# Синхронизация распределённого воспроизведения мультмедийных данных на полиэкранах*

***Б.С. Долговесов , М.А Городилов , И.Д. Храмцов***
bsd@iae.nsk.su|gorodilovm@gmail.com|khramtsov9203@ya.ru
Институт автоматики и электрометрии СО РАН, Новосибирск

*Статья посвящена решению некоторых вопросов отображения распределенных мультимедийных данных на больших экранах. В частности, рассмотрены проблемы синхронизации процесса параллельной обработки и вывода фрагментов видеоизображения на соответствующие экранные модули полиэкранной системы отображения. Предложен алгоритм синхронизации и его программная реализация с использованием графических ускорителей, обеспечивающий визуальную непрерывность динамических сцен при отображении на полиэкранах.*

**Ключевые слова:** *полиэкран, синхронизация, мультимедийные данные, распределённый рендеринг.*

# Synchronization of distributed data multmediynyh play split screen*

***B.S. Dolgovesov, M.A. Gorodilov, I.D. Khramtsov***
Institute of Automation and Electrometry of the Siberian Branch of the RAS, Novosibirsk, Russia

*When developing visualization systems based on tiled display walls it's crucial to provide seamless video playback across all individual tiles of the display wall. This paper proposes a GPU based software approach to solving the problem of inter-tile synchronization. The proposed approach doesn't require usage of specialized hardware, allows to effectively utilize frame buffering mechanism and thus is more resistant to irregular frame input rates than some other existing approaches to synchronization.*

**Keywords:** *tiled display walls, video synchronization, distributed rendering.*

## Введение

Внастоящее время актуальными являются решения для высококачественного отображения данных на экранах большого размера в многофункциональных мультимедийных системах для различных приложений. В одних приложениях это обуславливается необходимостью одновременной демонстрации информации большому числу пользователей (видеоконференции, ситуационные центры, спортивные мероприятия), в других – высокими требованиями к разрешению изображения (системы визуализации научных данных, интерактивные обучающие системы). Распространённым решением для этих целей является использование полиэкранов – набора из нескольких экранных модулей (плазменных панелей, мониторов или проекторов), составляющих один большой экран.

Для вывода изображений нарядус применением специализированных устройств, используются видеопроцессоры на основеперсональных компьютеров (ПК). Из-за высокого разрешения, необходимого для генерации выходного изображения полиэкрана, ресурсов одного ПК в большинстве случаев оказывается недостаточно. Самым распространённым решением этой проблемы является использование распределённого рендеринга – подхода, когда множество объединённых в сеть ПК, образующих вычислительный кластер, параллельно обрабатывают отдельные фрагменты общего изображения с последующим выводом их на соответствующие модули полиэкрана. Однако без дополнительной синхронизации различные ПК кластера будут отображать кадры в разное время из-за непостоянных задержек в передаче сетевых сообщений, а также различий в нагрузке на процессор и заполненности буфера видеокарты. Кроме того, различия по частоте и фазе развёртки экранных

модулейтакжевносятгрудноконтролируемые временные погрешности.Этоприводитк тому, что видеокадры, являющиеся частями общего изображения,будут отображаться на соответствующих экранных модулях в разное время. Такие различия во временах отображения кадров приводят к нарушению визуальной непрерывности изображения и отрицательно сказываются на субъективном восприятии видео зрителем. Минимальное значение разницы во времени кадров, приводящее к такому негативному эффекту, зависит от динамичности воспроизводимого видео и конфигурации полиэкрана и составляет от 15 до 30 мсек [1].

Предлагаемый в данной статье метод синхронизации видеопотоков в вычислительном кластере позволяет обеспечить визуальную непрерывность изображения на полиэкранной системе без использования дорогостоящих специализированных аппаратных средств.

Работа опубликована при финансовой поддержке РФФИ, грант 15-07-20347

## Синхронизация видеоданных для полиэкранного отображения

**Обзор существующих подходов.** Существующие подходы синхронизации видеопотоков для полиэкранных систем можно разделить на программные и аппаратные. Аппаратные решения используют специализированные внешние устройства для синхронизации развёртки экранных модулей или обновления данных в буфере графических акселераторов. Программные решения реализуют тот или иной протокол синхронизации передачи данных между компьютерами кластера и времени их обработки. Это позволяет снизить различия во времени отображения кадров.

В качестве примеров аппаратных решений можно рассмотреть устройства синхронизации *ATI FirePro S400* и *Nvidia Quadro Sync* со схожими характеристиками, объединяющие каждый до четырёх графических ускорителей на одном ПК с возможностью синхронного вывода данных, в том числе и на полиэкран. Данные модули совместимы лишь с весьма ограниченным набором моделей специализированных и дорогостоящих видеокарт.

Пример программного решения – алгоритм синхронизации для полиэкранного отображения на базе масштабируемой адаптивной графической среды SAGE (Scalable Adaptive Graphics Environment) [2]. Алгоритм использует два варианта синхронизации: с помощью сетевого барьера и с помощью синхронизации времени работы компьютеров. В первом варианте все компьютеры, входящие в состав кластера, осуществляют ожидание на сетевом барьере перед выводом данных на соответствующие экранные модули.

Когда все части изображения готовы и потоки данных на всех компьютерах доходят до барьера, происходит одновременный вывод изображений на все экранные модули полиэкрана. При таком подходе, в случае возникновения задержки на одном из компьютеров кластера, происходит задержка в работе всей системы. Во втором варианте время на всех компьютерах кластера синхронизируется с помощью протокола *NTP (Network Time Protocol)*. Один из компьютеров является главным — на каждом новом кадре он рассылает всем остальным компьютерам время, в которое они должны вывести свою часть изображения на экранный модуль. Это время вычисляется путём прибавления к текущему времени главного компьютера некоторой эмпирически задаваемой константы (*Presentation Time offset, PTO*), неудачный выбор которой может отрицательно повлиять на работу алгоритма. Малое значение *PTO* приведёт к тому, что некоторые компьютеры могут не успеть отобразить свои данные на экранном модуле к назначенному времени, а слишком большое приведёт к излишнему их простою. В приведённых в статье [2] результатах тестирования максимальная рассинхронизация времени кадров находится в пределах 1-2 мсек. Однако, оба варианта этого алгоритма применимы только в том случае, если используют предположение, что новые кадры поступают на компьютеры равномерно и без потерь и не пропускаются кадры при отображении. Подобные методы применяются и в других межплатформенных программных средствах для построения распределённых систем визуализации, например, Chromium и Equalizer. Данные программные продукты предоставляют сетевые барьеры, как основной примитив синхронизации кадров при полиэкранном отображении и, следовательно, имеют те же основные недостатки, что и в алгоритме, представленном в работе [2].

**Предложенный алгоритм синхронизации.** Вданной работе не рассматривается распределение видеоданных по компьютерам кластера. Если проигрывается видеофайл, считается, что каждый компьютер получает копию видео (либо только интересующий его фрагмент видео) перед началом воспроизведения. Для потокового видео (например, трансляции с видеокамеры) считается, что кадры разбиваются на фрагменты или передаются целиком в реальном времени.

Основная идея предлагаемого в данной статье алгоритма синхронизации заключается в том, что компьютеры кластера осуществляют буферизацию входящих кадров, затем на основе текущей заполненности буфера графического процессора определяется расчётное время, в которое очередной кадр будет отображен на экране. Важно отметить, что кадровая частота выходного видеопотока при этом совпадает с частотой вертикальной развёртки экранного модуля (режим вертикальной синхронизации). Рассмотрим подробно цикл работы алгоритма для одного компьютера кластера и одного нового кадра.

Все новые кадры предварительно загружаются во входной буфер (*FIFO*-очередь). У каждого кадра из этого буфера есть временная метка $T_i$, определяющая, когда кадр должен быть отображён. Но, поскольку практически во всех современных графических приложениях используется двойная или тройная буферизация при рендеринге изображения, то даже при отправке графическому процессору команды «Отобразить» для нового кадра, он не отобразится на экране в этот момент. Сначала он поступает в один из вторичных буферов (*back buffer*) и лишь после того, как все предыдущие кадры будут отображены, его буфер станет первичным (*front buffer*), и новый кадр будет отображён на экране. Таким образом, кадр, отправленный на отображение в момент времени $t$ на самом деле бу-

дет отображён в момент времени

$$t' = t + \triangle T \cdot k + dt$$

где $\triangle T$ – длительность одного кадра (в данном случае совпадающая с периодом развёртки экранного модуля), $k$ – количество кадров в буфере графического процессора, а $dt$ – фазовая составляющая, определяющаяся разницей между временем поступления запроса на отображение и началом следующего цикла развёртки монитора. Поскольку заполненность буфера графического процессора может зависеть от текущей нагрузки и значительно варьироваться в разные моменты времени на разных компьютерах, без дополнительной коррекции время отображения кадров также будет различаться более чем на $\triangle T$. Это составляет около 16 мсек, для типичного монитора с частотой обновления 60 Гц.

Таким образом, предлагается, используя данные о времени отображения предыдущих кадров и пропорционально-интегральный регулятор (ПИ-регулятор), определять момент времени, в который новый кадр будет в действительности отображён на экране. Этот момент времени (оценочное время) вычисляется, как:

$$T'_i = T'_{i-1} + \triangle T + dt_k$$

где $T'_{i-1}$ – (время отображения предыдущего кадра),
оценка того же момента для предыдущего кадра (для первого кадра – текущее системное время узла),
$\triangle T$ – период вертикальной развёртки монитора, а $dt_k$ – корректировочный коэффициент. В процессе функционирования алгоритма хранится информация о последних $N_{hist}$ кадрах: оценочное ($T'_i$) и реальное ($T_i$) времена отображения. С использованием этой информации значение коэффициента $dt_k$ вычисляется как средняя разница между реальным и оценочным временем, умноженная на настроечный коэффициент K:

$$dt_k = K\frac{1}{N_{hist}} = \sum_{i=1}^{N_{hist}} (T_i - T'_i)$$

До тех пор, пока хотя бы один из вторичных буферов графического процессора свободен для записи, компьютер кластера принимает решение о том, какой кадр отправить на отображение следующим. Компьютер сравнивает текущее время $t$ с оценочным временем $T'_k$ первого кадра во входной очереди.

Если $t < T'_k$ или входная очередь пуста, то будет повторно выбран последний отображённый кадр, иначе – первый кадр будет удалён из очереди и отправлен на отображение.

**Программная реализация алгоритма синхронизации.** Для проверки работы алгоритма был разработан прототип системы синхронизации видеопотоков для отображения на полиэкране. За основу был взят мультимедийный фреймворк *Microsoft Media Foundation*. Поскольку его архитектура основана на слабо связанных компонентах, формирующих граф потоковой обработки медиаданных. Это позволяет разместить логику синхронизации процесса отображения в одном независимом модуле. В качестве такого модуля был создан компонент-приёмник (sink) на основе *DirectX11 API*.

В предлагаемой архитектуре среди всех распределённых компьютеров кластера один выполняет роль *master*-компьютера, который управляет воспроизведением и является источником синхронизированного времени для остальных *slave*-компьютеров (ведомых). Временная синхронизация *начала*, *паузы* и *остановки* воспроизведения медиаданных между распределёнными компьютерами кластера осуществляется путём рассылки *master*-компьютером управляющих сообщений всем *slave*-компьютерам. Синхронизация времени реализована программно согласно протоколу *PTP (Precision Time Protocol)*, что позволило достичь точности синхронизации часов на распределённых компьютерах кластера менее 1 мсек. Информация о времени отображения видеокадров, требуемая для реализации алгоритма, определяется средствами D*XGI 1.2*, включающими интерфейс *IDXGISwapChain* для доступа к очереди буферов графического процессора и метод сбора статистики кадра — *GetFrameStatistics*. Исходные данные о временах отображения кадров поступают вместе с кадрами от вышележащих компонентов графа воспроизведения через интерфейс *IMFSample*.

## Тестирование алгоритма синхронизации

Разработанный прототип системы синхронизации апробирован на системе из двух компьютеров, соединённых через локальную сеть *Ethernet* и представляющих вычислительный кластер. Один из них исполнял роль *master*-компьютера, другой – *slave*-компьютера. Тестовый видеофайл был заранее разбит на фрагменты и размещён на соответствующих компьютерах.

За основную характеристику, определяющую качество синхронизации изображения, была выбрана разница между временами воспроизведения компьютерами видеокадров, являющихся частями общего кадра изображения полиэкрана. Чем ближе эта величина к нулю, тем лучше синхронизированы изображения на отдельных мониторах.

Для проверки влияния неравномерности поступления входных кадров на работу системы синхрони-

зации была введена периодическая задержка в модуль, осуществляющий приём кадров, т.е. каждые $N$ кадров работа процесса приостанавливалась на $T$ мсек. Таким образом была смоделирована ситуация, когда один из компьютеров кластера работает с переменной задержкой относительно других, либо когда кадры изображения поступают на компьютеры неравномерно. Размер входной очереди кадров и количество вторичных буферов графического акселератора были выбраны равными четырём.

Тестовый видеофайл с частотой 30 кадров/сек. был последовательно воспроизведён в трёх вариантах: без буферизации, с буферизацией без какой-либо коррекции времени кадров и с буферизацией с коррекцией времени кадров согласно алгоритму, описанному в секции 2.2.

В первом варианте буферизация кадров при выводе на экран не использовалась. Каждый компьютер, находясь в режиме ожидания, с приходом момента времени для очередного кадра отправляет его на отображение. Как показывают результаты измерений, возникающая при задержках разница во временах кадров совпадает с величиной задержки $T$ (рис. 1).

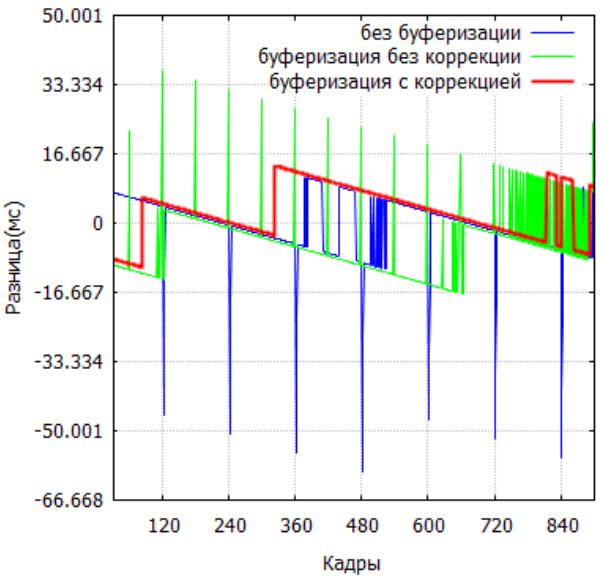


Рис. 1: Разница во времени кадров.

Во втором варианте при выводе кадров была использована буферизация с 4 вторичными буферами. Работа компьютеров осуществлялась по алгоритму, описанному в секции 2.2, за исключением того, что заполненность буфера не учитывалась при принятии решения в какой момент времени отправлять очередной кадр на отображение. В данном случае использование буферизации позволяет снизить разницу во временах воспроизведения, возникающую из-за задержек поступления кадров, до 30 мсек.

В третьем варианте, как и в предыдущем, используется буферизация кадров с теми же параметрами. В данном варианте разница находится в пределах $\pm 16$ мсек (при той же величине задержки, что и в варианте 2), что для монитора с частотой развёртки 60 Гц составляет менее одного кадра.

## Заключение

Предложенный алгоритм синхронизации распределенного процесса подготовки видеоданных для полиэкранных средств отображения позволяет обеспечить визуальную непрерывность выходного изображения без использования специализированных аппаратных средств. В отличие от аналогичных программных решений он также позволяет сохранить преимущества двойной/тройной буферизации при распределённом рендеринге и устойчив к неравномерности частоты поступления кадров, благодаря буферизации кадров на входе и на выходе. Размер буфера является настраиваемой величиной – чем он больше, тем выше устойчивость системы к задержкам в поступлении кадров, но и тем больше задержка между их выводом на полиэкран. Алгоритм не накладывает ограничений на число компьютеров в кластере, поскольку вычислительные затраты на синхронизацию на одном компьютере составляют незначительную часть (менее 1%) от общего объёма вычислений и не увеличиваются с ростом числа компьютеров.

## Литература

[1] *Deshpande S., Daly S.* Quality of Experience for Large Ultra-High-Resolution Tiled Displays with Synchronization Mismatch // EURASIP Journal on Image and Video Processing,2011. http://jivp.eurasipjournals.com/content/2011/1/647591.

[2] *Nam S., Deshpande S., Vishwanath V., Jeong B., Renambot L., Leigh J.* Multi-Application Inter-Tile Synchronization on Ultra-High-Resolution Display // Wall: http://www.mcs.anl.gov/papers/P1678.pdf

## Об авторах

Долговесов Б. С. – к.т.н., заведующий лабораторией синтезирующих систем визуализации института автоматики и электрометрии СО РАН.
e-mail: bsd@iae.nsk.su

Городилов М. А. – м.н.с. лаборатории синтезирующих систем визуализации института автоматики и электрометрии СО РАН.
e-mail: gorodilovm@gmail.com

Храмцов И. Д. – инженер-программист лаборатории синтезирующих систем визуализации института автоматики и электрометрии СО РАН.
e-mail: khramtsov9203@ya.ru