нарезанных участков, аналогично тому, как Google Maps [10] позволяет просматривать многомасштабные пирамиды изображений (рис. 1). Доступ к видеопирамиде можно сделать децентрализованным, чтобы потоки до каждого монитора или клиента видеостены были независимы, обеспечивая максимальную скорость доставки необходимой части изображения для каждого монитора. Это экономит как ресурсы системы, так и сетевой трафик.

Однако у этого метода есть определенные недостатки. Одним из таких недостатков является то, что каждый кусочек видеопотока представляет собой отдельный видеотайл и должен быть декодирован независимо от остальных. Однако тогда возникает вопрос о том, что надо как-то синхронизировать демонстрацию видеопотока. Для визуализации видеопирамиды на одном компьютере эта проблема решается средствами HTML5 [3] и ECMAScript [4], но данный метод не может быть использован для видеостены, так как необходимо выполнить синхронизацию не на одном компьютере, а на нескольких.

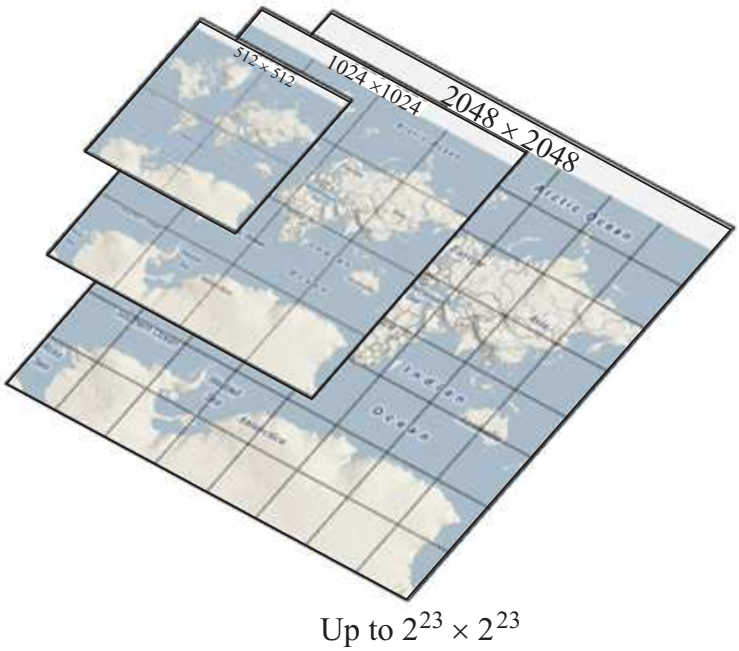

Рис. 1. Пирамида изображений

Задача синхронизации независимых видеопотоков на машинах, соединенных локальной сетью, решена в некоторых реализациях проекта mplayer [11], но на одном компьютере должна быть возможность запуска как минимум четырех полностью (с точностью до десятых долей секунды) синхронизованных плееров, чтобы они покрывали весь экран, даже если мы передвигаем видео или изменяем масштаб просмотра. Решение этой задачи было невозможно на основе той реализации, которая предоставляется с исходным кодом данного приложения, так как UDP-порт (User Datagram Protocol), на котором вещает сервер, может прослушиваться только одним приложением на одном компьютере без внедрения специальных изменений в ядро ОС.

Существует протокол RTP/RTCP [5] для синхронизации видео на двух удаленных машинах, но этот протокол разрабатывался с целью синхронизации одного видео на разных машинах, а не синхронизации видеопирамиды. Протокол хоть и обладает массой преимуществ, но имеет такой же недостаток, как и UDP-синхронизация в плеерах mplayer; кроме того, будет не очень легко запустить на разных плеерах разные фрагменты, синхронизированные этим протоколом.

Задача о позиционировании нескольких плееров в пространстве, их передвижении, масштабировании, так же как и задача выбора нужного фрагмента, по отдельности решены во многих приложениях, но ни в одном программном продукте они не объединены вместе.

В настоящей статье предложен подход и его программная реализация, обеспечивающая позиционирование нескольких плееров в пространстве, их передвижение, масштабирование, выбор нужного фрагмента и позволяющая визуализировать видеопирамиды на видеостене.

1. Подход к организации системы.

1.1. Введение в видеостены. Чтобы описать способ вывода видеопирамиды на видеостену, надо дать небольшое введение в архитектуру и схему работы видеостены.

Все чаще в научных исследованиях используются видеостены сверхвысокого разрешения, составлен-

ные из десятков дисплеев, управляемых кластером визуализации. Каждый компьютер такого кластера обычно используется для вывода изображения на 2–4 дисплея. Одна из основных технических сложностей таких систем — это синхронизация изображения на всех дисплеях.

Рассмотрим архитектуру многодисплейных видеостен более детально на примере технологии SAGE [6]. Приложения-клиенты подключаются к визуализационному кластеру с помощью библиотеки SAGE Application Interface Library (SAIL), которая направляет части визуализации на различные узлы кластера и синхронизирует смену кадров с помощью специального сервера FreeSpace Manager, также являющегося частью видеостены (рис. 2).

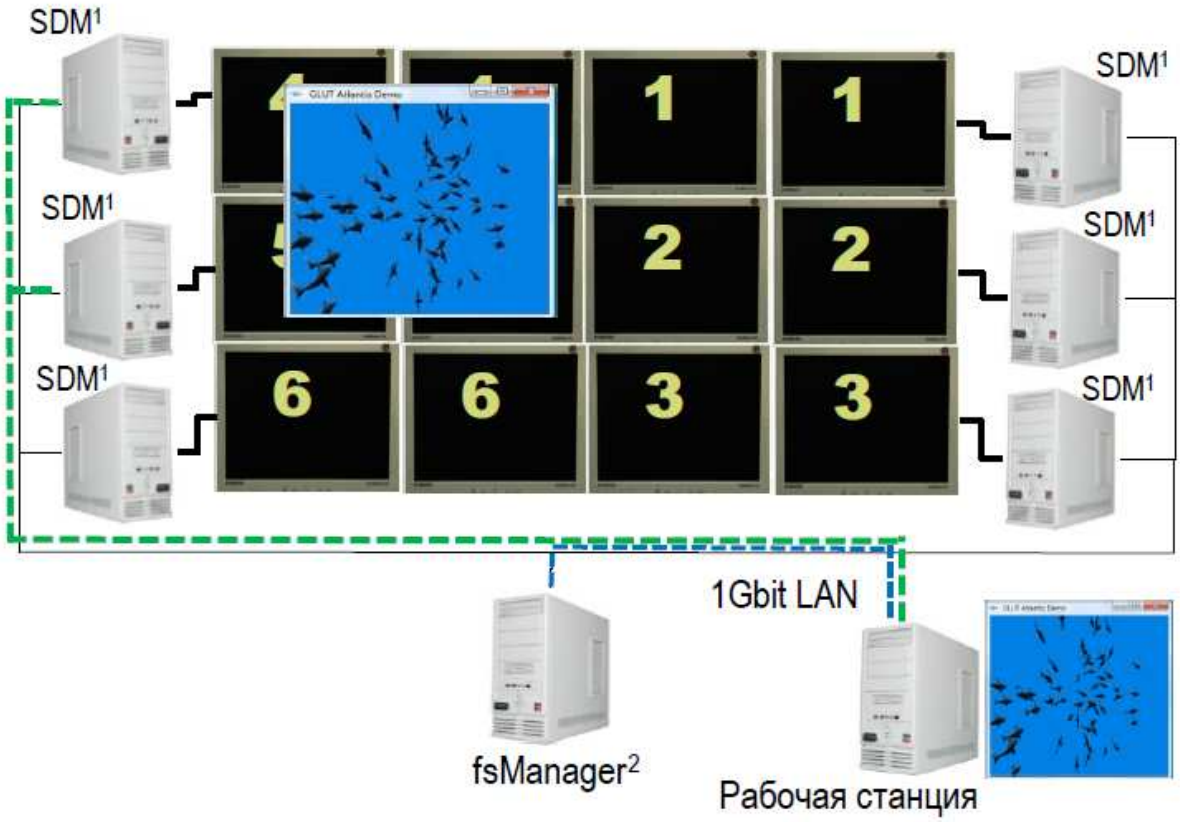

Рис. 2. Архитектура кластера визуализации для многодисплейной видеостены

В технологии SAGE каждый узел кластера, непосредственно управляющий выводом изображения на экран, является тонким клиентом и не производит обработку данных, а получает уже сформированную картинку от кластера визуализации. Это приводит к тому, что узлы вывода на экран используют лишь малую часть своих ресурсов, а SAGE-клиент при этом должен быть запущен на высокопроизводительном узле или на кластере визуализации в зависимости от количества дисплеев.

1.2. Архитектура разработанной системы. На рис. 3 приведена архитектура системы визуализации.

Основная идея заключается в запуске на каждом экране видеостены нескольких копий видеоплеера (MPlayer), вплотную состыкованных друг с другом без промежуточных панелей и полностью покрывающих все экраны видеостены (рис. 4). Дополнительные модули позиционирования плееров (video positioning system), запускаемые на каждом узле видеостены, распределяют каждому плееру видеотайлы, которые плеер должен показывать. Синхронизация между узлами и модулями позиционирования осуществляются общим контроллером (server state cast).

Список основных компонентов и их функциональное назначение приведены в таблице.

2. Реализация системы визуализации. Задача синхронизации независимых видеопотоков на машинах, соединенных локальной сетью, решена в некоторых реализациях проекта mplayer, но на одном компьютере должна быть возможность запуска как минимум четырех полностью (с точностью до десятых долей секунды) синхронизованных плееров, чтобы они покрывали весь экран, даже если мы передвигаем видео или изменяем масштаб просмотра. Решение этой задачи было невозможно на той реализации, которая предоставляется с исходным кодом данного приложения, так как UDP-порт, на котором вещает

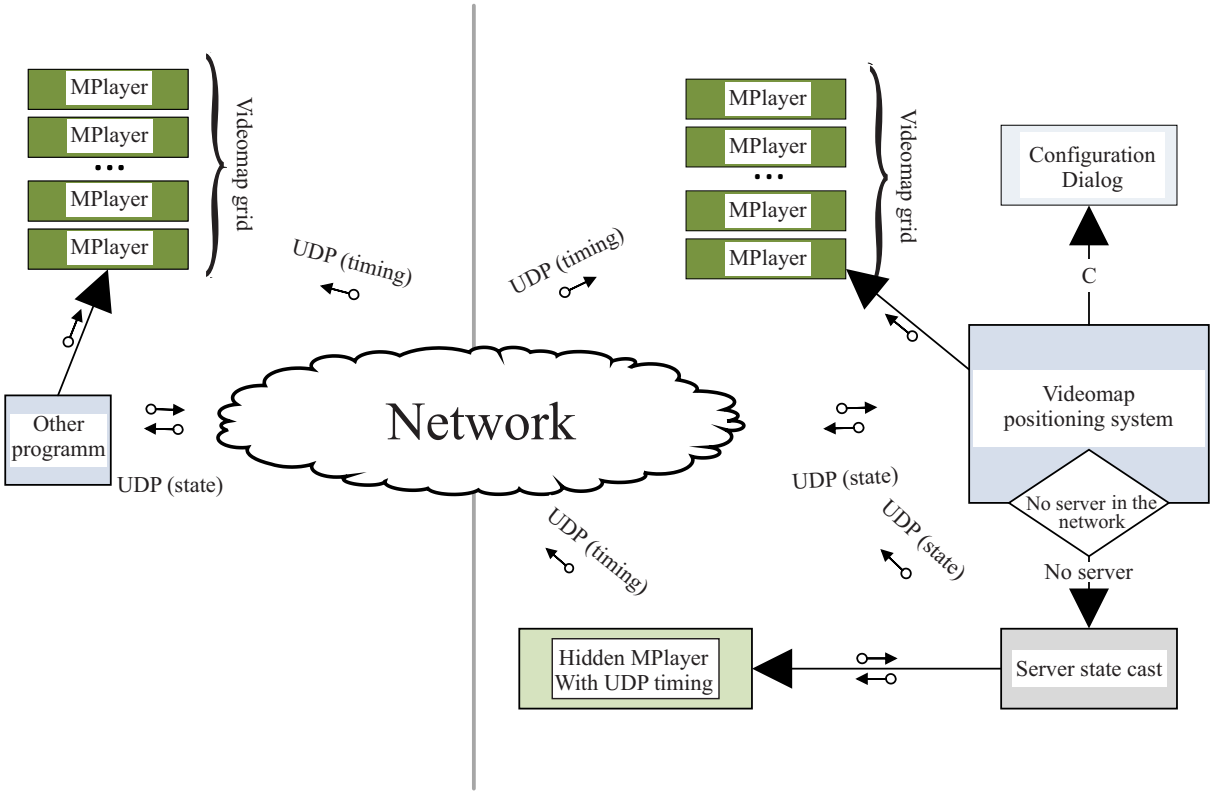

Рис. 3. Архитектура системы визуализации

Рис. 4. Визуализация видеопирамиды движения фрактала на видеостене в Институте космических исследований РАН

Список основных программных компонентов приложения и их функциональное назначение

Программные компоненты	
Название компонента	Функциональное назначение
Videomap positioning system (так же “Other program” на диаграмме)	обработка событий передвижения мыши и колесика для изменения масштаба и области просмотра; размещение видеотайлов в пространстве сцены и экранов видеостены; получение данных о состоянии воспроизводимой видеопирамиды; вызов окна настройки компонента “Configuration Dialog”; запуск плееров “MPlayer”; вызов серверного механизма в случае необходимости.
Configuration Dialog (вызывается при нажатии клавиши “C”)	редактирование конфигурации приложения.
Server state cast (вызывается, если в сети не был обнаружен уже работающий сервер)	управление теневым плеером “Hidden MPlayer”; управление синхронизацией; определение и рассылка серверных настроек, а также настроек видеостены клиентским программам.
Hidden MPlayer	воспроизведение эталонного объекта видеостены; получение информации о видеотайлах; синхронизация плееров во времени; управление проигрыванием (пауза, проигрывание, остановка).
MPlayer	воспроизведение видеотайлов.

Управляющие интерфейсы	
Название	Функциональное назначение
UDP (timing)	Набор UDP-соединений (по числу максимального количества плееров на компьютер). Отвечает за синхронизацию плееров. Сервером является Hidden MPlayer, клиентами — плееры MPlayer.
UDP (state)	Соединения, контролирующие саму программу. Передается полное состояние видеостены, либо сообщение об изменении масштаба, или перемещении. Состояние передается с заданным интервалом, а также во время взаимодействий пользователя. Управляющие сигналы принимаются сервером, передавать может любая программа. Сигналы состояний рассылаются сервером, а принимаются всеми программами.

сервер, может прослушиваться только одним приложением на одном компьютере без внедрения специальных изменений в ядро ОС.

Для того чтобы исправить недостатки, решено было переписать код mplayer и создать управляющие программные компоненты, автоматизирующие положение плееров, их запуск и синхронизацию.

В качестве плеера был взят mplayer2 [12] — переработанная версия mplayer с улучшенной поддержкой кодеков и дополнительными улучшениями. Кроме того, mplayer2 совместим с большинством приложений, разрабатывавшихся для взаимодействия с плеером mplayer.

Для программы, управляющей запуском плееров, их положением и размером, были выбраны язык C++ и библиотека Qt [7] версии 4, так как в данном варианте можно легко реализовать необходимый

интерфейс. В качестве протокола синхронизации был выбран протокол UDP, чтобы программы были полностью автономны и независимы друг от друга.

2.1. Изменения в коде mplayer2. Из кода mplayer в код одной из стабильных версий mplayer2 был внедрен UDP-протокол синхронизации двух плееров. Для того чтобы можно было запустить несколько плееров на одном компьютере, синхронизированных по данному протоколу, надо было обойти ограничение на подключение к порту, так как к одному порту может быть прикреплена только одна программа. Проще всего это сделать с помощью открытия соединений на несколько портов и простого перебора этих портов в ведомом плеере. В сети не будет перегруженности и проблем, если программа и плееры займут вплоть до 500 портов. Конечно, лучше обойтись меньшим количеством, рассчитанным исходя из максимального количества фрагментов на один компьютер.

После изменений в коде mplayer2, запущенный с параметром “-udp-master”, посылает UDP-пакеты сразу на несколько портов, следующих друг за другом, заданных парой параметров: “-udp-port” и “-udp-port-range”, задающих первый порт и количество портов соответственно. Если плеер будет запущен с параметром “-udp-slave”, то он выбирает первый свободный порт из диапазона, заданного теми же параметрами, резервирует его под себя и использует в качестве порта для синхронизации. Таким образом, плееры сами выбирают открытые окна для синхронизации. Предоставлены также параметры для выбора адресов вещания и захвата. Измененный код mplayer2 опубликован в [13].

2.2. Управляющие программные компоненты. Для того чтобы запустить нужные плееры, синхронизировать их и выбрать нужные фрагменты для каждого из файлового хранилища, нужно было написать свой скрипт или программу, которая бы контролировала весь этот процесс. Решено было использовать полноценную оконно-графическую программу, так как тогда можно дополнительно настроить взаимодействие между пользователем и видеостеной вне зависимости от конфигурации последней.

Для того чтобы программа могла передавать информацию о своем состоянии, об имени открываемого видеопотока, о его конфигурации и о конфигурации зарезервированных сервером портов, было решено использовать тот же метод, который используется в плеере — широковещательные UDP-запросы.

Программа при старте пытается найти в сети сервер, ожидая информацию о нем на первом доступном порту из диапазона зарезервированных. Если информации о сервере нет, то программа сама открывает сервер на своей стороне и выдает в сеть информацию о том, что она является сервером.

Состояние сервера включает в себя позицию левого верхнего угла видеопотока относительно левого верхнего угла видеостены, диапазоны портов, зарезервированных сервером и плеером, название воспроизводимого видеопотока, используемый размер одного фрагмента, номер слоя видеопирамиды и некоторые другие необходимые данные.

Главное окно программы — окно без каких-либо интерактивных элементов кроме набора плееров. Это окно может быть развернуто на весь экран, чтобы формировать элемент видеостены или может быть запущено как отдельный, автономный проигрыватель видеопотока. Программа предполагает, что при ее запуске доступными для обмена информацией с сервером порты будут теми же, что и при предыдущем ее запуске. Поэтому если ей был зарезервирован какой-то из этих портов, то она выбирает сохраненное состояние окна, связанное с этим портом. В состояние окна входит режим полноэкранной работы или режим оконной работы, позиция окна относительно всей видеостены и позиция окна относительно суммы экранов компьютера, на котором запущено приложение. Эти параметры позволяют программе автоматически восстанавливать видеостену без специальных действий со стороны пользователя. Каждый из параметров автоматически сохраняется при его изменении.

Программа имеет интерфейс ручной настройки и систему настройки конкретного экземпляра стены наряду с интерфейсом взаимодействия пользователя с сервером видеостены.

Взаимодействие с сервером происходит с помощью комбинаций клавиш, а также управления мышью. С помощью мыши можно переместить видеопоток относительно видеостены. Тогда информация о перемещении отсылается на сервер для обработки, а с сервера приходит новое состояние видеопотока. Можно так же открыть новый файл конфигурации видеостены.

Для изменения параметров приложения существует менеджер конфигурации, вызываемый по клавише “C”, который позволяет настроить порты сервера и клиента, порты, зарезервированные плеерами, положение программы относительно всей видеостены, пути к плеерам и папкам, содержащим файлы конфигурации видеопирамид, а также время ожидания соединения с сервером. Помимо использования менеджера конфигурации имеется еще возможность перемещения программы-клиента относительно видеостены с помощью мыши. Для этого достаточно зажать клавишу “Shift”, тогда перемещение видеопотока не произойдет, а изменится позиция окна относительно видеостены. Это сделано для удобной и быстрой настройки видеостены.

Из клавиатурных комбинаций, созданных в текущей версии программы, имеются: переключение полноэкранного режима (F), выход из приложения (Q) и открытие диалога выбора файлов видеопирамиды. Все комбинации состоят из одной клавиши без модификаторов. Текущая версия программы опубликована в [14].

3. Ограничения, выявленные в процессе реализации, и способ их решения. На этапе реализации системы был выявлен ряд проблем и ограничений, часть из которых следует описать и указать их последствия и возможные решения.

3.1. Задержки декодера. Первое ограничение, с которым мы столкнулись, было вызвано тем, что при получении сообщения о позиции видео на серверном плеере (компонента "Server state cast") клиентским плеером, клиентский плеер не успевает быстро перейти на требуемый кадр, в связи с чем в следующий момент от него снова потребуется перестроение кадров.

Проблема давно известна и решения уже разработаны. Мы использовали линейное приближение задержек, предполагающее, что задержка на декодирование всегда одинакова либо медленно зависит от времени. Таким образом, можно хранить только последнюю задержку и использовать ее величину для расчета дополнительного времени.

3.2. Скорость чтения с диска. Еще одно ограничение, с которым пришлось столкнуться, — ограничение на скорость считывания данных с диска. В использовавшейся для тестирования конфигурации каждый узел обслуживает два монитора. Каждый монитор имеет разрешение 1920×1200, что чуть превышает размеры классического Full-HD. Одним из важнейших ограничений при использовании формата Full-HD является скорость доступа к данным. Как было показано во введении, при простейшей конфигурации для видео с размерами Full-HD требуется минимальная скорость обращения один гигабит в секунду. И это только для одного видео, с частотой 30 кадров в секунду, тогда как стандарт Full-HD предполагает более высокую частоту кадров.

Помимо ограничения на скорость Full-HD существует еще проблема неполных кадров. Как и Google Maps, мы используем видео, разрезанное на куски, каждый из которых выводится отдельно в отдельном плеере. Все плееры на экране должны полностью покрывать его пространство. В результате могут появиться плееры, которые помещаются на экран только частично, однако эти плееры требуют данные в полном объеме. Количество плееров можно легко получить из следующей формулы:

$$N = \Big([H_{\mathrm{S}}/H_{\mathrm{P}}] + 2\Big) \approx \Big([W_{\mathrm{S}}/W_{\mathrm{P}}] + 2\Big),$$

где W_{S} — ширина экрана, W_{P} — ширина плеера, H_{S} — высота экрана и H_{P} — высота плеера.

Двойки в выражении обусловлены тем, что если разместить несколько плееров, то последний будет выходить за рамки экрана; это плюс один. Если же подвинуть видеопирамиду чуть вниз, то должен будет появиться еще один плеер, дополняющий видеопирамиду сверху, отсюда два плеера по обе стороны.

Еще важно учесть тот факт, что на один компьютер может приходиться несколько мониторов. Каждый монитор должен содержать свой отдельный набор плееров, не всегда связанный с предыдущим, ведь мониторы могут находиться не обязательно рядом, кроме того между ними всегда есть некоторое расстояние, как минимум в несколько единиц пикселей. Таким образом, может образовываться значительная перегрузка на чтение и декодирование видео в сравнении с реально используемыми данными, в связи с чем появляются значительные задержки.

Эту проблему можно решить тремя способами:

1) уменьшить размер фрагмента;

2) использовать более мощное оборудование;

3) создать единый сервер воспроизведения.

Последний способ не всегда возможен, если речь идет о синхронизации мониторов с высоким разрешением, что отмечено во введении настоящей статьи. Второй способ также не всегда возможен. Первый способ наиболее приемлем, но при этом требуется обойти проблему нестабильности системы, связанную с большим числом копий плееров, о чем пойдет речь в следующем подразделе.

3.3. Количество одновременно воспроизводимых видеотайлов на одном узле. В процессе реализации системы была выявлена проблема, связанная с ограничением на количество одновременно запущенных копий плеера на одном узле. Опытным путем было получено, что их число не может превышать шестнадцати. С чем именно это связано, мы пока не разобрались, но если это количество будет превышено, то плееры не будут отображать видео или будет перезагружен графический сервер X.org. Были также попытки выводить видео с помощью Open Gl ускорения, но и это не дало ощутимой прибавки в количестве. Для решения проблемы может быть использовано два способа:

1) увеличение размера фрагмента;

2) реализация нового плеера.

4. Будущая работа. В настоящее время плееры работают достаточно стабильно, но в будущем может возникнуть потребность в более качественном анализе задержек, что потребует изменить логику расчета задержки.

Управляющие программные компоненты пока не позволяют взаимодействие с видео, например прокрутку по времени, паузу, остановку и т.п.

Выводы. Разработанный подход и программная система, обеспечивающие позиционирование нескольких плееров в пространстве, их передвижение, масштабирование, выбор нужного фрагмента и позволяющие визуализировать видеопирамиды на видеостене, могут быть использованы как для визуализации сверхбольших наборов данных в фундаментальных приложениях, таких как астрономия, климатология, газо- и гидродинамика, механика сплошных сред, геофизика, так и в чисто практических целях, например для экологического мониторинга или дистанционного зондирования.

Новые алгоритмы интерактивной визуализации видеоданных сверхвысокого разрешения найдут широкое применение в военных технологиях, например для взаимодействия с видеосистемами беспилотных летательных аппаратов или для слежения за ближним космосом.

Работа выполнена в рамках государственного контракта № 07.514.11.4140 ФЦП “Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2013 годы”.

СПИСОК ЛИТЕРАТУРЫ

1. *Янушевич В.Д., Жижин М.Н., Пойда А.А., Новиков А.М.* Разработка алгоритмов многомасштабной интерактивной визуализации гигапиксельных видеопотоков // Вычислительные методы и программирование. 2012. **13**. 153–159.
2. *Sullivan G.J., Ohm J.-R., Han W.-J., Wiegand T.* Overview of the High Efficiency Video Coding (HEVC) standard // IEEE Trans. on Circuits and Systems for Video Technology. 2012. **22**, N 12. 1649–1668.
3. *Pilgrim M.* HTML5: up and running. Cambridge: O’Reilly Media, 2010.
4. *Zakas N.* ECMAScript. Professional JavaScript for web developers. New York: Wiley, 2009.
5. *Zurawski R.* RTP, RTCP, and RTSP protocols // The Industrial Information Technology Handbook. Boca Raton: CRC Press, 2004. 28-1–28-11.
6. *Nam S., Deshpande S., Vishwanath V., Jeong B., Renambot L., Leigh J.* Multiapplication, intertile synchronization on ultra-high-resolution display walls // Proc. of the First Annual ACM SIGMM Conference on Multimedia Systems. New York: ACM Press, 2010. 145–156.
7. *Боровский А.* Qt 4.7+ Практическое программирование на C++. СПб.: БХВ-Петербург, 2012.
8. www.extremetech.com/extreme/130238-8k-uhdtv-how-do-you-send-a-48gbps-tv-signal-over-terrestrial-airwaves
9. http://www.hdtvtest.co.uk/news/panasonic-uhdtv-8k-4k-201208292152.htm
10. https://maps.google.ru/
11. http://www.mplayerhq.hu
12. http://www.mplayer2.org
13. https://github.com/complynx/mplayer2/
14. https://bitbucket.org/complynx/tiledvideos

Поступила в редакцию
19.04.2013