УДК 004.92

РАЗРАБОТКА АЛГОРИТМОВ МНОГОМАСШТАБНОЙ ВИЗУАЛИЗАЦИИ ГИГАПИКСЕЛЬНЫХ ВИДЕОПОТОКОВ

В. Д. Янушевич[1], М. Н. Жижин[1], А. А. Пойда[1], А. М. Новиков[1]

Рассмотрены алгоритмы интерактивной визуализации событий с различной степенью детализации по времени и пространству в больших потоках и массивах данных. В основе работы лежит идея разбиения гигапиксельных видеопотоков на многомасштабные видеопирамиды, состоящие из видеопотоков меньшего разрешения, что позволяет транслировать данные гигапиксельных визуализаций на тонкие клиенты в виде веб-браузеров. Для решения поставленной задачи были разработаны новые алгоритмы временно́й и пространственной синхронизации нескольких потоков видео в современных веб-браузерах с поддержкой технологий HTML5. Проведен ряд экспериментов по распределению нагрузки на многопроцессорную систему при кодировании видеопирамид; получены оценки оптимального разбиения гигапиксельных видеопотоков для параллельной передачи данных по сети и синхронной визуализации в современных веб-браузерах. Созданное на основе разработанных алгоритмов программное обеспечение использовалось для визуализации пролета Венеры по диску Солнца 5 июня 2012 г., заснятого американским спутником Solar Dynamics Observatory (SDO). В результате была получена многомасштабная видеопирамида, доступная для просмотра в веб-браузере с возможностью изменять масштаб и область просмотра.

Ключевые слова: видеопотоки, видеопирамиды, гигапиксельные изображения, сверхвысокое разрешение, Gigapan Timemachine.

Введение. В последнее время все более широкое распространение получают изображения в сверхвысоком разрешении. Вычислительные мощности современных компьютеров позволяют непосредственно в веб-браузере просматривать многомасштабные пирамиды изображений — примером этой технологии могут служить известные сервисы Google maps или Яндекс.Карты. Для многих научных и технических применений требуется интерактивная визуализация изменяющихся динамических сцен в гигапиксельных разрешениях. Используя для построения пирамид не изображения, а видеопотоки, можно получить очень гибкую систему визуализации событий с возможностью изменения масштаба и скорости проигрывания.

С ростом производительности суперкомпьютеров становится актуальной задача потоковой визуализации `in situ` результатов численного моделирования, поскольку сохранение промежуточных результатов в дисковом хранилище снижает общую производительность вычислителя. Задачи интерактивной визуализации многомасштабных физических моделей (например, турбулентности) прямо по ходу расчетов можно рассматривать как еще один возможный источник видеопирамид сверхвысокого разрешения. В отличие от традиционного подхода к научной визуализации, когда по результатам расчетов готовится последовательность JPEG- или GIF-изображений, которые показываются в виде быстро меняющихся слайдов или объединяются в видеофайл AVI или QuickTime, предлагаемый подход с использованием видеопирамид позволит обеспечить не только переходы по времени, но и выделение и приближение интересующих областей модельных данных.

1. Обзор существующих технологий. Методы визуализации гигапиксельных изображений и видеопотоков. Даже небольшая серия изображений в гигапиксельном разрешении может занимать терабайты, и передать ее обычными путями (по электронной почте, загрузить с веб-сайта) не представляется возможным. Эта проблема решается специализированным сервисом для потоковой трансляции данных нескольким клиентам. Клиент в каждый момент времени запрашивает лишь малую часть общего объема информации, основанную на текущей области просмотра и масштабе. Для этого каждое гигапиксельное изображение конвертируется в так называемую пирамиду изображений [1]. Это многомасштабное представление обычно получается следующим путем. Изображение делится на набор плиток

[1] Национальный исследовательский центр “Курчатовский институт” (НИЦ “Курчатовский институт”), пл. Академика Курчатова, д. 1г, 123182, Москва; В. Д. Янушевич, лаборант-исследователь, e-mail: vavius@gmail.com; М. Н. Жижин, нач. лаб., e-mail: jjn@kiae.ru; А. А. Пойда, вед. инженер, e-mail: poyda@wdcb.ru; А. М. Новиков, аспирант, e-mail: novikov@wdcb.ru

(tiles) фиксированного небольшого размера (обычно это 256 × 256 пикселей). Далее размер исходного изображения уменьшается вдвое и нарезка происходит еще раз (рис. 1). Таким образом, получается следующий уровень в многомасштабной пирамиде. Процесс уменьшения и нарезки обычно повторяется до тех пор, пока все изображение не поместится в одну плитку.

Для просмотра полученных пирамид на клиенте используется JavaScript-программа (Silverlight или Flash), позволяющая плавно перемещаться по изображению и изменять масштаб.

Для гигапиксельной визуализации динамических событий можно использовать наборы изображений с некоторым временны́м разрешением и подменять их несколько раз в секунду на клиенте, но это неизбежно приведет к подвисаниям и задержкам из-за сильного увеличения нагрузки на процессор и пропускную способность сети. Более эффективным методом является кодирование последовательностей изображений в видеопоток [3].

На сегодняшний день существует только одно решение для визуализации гигапиксельных видеопотоков в браузере — это Gigapan Timemachine [2]. Данная технология хоть и позволяет изменять масштаб и область показываемого видео, однако воспроизводится лишь один видеопоток единовременно, что приводит к большим областям пространственного перекрытия соседних потоков. При смещении окна просмотра на четверть в любом направлении начинает воспроизводиться новый поток. Таким образом, если полностью убрать перекрытия, то размер видеопирамиды Gigapan можно уменьшить в 16 раз.

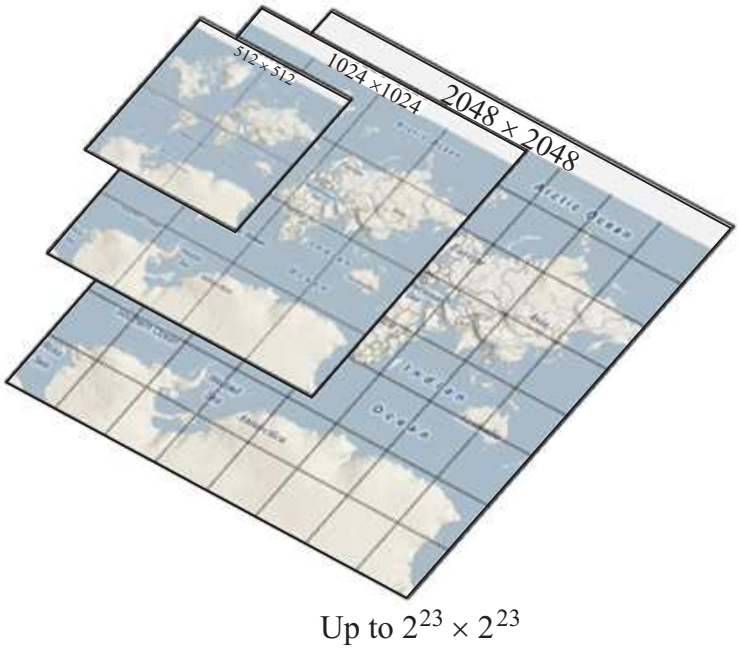

Рис. 1. Многомасштабная пирамида изображений

2. Детали реализации.

2.1. Общая архитектура взаимодействия. Система многомасштабной интерактивной визуализации данных состоит из серверной и клиентской части. Серверная часть отвечает за создание, хранение и трансляцию многомасштабных видеопирамид. Клиентская часть написана на JavaScript, поэтому для просмотра данных не нужно дополнительное программное обеспечение, поскольку достаточно использовать любой современный веб-браузер.

Сервер выполняет следующие функции:

— создание видеопирамид из последовательностей изображений сверхвысокого разрешения;

— поддержка REST-сервиса для многопоточной передачи запрошенных видеопотоков клиентам; в самом простом варианте этот сервис реализуется как веб-сервер, отдающий статический контент в виде видеопотоков.

Клиентская JavaScript-программа загружает необходимые видеопотоки с сервера и отображает их в браузере. Несколько потоков выстраиваются в единую целостную картинку и проигрываются одновременно. Пользователь может изменять область просмотра и масштаб, при этом происходит параллельная подгрузка новых потоков и их отображение. Использование стека веб-технологий автоматически решает проблему кэширования потоков, т.е. веб-браузер сохраняет в свой внутренний кэш уже загруженные потоки и при повторном просмотре некоторых областей подгрузка потоков заново не производится. Клиент также может предварительно подгружать потоки, которые скорее всего пользователь захочет посмотреть, это обычно соседние плитки относительно текущей области просмотра.

2.2. Формат представления данных видеопирамиды. Сервер может хранить несколько видеопирамид, сами видеопирамиды могут быть разных размеров и с разной частотой кадров, поэтому необходимо иметь способ описания параметров каждой видеопирамиды. Пирамиды описываются xml-файлом следующего вида:

```
<VIDEO_PROPERTIES
        WIDTH="3840" HEIGHT="3888" <!– высота и ширина всей пирамиды –>
```

TILE_WIDTH=“768” TILE_HEIGHT=“432” <!– размеры одной плитки –>
/>

Это минимально необходимая информация для отображения пирамиды. Помимо этого, в xml-описание можно легко добавить поддержку мета-информации с текстовым описанием визуализируемого события, временными кодами с интересными моментами или последовательность перемещений камеры для организации виртуальных экскурсий на клиенте.

Сервер видеопирамид — это REST-сервис, который по запросу отдает видеопоток для заданного уровня масштаба и строки-столбца пирамиды. Формат запросов (и способ хранения данных пирамиды) было решено взять из одного из готовых решений существующих для просмотра гигапиксельных изображений. Наиболее распространены два формата пирамид: Zoomify и DeepZoom [4, 5]. Был выбран вариант Zoomify как более простой (рис. 2).

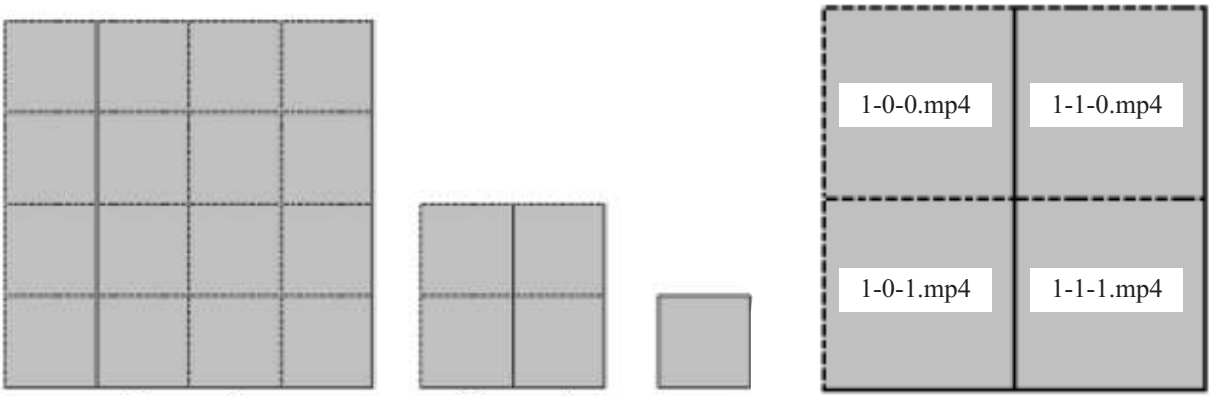

Рис. 2. Пример пирамиды в формате Zoomify

Рис. 3. Схема наименования файлов видеопотоков

Файлы видеопотоков (рис. 3) хранятся в директориях группами по 256. Хотя сами видеопирамиды могут храниться в распределенных базах данных или файловых системах, REST-сервис имеет единый интерфейс, чтобы отдавать по запросу нужную плитку.

Такой REST api используется JavaScript-клиентом асинхронно с помощью технологии AJAX для отображения видеопирамид в браузере.

2.3. Алгоритмы отрисовки файлов. JavaScript-клиент использует последние веб-технологии, такие как HTML5 и CSS3. Это накладывает ограничения на количество поддерживаемых веб-браузеров, но на старом стеке технологий невозможно создать решение, сравнимое по производительности с текущим.

Клиент построен на модульной архитектуре для поддержки разных источников и форматов видеопирамид. Основные части архитектуры следующие.

1. Обработчик пользовательского ввода. Перехватывает события передвижения мыши и колесика для изменения масштаба и области просмотра.

2. Обертка для источника данных о видеопирамиде. Поддерживает данные о формате используемой пирамиды и возвращает URL плитки по ее координате в пространстве области просмотра.

3. Контейнер для всех видеопотоков. Его размер задается равным размеру текущего уровня пирамиды. При изменении масштаба или области просмотра перемещается весь контейнер, при этом подгружаются новые плитки, а воспроизведение старых останавливается. Еще одна функция контейнера — обеспечение синхронизации отдельных потоков.

4. Менеджер кэширования. На основе текущей области просмотра подгружает заранее необходимые потоки. Идет одновременная подгрузка всех соседних плиток как на данном уровне пирамиды, так и на соседних уровнях. При заполнении кэша в первую очередь убираются наиболее удаленные плитки от текущей области просмотра.

Изменение масштаба и области просмотра осуществляется сразу для всех потоков путем задания масштаба и координаты содержащему их контейнеру. Здесь используются возможности CSS3:

```
#videoContainer {
        transform:          matrix(1, 2, 3, 4, 5, 6);
```

```
        -webkit-transform:  matrix(1, 2, 3, 4, 5, 6);
        -moz-transform:     matrix(1, 2, 3, 4, 5, 6);
        -ms-transform:      matrix(1, 2, 3, 4, 5, 6);
}
```

Поскольку на сегодняшний день официально стандарт CSS3 находится в разработке, браузеры поддерживают только альтернативы стандартным свойствам со своими префиксами. При задании преобразований с помощью свойства `transform` объектная модель документа не изменяется, а преобразуемый элемент меняет свое местоположение лишь при отрисовке. С точки зрения производительности этот метод предпочтительней, так как браузеры поддерживают аппаратное ускорение для свойства `transform`.

Значения смещения и масштаба задаются матрицей аффинного преобразования на плоскости. Свойство matrix имеет шесть аргументов: $\mathrm{matrix}\,(1,2,3,4,5,6) = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 0 & 0 & 1 \end{pmatrix}$. Последняя строка для плоских преобразований всегда (0 0 1), поэтому шести аргументов достаточно (рис. 4).

scale(a) $\begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{pmatrix}$ scaleX(x) $\begin{pmatrix} x & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ scaleY(y) $\begin{pmatrix} 1 & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & 1 \end{pmatrix}$ translateX(x) $\begin{pmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ translateY(y) $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}$ translate(x,y) $\begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}$

skewX(x) $\begin{pmatrix} 1 & \tan x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ skewY(y) $\begin{pmatrix} 1 & 0 & 0 \\ \tan y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ skew(x,y) $\begin{pmatrix} 1 & \tan x & 0 \\ \tan y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ rotate(θ) $\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Рис. 4. Матрицы преобразований для масштабирования, перемещения, поворота и искажения

Таким образом, для сдвига на (X, Y) и масштабирования в `Scale` раз значение свойства transform имеет вид: matrix (Scale, 0, 0, Scale, X, Y).

2.4. Временна́я синхронизация нескольких видеопотоков. Поскольку клиент одновременно воспроизводит несколько видеопотоков, которые при этом могут иметь общую сторону, необходимо проработать механизм синхронизации всех потоков, чтобы картинка не разваливалась на отдельные плитки, а воспринималась как единое целое.

Для синхронизации потоков используется следующий алгоритм.

1. Все активные видеопотоки заносятся в отдельный список.

2. Видеоконтейнер хранит глобальное время — количество секунд, прошедшее с начала ролика. Глобальное время принимается равным времени первого потока в списке активных в том случае, если разница между ними меньше 3 секунд, иначе всем потокам время выставляется в соответствии с глобальным.

3. С периодичностью в 0.1 секунды вызывается функция для синхронизации каждого активного видеопотока с глобальным временем. В случае если разница его времени и глобального превышает определенное значение (зависит от частоты кадров потока), то потоку выставляется новое значение времени, равное глобальному. Если разница не превышает заданного порога, то для потока изменяется скорость воспроизведения с целью “догнать” глобальное время.

Таким образом, активные видеопотоки могут воспроизводиться с разной скоростью для компенсации отставания или опережения относительно первого потока. Такой метод работает намного более плавно, нежели прямое выставление одинакового временно́го положения каждому потоку.

2.5. Выбор видеокодека и параметров кодирования потоков. На сегодняшний день наиболее перспективным выглядит кодек H264. Он хорош всем: качество изображения, размер получающегося файла, скорость кодирования, аппаратное ускорение в современных процессорах. Согласно исследованию видеокодеков [6], проведенному в лаборатории компьютерной графики и мультимедиа ВМК МГУ, среди всех кодеков уверенно лидирует свободная реализация H264 — x264.

Реализация x264 имеет самую высокую степень сжатия информации при фиксированном качестве картинки, при этом скорость кодирования видео остается на приемлемом уровне (рис. 5).

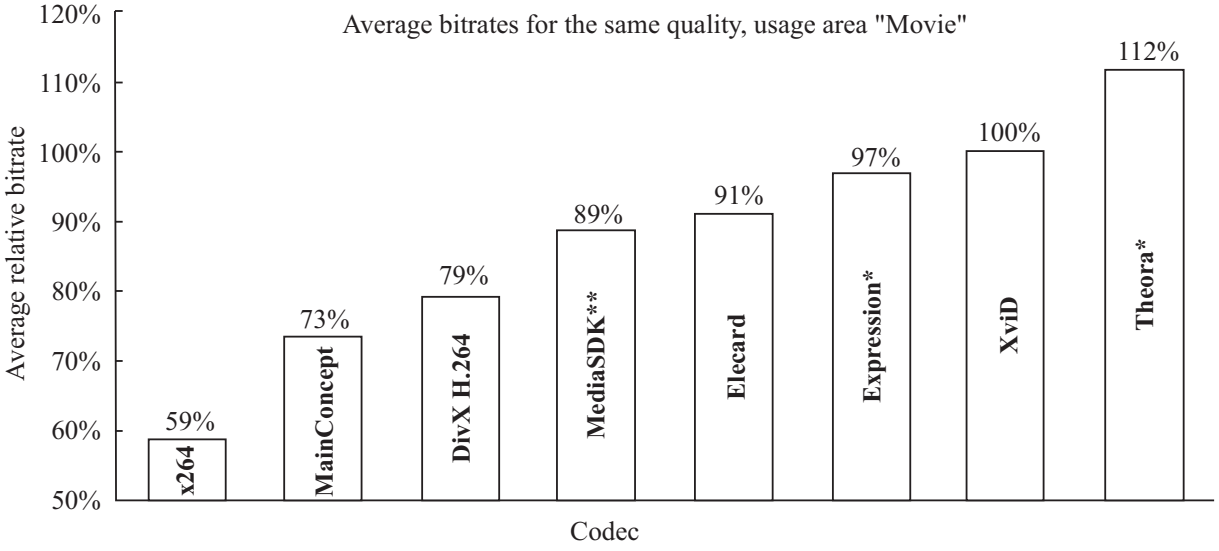

Рис. 5. Сравнение видеокодеков. Средний битрейт видео для одинакового качества картинки

Стандарт H.264/MPEG-4 определяет 21 профиль — набор возможностей декодеров/энкодеров для разных типов приложений. Для данной системы лучше всего подходит High Profile (HiP): поддерживается большинством реализаций h264, а с точки зрения стандарта — это профиль для видеоматериала высокой четкости. Существуют профили и для более качественного кодирования, но, во-первых, возрастает битрейт видео и требуются более широкие каналы передачи данных, а во-вторых, возрастает нагрузка на клиента при декодировании (рис. 6).

Особое внимание уделялось скорости декодирования потоков для одновременной трансляции девяти видео-тайлов. Нагрузка на процессор при воспроизведении видео зависит от следующих факторов.

1. Битрейт видео. Очевидно, что чем больше информации в секунду необходимо обрабатывать, тем больше будет нагрузка при декодировании.

2. Алгоритмическая сложность декодирования. Например, реализация x264 предоставляет на выбор два алгоритма энтропийного кодирования: CABAC (Context-Adaptive Binary Arithmetic Coding) и CAVLC (Context-Adaptive Variable-Length Coding). Алгоритм CABAC лучше сжимает информацию (в среднем на 10–15%), но требует больше вычислительных ресурсов и труден в распараллеливании.

Отказ от использования сложных вычислительных алгоритмов уменьшает нагрузку, но одновременно увеличивается битрейт видео и размер файла так, что выигрыш от простоты декодирования может легко нивелироваться возросшим потоком информации для обработки. Таким образом, отключать имеет смысл лишь несколько самых ресурсоемких алгоритмов.

3. Профилирование. Для сравнения эффективности предложенных алгоритмов были использованы видеопотоки одной из пирамид, построенной на технологии Gigapan Timemachine. Видеопирамиды от Gigapan закодированы с большим пространственным пересечением, а клиент отображает лишь один видеопоток одновременно. Алгоритмы, предложенные в настоящей статье, позволяют снизить затраты на дисковое пространство для хранения пирамид в 16 раз. При этом также снижается трафик к клиенту, но увеличивается нагрузка на процессор клиента.

Сравнение производилось на одних и тех же данных (одинаковых видеопотоках). Тестировалась нагрузка процессора, переданные данные и время подгрузки новых потоков. Замеры производились для следующих сценариев использования:

— Level 0 — воспроизведение нулевого уровня целиком;

— Level 0+ — масштабирование и небольшое перемещение нулевого уровня на протяжении всей длительности ролика таким образом, чтобы новые потоки не подгружались (изменения масштаба и положения совсем небольшие);

— Level 0–3 — масштабирование вокруг одной точки нулевого до самого большого масштаба с ожиданием прогрузки потока всех промежуточных уровней; панорамирование не производилось; таким образом, на каждый уровень масштаба достаточно подгрузки одного нового потока;

— Level 3 pan — просмотр пирамиды от крайнего левого потока до самого правого максимального

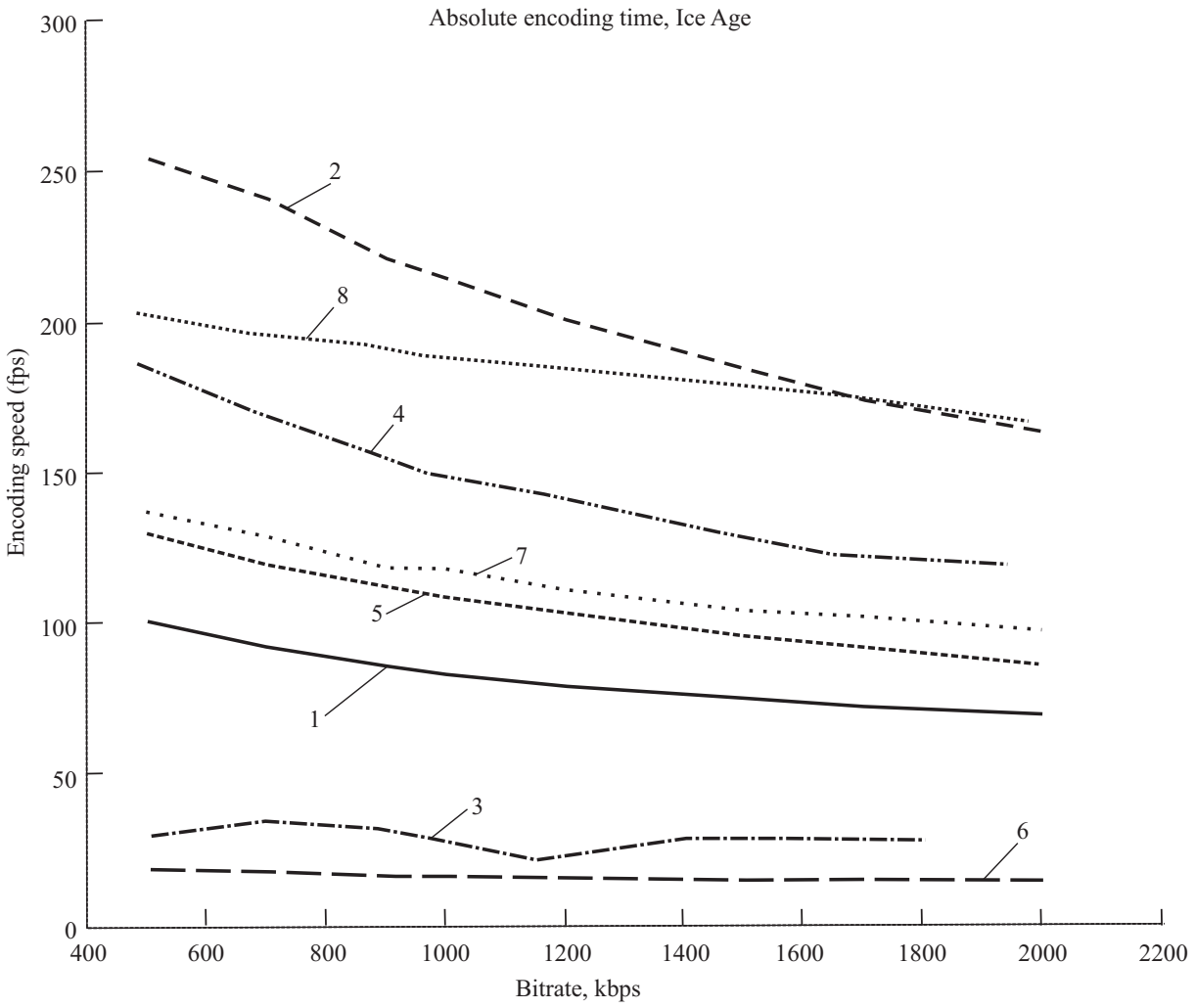

Рис. 6. Сравнение видеокодеков. Скорости кодирования видео: 1) DivX h.264, Normal preset; 2) Eleoarg, Normal preset; 3) MS Expression Encoder, Normal preset; 4) MediaSDK, Normal preset; 5) Main Concept, Normal preset; 6) Theora, Normal preset; 7) x264, Normal preset; 8) XviD, Normal preset

масштаба (всего 9 потоков);

— Level 3 pan cached — просмотр пирамиды повторно после кэширования браузером.

Результаты сравнения представлены в табл. 1–3.

Таблица 1
Сравнение загрузки ЦП для разных сценариев использования

	Gigapan Timemachine	Tiled Video
Level 0	5.9%	5.3%
Level 0+	8.8%	6.1%
Level 0–3	12.1%	15.1%
Level 3 pan	15.3%	19.8%

Таблица 2
Трафик (Мбайт), переданный во время просмотра ролика целиком

	Gigapan Timemachine	Tiled Video
Level 0	12.7	12.7
Level 0–3	139.4	120.1
Level 3 pan	413.3	274.9

Как и ожидалось, предложенные алгоритмы (Tiled Video) сильнее нагружают процессор при просмотре уровней большого масштаба и панорамировании. Для нулевого уровня воспроизводится лишь один видеопоток, поэтому результаты почти одинаковы.

Результаты небольшого масштабирования (Level 0+) показывают преимущество нового свойства CSS3 transform для изменения местоположения и размера элементов в браузере.

В целом, загрузка процессора остается в разумных пределах и никак не сказывается на отзывчивости системы и других приложений.

Переданный трафик для нулевого уровня одинаков, хотя при масштабировании вокруг неподвижной точки немного отличается. Возможно, это из-за того, что Timemachine начинает подгружать соседний поток уже при небольшом смещении. При панорамировании видна существенная экономия трафика. Требования к пропускной способности интернет-соединения зависят от размера одной плитки и области просмотра. Для плиток 640 × 480 и области просмотра 1200×960 необходимо интернет-соединение около 15 Мбит/сек.

Таблица 3

Время прокрутки пирамиды по горизонтали от первого до последнего потока

	Gigapan Timemachine	Tiled Video
Level 3 pan	54.6	38.2
Level 3 pan cached	26.7	27.5

В табл. 3 приведены данные по времени полной прокрутки пирамиды от одного конца до другого. Эти измерения имеют низкую точность (прокрутка производилась вручную, скорость могла изменяться от измерения к измерению), но общую картину отражают. Во время прокрутки при начале загрузки каждого видео производилось ожидание окончания этой загрузки и завершения процесса синхронизации потоков, после чего прокрутка возобновлялась. Timemachine начинает подгружать новые потоки при смещении на 0.25 ширины, из-за этого количество подгружаемых видеопотоков было больше.

Во втором проходе, когда браузер закэшировал все потоки, время прокрутки почти не отличается и обусловлено больше механизмом синхронизации потоков.

В целом, предложенные алгоритмы визуализации позволяют просматривать многомасштабные пирамиды с поддержкой изменения “на лету” масштаба и области просмотра в веб-браузере, причем потребление ресурсов остается довольно низким и вполне укладывается в современные рамки производительности ПК и скорости передачи данных.

Выводы. В результате работы были созданы новые алгоритмы визуализации научных данных в сверхвысоком разрешении. Современные веб-технологии и распределенные системы кодирования видеопирамид позволяют осуществлять трансляцию гигапиксельных видеопотоков по сети Интернет с поддержкой динамической смены масштаба, области просмотра и навигацией по временно́й шкале.

Работа выполнялась 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. *Жижин М.Н.* Интерактивная программа “Всемирный телескоп” // Земля и Вселенная. 2011. N 4. 70–79.
2. *Sargent R., Bartley C., Dille P., Keller J., LeGrand R., Nourbakhsh I.* Timelapse GigaPan: Capturing, Sharing, and Exploring Timelapse Gigapixel Imagery // Proc. of the Fine Intl. Conference on Gigapixel Imaging for Science. Pittsburgh, 2010. 1–9.
3. *Matson J.* Seeing the big (and small) picture: panoramic tool lets users observe dynamic imagery. 2011 (http://www.scientificamerican.com/article.cfm?id=gigapan-time-machine).
4. Deep Zoom File Format Overview (http://msdn.microsoft.com/enus/library/cc645077(VS.95).aspx#Sparse_Images).
5. *Smith A.* Introducing zoomify image. 2007 (http://hdl.handle.net/1813/5410).
6. Sixth MPEG-4 AVC/H.264 Video Codecs Comparison (http://compression.ru/video/codec_comparison/h264_2010/).

Поступила в редакцию
13.11.2012