УДК 004.934

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ NVIDIA CUDA ДЛЯ ОБУЧЕНИЯ И ДЕКОДИРОВАНИЯ СКРЫТЫХ МАРКОВСКИХ МОДЕЛЕЙ

Зацепин Павел Михайлович,
канд. физ.-мат. наук, доцент, заведующий кафедрой вычислительной техники и электроники Физико-технического факультета Алтайского Государственного университета, Россия, 656049, г. Барнаул, пр. Ленина, 61.
E-mail: zpm@phys.asu.ru

Гефке Денис Алексеевич,
аспирант кафедры вычислительной техники и электроники Физико-технического факультета Алтайского Государственного университета, Россия, 656049, г. Барнаул, пр. Ленина, 61. E-mail: bspugda@mail.ru

Актуальность работы обусловлена необходимостью оптимизации алгоритмов обработки больших речевых баз данных при разработке качественных систем автоматического распознавания речи. Развитие современных многоядерных процессоров, в частности графических процессоров GPU, позволяет получить существенный прирост производительности при реализации сложных ресурсоемких алгоритмов цифровой обработки сигналов и значительно сократить время обработки данных.
Цель работы: *оптимизация алгоритмов обучения (Baum-Welch re-estimation) и декодирования (Витерби) Скрытых Марковских Моделей с помощью технологии параллельного программирования NVIDIA CUDA и оценка прироста производительности относительно центрального процессора.*
Методы исследования: *определение участков алгоритмов обучения и декодирования Скрытых Марковских Моделей, подходящих для эффективной параллельной реализации с учетом особенности программной модели CUDA, с последующей реализацией.*
Результаты: *Получена практическая параллельная реализация алгоритмов обучения и декодирования Скрытых Марковских Моделей с помощью графического процессора GPU. Произведена оценка прироста производительности относительно центрального процессора для различных параметров модели (количества состояний и размерности параметрического вектора). Результаты данной работы могут быть полезны как инженерам, работающим над созданием и улучшением систем автоматического распознавания речи, так и исследователям, работающим в области обработки сигналов и искусственного интеллекта.*

Ключевые слова:
Распознавание речи, параллельные вычисления, Скрытые Марковские Модели, NVIDIA CUDA, алгоритм Витерби, алгоритм Баума–Велча.

Введение

Математический аппарат Скрытых Марковских Моделей (СММ) представляет собой универсальный инструмент моделирования стохастических процессов, для описания которых не существует точных математических моделей, а их свойства меняются с течением времени в соответствии с некоторыми статистическими законами. Наиболее широкое применение СММ нашли при решении таких задач, как распознавание речи, анализа последовательностей ДНК и ряда других [1].

Современные системы распознавания речи предполагают наличие нескольких сотен, а то и тысяч Скрытых Марковских Моделей и их сочетаний, вследствие чего работа со СММ связана со значительными вычислительными затратами, как на этапе обучения – при обработке огромного массива речевых данных, так и при последующем декодировании – в зависимости от сложности языковой модели. Например, обучение хорошей помехозащищенной дикторо-независимой системы распознавания слитной речи может занять несколько недель, а то и месяцев. Поэтому задача оптимизации алгоритмов обработки СММ остается актуальной в настоящее время [2, 3].

Применение современных технологий параллельного программирования, в частности графических мультипроцессоров, позволяет получить значительный прирост производительности и перейти на качественно более высокий уровень в решении задач распознавания речи.

Целью данной работы является оптимизация алгоритмов обучения Скрытых Марковских Моделей (Baum-Welch re-estemation и forward-backward) и алгоритма декодирования (обобщенный алгоритм Витерби) с помощью графического процессора (CUDA) и оценка прироста производительности относительно центрального процессора (CPU).

Структура Скрытой Марковской Модели

В основе Скрытой Марковской Модели лежит конечный автомат, состоящий из **N** состояний. Переходы между состояниями в каждый дискретный момент времени t не являются детерминированными, а происходят в соответствии с некоторым вероятностным законом и описываются матрицей вероятностей переходов A_{NN}. Схематическое изображение диаграммы переходов между состояниями в СММ приведено на рис. 1 [4].

При каждом переходе в новое состояние i в момент времени t происходит генерация выходного значения x_t в соответствии с функцией распределения f_i. Результатом работы СММ является последо-

вательность параметрических векторов $\{x_1,x_2,...,x_T\}$ длиной T [5].

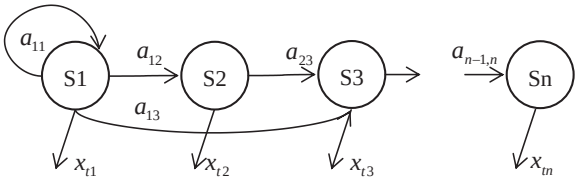

Рис. 1. *Структурная схема переходов в СММ*

На практике, как правило, решается обратная задача: при известной структуре Марковской Модели требуется определить, какова вероятность, что наблюдаемая последовательность $\{x_1,x_2,...,x_T\}$ может быть сгенерирована данной СММ [6].

Таким образом, основными параметрами СММ являются:

1) N – количество состояний;
2) матрица вероятностей переходов A_{NN} между состояниями;
3) N функций плотности вероятности $f_i(x)$.

Функция плотности вероятности $f_i(x)$ описывается, как правило, взвешенной Гауссовой смесью:

$$f(x) = \sum_{i=1}^{M} w_i p_i(x),$$

где M – количество компонент смеси; w_i – вес компонента смеси, а $p_i(x)$ – нормальное распределение вероятностей для D-мерного случая:

$$p_i(x) = \frac{1}{(2\pi)^{D/2}\left|\sigma_i\right|^{1/2}} \exp\left\{-\frac{1}{2(x-\mu_i)^T \sigma_i^{-1}(x-\mu_i)}\right\},$$

где μ_i – вектор математического ожидания; σ_i – матрица ковариации.

Работа со Скрытыми Марковскими Моделями, как и с любой другой адаптивной системой, осуществляется в 2 этапа: обучение – алгоритм Баума–Велча (Baum–Welch re-estemation), декодирование – алгоритм максимума правдоподобия (Витерби) [7].

В качестве параметрического вектора чаще всего используют мел-частотные кепстральные коэффициенты [8] (MFCC – mel-frequency cepstral coefficients), либо коэффициенты линейного предсказания [9] (LPC – linear prediction coefficients), а также их первые и вторые производные. Данные преобразования, в силу их гармонической природы, позволяют с высокой достоверностью локализовать вокализованные составляющие сигнала (гласные звуки) [10]. В тоже время, речевой сигнал имеет более сложную природу и помимо вокализованных звуков содержит ударные, шипящие и прочие составляющие. Поэтому для обработки речевого сигнала представляется перспективным применение других операторов, например Вейвлет-преобразований [11], рассмотрение которых выходит за рамки данной работы, т. к. реализованные алгоритмы являются универсальными и не зависят от природы параметрического вектора.

Сравнение Скрытой Марковской Модели и искусственных нейронных сетей при решении задачи автоматического распознавания речи

В настоящее время Скрытые Марковские Модели являются основой большинства успешных систем автоматического распознавания речи. Это связано с наличием ряда важных свойств, которыми обладают СММ по сравнению с основной альтернативной моделью классификации – искусственными нейронными сетями [12, 13]:

1) возможность моделирования длительных временных последовательностей (слов или целых высказываний), в то время как искусственные нейронные сети (ИНС) хорошо подходят для классификации кратковременных акустико-фонетических единиц, а их эффективность сильно снижается, когда на входе появляется некоторая динамика, т. е. образы подвержены, например, нелинейным изменениям во времени;
2) обучение ИНС при построении систем автоматического распознавания речи происходит, как правило, с учителем и требует точной временной разметки на фонемы обучающей выборки, что является чрезвычайно трудоемкой задачей для современных речевых баз данных. В тоже время обучение СММ может производиться без точной временной разметки;
3) возможность объединения, совместного обучения и декодирования набора СММ, представляющих отдельные фонемы, в соответствии с языковой моделью. Говоря простым языком, СММ хорошо адаптированы для построения составных моделей из исходных «примитивных» языковых элементов – фонем.

В то же время СММ обладают существенным недостатком по сравнению с ИНС – слабой дискриминантной мощностью, т. е. возможностью разделять классы образов. Особенно это проявляется при обучении с использованием критерия максимума правдоподобия. В связи с этим в последнее время исследования в области построения систем автоматического распознавания речи направлены на поиск гибридных моделей, объединяющих достоинства СММ и ИНС [14].

Рассмотрение гибридных моделей СММ/ИНС выходит за рамки данной статьи. Однако в их основе по-прежнему лежат СММ, поэтому результаты данных работы (оптимизация алгоритмов обучения и декодирования СММ с помощью GPU) могут быть успешно применены и при построении гибридных моделей.

Краткое описание алгоритма обучения Скрытой Марковской Модели

Процесс обучения Скрытой Марковской Модели заключается в определении на основе набора обучающих образцов следующих параметров:

1) матрицы вероятностей переходов между состояниями A_{NN};

2) параметров Гауссовых смесей (математическое ожидание, матрица ковариации и весового коэффициента) для каждого состояния.

Для решения этих задач совместно применяются два итерационных алгоритма: forward–backward и Baum–Welch re-estemation [2].

В алгоритме forward–backward вводятся две функции: прямого распространения вероятности $\alpha_j(t)$ и обратного $\beta_j(t)$.

Значение величины $\alpha_j(t)$ представляет собой вероятность наблюдения последовательности векторов $\{x_1,x_2,...,x_T\}$ и нахождения СММ в состоянии j в момент времени t:

$$\alpha_j(t) = P(x_1, x_2, ..., x_t \mid state_t = j). \quad (1)$$

Величины $\alpha_j(t)$ и $\alpha_j(t-1)$ связаны итерационным выражением:

$$a_j(t) = \left[\sum_{i=2}^{N-1} a_i(t-1)A_{ij}\right] f_j(x_t). \quad (2)$$

Обратная функция $\beta_j(t)$ представляет собой вероятность нахождения СММ в состоянии j в момент времени t с последующим наблюдением последовательности $\{x_{t+1}\ x_{t+2},...,x_T\}$:

$$\beta_j(t) = P(x_{t+1}, x_{t+2}, ..., x_T \mid state_t = j). \quad (3)$$

Величины $\beta_j(t)$ и $\beta_j(t+1)$ связаны аналогичным тождеством:

$$\beta_j(t) = \sum_{i=2}^{N-1} A_{ji} f_i(x_{t+1})\beta_i(t+1). \quad (4)$$

Величины $\alpha_j(t)$ и $\beta_j(t)$ позволяют определить вероятность нахождения СММ в состоянии j в момент времени t при наблюдении последовательности $\{x_{t+1},x_{t+2},...,x_T\}$:

$$L_j(t) = \frac{1}{P}\alpha_j(t)\beta_j(t),$$

где $P=\alpha_N(t)$ – общая вероятность наблюдения последовательности $\{x_{t+1},x_{t+2},...,x_T\}$ данной СММ.

Алгоритм Баума–Велча (Baum–Welch re-estemation) на очередном шаге обучения позволяет, используя вышеприведенные выражения, сделать переоценку параметров модели [2].

Пусть имеется R обучающих образцов, тогда вероятность перехода из состояния i в состояние j определяется как:

$$A_{ij} = \frac{\sum_{r=1}^{R}\frac{1}{P_r}\sum_{t=1}^{T_r-1}\alpha_i^r(t)A_{ij}f_j(x_{t+1}^r)\beta_j^r(t+1)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}L_j^r(t)}. \quad (5)$$

Для каждого состояния j и для каждой компоненты Гауссовой смеси m математическое ожидание, матрица ковариации и вес определяются следующими выражениями

$$\mu_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}L_{jm}^r(t)x_t^r}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}L_{jm}^r(t)}, \quad (6)$$

$$\sigma_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}L_{jm}^r(t)(x_t^r-\mu_{jm})(x_t^r-\mu_{jm})^T}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}L_{jm}^r(t)}, \quad (7)$$

$$w_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}L_{jm}^r(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}L_j^r(t)}. \quad (8)$$

Для качественного обучения Скрытой Марковской Модели требуется множество образцов сигнала: от нескольких десятков до нескольких сотен и тысяч экземпляров. Также необходимо соблюдать условие линейной независимости обучающих образцов, в противном случае в процессе обучения происходит вырождение матрицы ковариации, следствием чего является полная неработоспособность модели [2].

Алгоритм максимума правдоподобия (Витерби)

Суть алгоритма декодирования СММ заключается в поиске последовательности состояний, при прохождении которой наблюдаемая входная последовательность имела бы максимальную вероятность. Схема переходов и выбора цепочки состояний в момент времени t изображена на рис. 2.

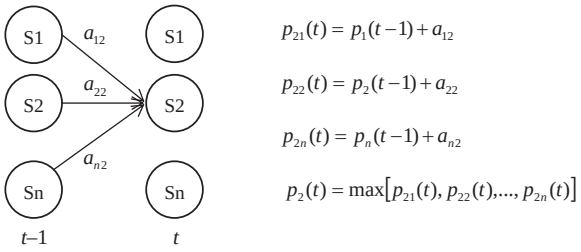

Рис. 2. *Алгоритм декодирования Витерби*

В момент времени t осуществляется переход в состояние i из *всех* предыдущих состояний, после чего выбирается последовательность, имеющая максимальную суммарную вероятность в моменты времени $t-1$ и t. Время выполнения алгоритма пропорционально длине последовательности L и квадрату количества состояний N. Алгоритм имеет сложность $O(N^2L)$.

Декодирование последовательности одной моделью не является ресурсоемкой задачей. Однако при наличии сложной языковой модели СММ объединяются вместе в виде графа, в зависимости от последовательностей следования фонем, и процесс декодирования осуществляется одновременно для множества моделей. В связи с чем алгоритмическая сложность задачи резко возрастает [3].

Особенности параллельного программирования на CUDA

Применение современных графических процессоров позволяет получить многократный прирост производительности при решении ряда научных и технических задач. Однако для достижения наилучших результатов необходимо учитывать ряд особенностей [15–18]:

1) графический процессор (GPU) состоит из нескольких мультипроцессоров, которые, в свою очередь, состоят из ядер. Каждое ядро одновременно выполняет 32 потока (warp). Например, NVidia GeForce GTX 480 состоит из 15×32=480 ядер и параллельно может выполнять до 15360 легковесных потоков. Потоки объединяются в блоки и сетки блоков. Каждый поток имеет идентифицирующие его координаты;
2) максимальной производительности удается достичь при выполнении *однотипных* действий над большим числом *обрабатываемых единиц* данных;
3) архитектура памяти имеет сложную организацию: глобальная память (объемная, но медленная), локальная память, разделяемая память (быстрая), память констант и т. д.;
4) особенности доступа к памяти: для получения максимальной пропускной способности все запросы к памяти должны быть выровнены.

Подробное описание и особенности применения графических процессоров можно найти в соответствующих руководствах [17].

Реализация декодера CMM на CUDA

Пусть N=(8,16,32) – количество состояний модели; L=50 – длина последовательности; C=32 – число одновременно декодируемых последовательностей одной CMM; N_{NN} – матрица вероятностей переходов между состояниями; B_{LN} – матрица вероятности наблюдения вектора последовательности X_L в состоянии N (C матриц рассчитываются заранее).

Для декодирования одной последовательности используются N параллельных потоков, каждый из которых декодирует свое состояние 1...N. Данная схема изображена на рис. 3. Итоговая вероятность выбирается как максимум вероятностей, полученных каждым потоком. Сложность алгоритма для одного потока – O (N×L) [19].

Один вычислительный блок CUDA одновременно декодирует C последовательностей (для одной CMM). Таким образом, для декодирования CMM из 32 состояний задействовано N×C=1024 потока, что соответствует максимальному количеству потоков на один мультипроцессор для CUDA версии 2.0. Каждый мультипроцессор используется для декодирования отдельной CMM. В итоге на GPU параллельно происходит декодирование 32×15=480 последовательностей для видеопроцессора NVidia GeForce GTX 480.

Разделяемая (быстрая) память используется для хранения матрицы вероятностей A и промежуточных вероятностей для N состояний. Объем необходимой памяти [19] – O (N^2+N×C).

Реализация алгоритма обучения CMM с помощью CUDA

Пусть R – количество обучающих образцов для одной CMM (32); N – количество состояний модели (8, 16, 32); M – количество компонент Гауссовой смеси (16); D – размерность параметрического вектора (16, 32).

Предварительный анализ показал, что в процедуре forward–backward основную долю времени (порядка 95 %), при вычислении вероятностей $\alpha_j(t)$ и $\beta_j(t)$, занимает расчет вероятности наблюдения вектора x_t в состоянии j для модели r – $f_j(x_{rt})$. В тоже время эта часть вычислений является *менее* сложной с алгоритмической точки зрения [2].

Размерность параметрического вектора D выбиралась равной warp или половине warp (количеству потоков, выполняемых ядром CUDA одновременно). Это же количество потоков (D) использовалось для реализации матричных операций при вычислении функции плотности вероятности

$$p_i(x) = \frac{1}{(2\pi)^{D/2}\left|\sigma_i\right|^{1/2}} \exp\left\{-\frac{1}{2(x-\mu_i)^T \sigma_i^{-1}(x-\mu_i)}\right\},$$

где вектора x, μ имеют размерность D, и матрица σ_i^{-1} имеет размерность D×D.

Для видеопроцессора GeForce GTX 480 каждый мультипроцессор содержит 32 ядра CUDA. Таким образом, на одном мультипроцессоре производится одновременный расчет значений функции вероятности для 32 последовательностей в одном из состояний. Вычисления производятся по всей длине последовательности – T. В свое время, каждый мультипроцессор обрабатывает одно из N состояний. Итоговая схема расчета приведена на рис. 4.

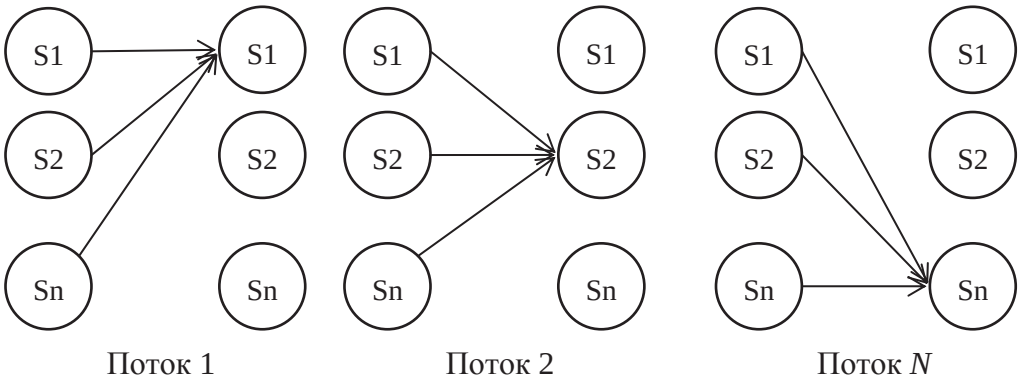

Рис. 3. *Параллельная схема декодирования*

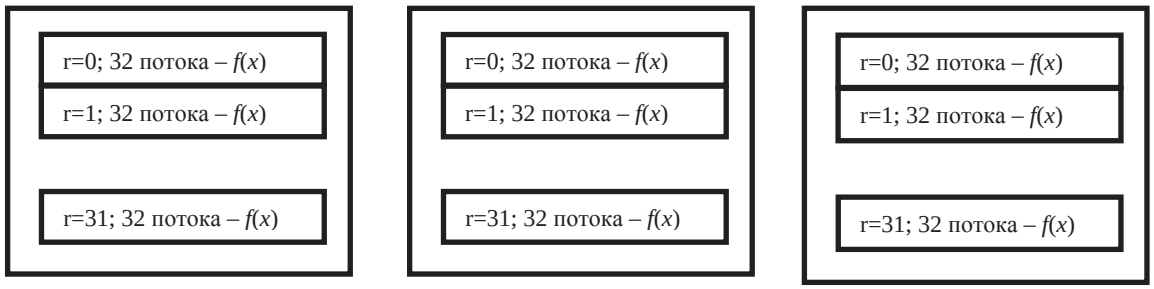

Рис. 4. *Параллельная схема вычисления значений вероятности модели во всех состояниях для всех обучающих образцов*

В результаты работы алгоритма получается массив вероятностей $f_j(x_{rt})$, который затем используется для вычисления окончательных $\alpha_j(t)$ и $\beta_j(t)$ на CPU в соответствии с итеративными формулами (1–4). В процессе вычисления одновременно задействовано 32×32×16=16384 потока.

В алгоритме Баума–Велча (формулы 5–8) схема распределения вычислений по ядрам аналогична. Векторные и матричные операции для каждого обучающего образца используют по 32 потока (размерность параметрического вектора). При этом каждый мультипроцессор обрабатывает одновременно 32 обучающих образца. После чего осуществляется редуцирование (свертка) для окончательного расчета величин математического ожидания μ_{jm} и матрицы ковариации σ_{jm}. Каждый мультипроцессор независимо обрабатывает одно из N-состояний модели.

Расчет вероятностей переходов между состояниями A_{ji} и весов компонент Гауссовых смесей w_{im} осуществляется с помощью CPU, т. к. все входящие в формулы (5) и (8) величины были предварительно рассчитаны, и для получения окончательных значений не требуются существенные вычислительные затраты [20].

Тестирование

Для тестирования алгоритма использовалась вычислительная машина следующей конфигурации:

1. Центральный процессор – Intel Core i7 930 2,8 ГГц.
2. Операционная система – Windows 7 64-Bit.
3. Оперативная память – Kingston 12 Gb RAM DDR3.
4. Видеокарта – NVidia GeForce GTX 580 (1536 Mb, 512 ядер CUDA).
5. Версия CUDA – 4.0.
6. Компилятор Microsoft Visual Studio 2008 (векторизация SSE2).
7. Тип данных – double (64 разряда).

Результаты тестирования алгоритмов обучения [19] и декодирования [20] приведены в табл. 1, 2 соответственно.

Таблица 1. *Результаты тестирования алгоритмов forward–backward и Baum–Welch re-estimation*

N	D	CPU FB, с.	GPU FB, с.	CPU/GPU FB	CPU BW, с.	GPU BW, с.	CPU/GPU BW
8	16	8,688	1,494	**5,815**	21,926	5,118	**4,284**
16	16	17,286	1,965	**8,797**	44,753	5,890	**7,598**
32	16	33,881	3,778	**8,968**	95,077	11,574	**8,215**
8	32	27,578	2,594	**10,631**	72,714	8,276	**8,786**
16	32	55,241	3,153	**17,520**	146,278	9,537	**15,338**
32	32	109,676	6,006	**18,261**	302,857	18,473	**16,395**

Таблица 2. *Результаты тестирования алгоритма декодирования CMM*

Число состояний (N)	Процессор	Количество CMM	Количество декодируемых последовательностей	Время, мс	CPU/GPU
8	CPU	1	32	1,54	50,7
		64		1,22	
16		1		5,70	41,9
		64		3,73	
32		1		2,120	58,2
		64		1,930	

Примечание к табл. 1, 2: FB – Forward–Backward. BW – Baum–Welch. CPU/GPU – отношение времени выполнения CPU к GPU. Время пересылки в память GPU включено в общее время. Наибольший прирост производительности достигается при размере параметрического вектора D=32 и N=32 состояниях модели, т. к. в этом случае задействованы все ресурсы GPU.

Выводы

Применение современных графических процессоров для реализации математического аппарата Скрытых Марковских Моделей позволяет получить существенный прирост производительности относительно центрального процессора, как на этапе обучения, так и при декодировании CMM, особенно при построении моделей с большим количеством состояний.

Однако с практической точки зрения целесообразно оптимизировать лишь те участки алгоритма, на которых выполняются ресурсоемкие *матричные* и *векторные операции*, в частности вычисление значений функции Гауссовой смеси для обрабатываемых последовательностей и расчет матрицы ковариации. Итеративная часть алгоритма for-

ward–backward (формулы 2 и 4) занимает малую часть от общего времени вычисления и сложна в реализации на GPU, поэтому ее оптимизация не представляется целесообразной. Такой же вывод можно сделать относительно расчета матрицы вероятностей переходов между состояниями и весов компонент Гауссовой смеси (формулы 5 и 8).

Повышение итоговой производительности в несколько десятков раз позволит применять более сложные Марковские Модели, а также увеличить объем материала, используемого при обучении, что представляется перспективным с точки зрения реализации более качественных систем распознавания речи.

СПИСОК ЛИТЕРАТУРЫ

1. Jurafsky D., Martin J.H. Speech and Language processing. 2nd ed. – Englewood Cliffs, New Jersey: Pretince Hall Inc., 2008. – 302 p.
2. Hidden Markov Model Toolkit Book. – Cambridge: Cambridge University Engineering Department, 2001–2009. – 399 p.
3. Rabiner L.R. A tutorial on Hidden Markov Models and selected applications in speech recognition // Proceedings of The IEEE. – 1989. – V. 77. – № 2. – P. 257–286.
4. Rabiner L.R., Juang B.H. Fundamentals of Speech Recognition. – Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1993. – 553 p.
5. Rabiner L.R., Levinson S.E. Isolated and Connected word Recognition – Theory and Selected Applications // IEEE Transactions on communications. – 1981. – V. 29. – № 5. – P. 301–315.
6. Vaseghi S.V. Advanced digital signal processing and noise reduction. 3rd ed. – Chichester, England: John Wiley & Sons, 2006. – 453 p.
7. Huandg Xuedong. Spoken language processing: a guide to theory, algorithm, and system development. – Englewood Cliffs, New Jersey: Prentice Hall Inc., 2001. – 480 p.
8. Айфичер Э.С., Джервис Б.У. Цифровая обработка сигналов: практический подход, 2-е изд. / пер. с англ. – М.: Изд. дом «Вильямс», 2004. – 992 с.
9. Vaidyanathan P.P. The Theory of Linear Prediction. – California: Morgan & Claypool, 2008. – 198 p.
10. Сорокин В.Н., Цыплихин А.И. Сегментация и распознавание гласных // Информационные процессы. – 2004. – Т. 4. – № 2. – С. 202–220.
11. Гефке Д.А., Зацепин П.М. Применение нейронных сетей для классификации сигналов звукового диапазона // Нейроинформатика, ее приложения и анализ данных: XVII Всеросс. семинар. – Красноярск, 2009. – С. 37–40.
12. Хайкин С. Нейронные сети: полный курс, 2-е изд. / пер. с англ. – М.: Изд. дом «Вильямс», 2006. – 1104 с.
13. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы / пер. с польск. И.Д. Рудинского. – М.: Телеком, 2006. – 452 с.
14. Маковкин К.А. Гибридные модели – Скрытые марковские модели/Многослойный персептрон – и их применение в системах распознавания речи // Речевые технологии. – 2012. – № 3. – С. 58–83.
15. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах. Введение в программирование графических процессоров. – М.: ДМК Пресс, 2011. – 256 с.
16. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2011. – 232 с.
17. NVIDIA CUDA SDK 4.0 // NVIDIA Corporation. URL: http://www.nvidia.com/object/cuda_sdks.html (дата обращения: 10.07.2013).
18. Гефке Д.А. Применение технологии CUDA для частотного разделение каналов широкополосного тракта // Многоядерные процессоры и параллельное программирование: Регион. научно-практ. конф. – Барнаул, 2011. – С. 12–15.
19. Гефке Д.А., Зацепин П.М. Применение технологии CUDA для декодирования Скрытых Марковских Моделей // Многоядерные процессоры, параллельное программирование, ПЛИС, системы обработки сигналов: сборник статей II регион. научно-практ. конф. – Барнаул, 2012. – С. 45–51.
20. Гефке Д.А., Зацепин П.М. Применение технологии CUDA для обучения Скрытых Марковских Моделей // Многоядерные процессоры, параллельное программирование, ПЛИС, системы обработки сигналов: сборник статей III Всеросс. научно-практ. конф. – Барнаул, 2013. – С. 30–39.

Поступила 10.07.2013 г.

UDC 004.934

NVIDIA CUDA APPLICATION TO TRAIN AND DECODE THE HIDDEN MARKOV MODELS

Pavel M. Zatsepin,
Cand. Sc., Altai State University, 61, Lenin Avenue, 656049, Barnaul, Russia.
E-mail: zpm@phys.asu.ru

Denis A. Gefke,
Altai State University, 61, Lenin Avenue, 656049, Barnaul, Russia.
E-mail: bspugda@mail.ru

The urgency of the discussed issue is caused by the need of optimization of huge speech corpus's processing algorithms required for developing robust automatic speech recognition systems. The evolution of modern multicore processors, specifically graphical processor units GPU, allows improving sufficiently the performance of difficult and resource-intensive digital signal processing algorithms and reducing sufficiently a data processing time.

The main aim of the study *is to optimize education (Baum–Welch re-estimation) and decoding (Viterbi) algorithms of Hidden Markov Models by parallel programming technology NVIDIA CUDA and to estimate performance increase in comparison within the CPU.*

The methods used in the study: *the search of education and decoding algorithm's parts suitable for effective parallel realization by NVIDIA CUDA and its implementation.*

The results: *The authors have developed parallel realization of education and decoding Hidden Markov Models algorithms by GPU and have estimated the performance increase in comparison within the CPU for different model's parameters (the number of model state and dimension of a feature vector). The results of the paper can be used both by engineers developing and improving the automatic speech recognition systems and by explorers working on a digital signal processing and artificial intelligence systems.*

Key words:
Speech recognition, parallel computing, Hidden Markov Models, NVIDIA CUDA, Viterbi algorithm, Baum-Welch re-estimation algorithm.

REFERENCES

1. Jurafsky D., Martin J.H. *Speech and Language processing*. 2nd ed. Englewood Cliffs, New Jersey: Pretince Hall Inc., 2008. 302 p.
2. *Hidden Markov Model Toolkit Book*. Cambridge, Cambridge University Engineering Department, 2001–2009. 399 p.
3. Rabiner L.R. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*. 1989, vol. 77, no. 2, pp. 257–286
4. Rabiner L.R., Juang B.H. *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1993. 553 p.
5. Rabiner L.R., Levinson S.E. Isolated and Connected word Recognition – Theory and Selected Applications. *IEEE Transactions on communications*, 1981. vol. 29, no. 5, pp. 301–315.
6. Vaseghi S.V. *Advanced digital signal processing and noise reduction*. 3rd ed. Chichester, England, John Wiley & Sons, 2006. 453 p.
7. Huandg Xuedong. *Spoken language processing: a guide to theory, algorithm, and system development*. Englewood Cliffs, New Jersey, Prentice Hall Inc., 2001. 480 p.
8. Ifeachor E.C., Jervis B.W. *Digital Signal Processing. A practical approach*. 2nd ed. Englewood Cliffs, New Jersey, Pretince Hall Inc., 2004. 992 p.
9. Vaidyanathan P.P. *The Theory of Linear Prediction*. California, Morgan & Claypool, 2008. 198 p.
10. Sorokin V.N., Tsyplikhin A.I. Segmentatsiya i raspoznovanie glasnykh [Segmentation and vocal recognition]. *Informatsionnye protsessy – Informational processes*, 2004, vol. 4, no. 2, pp. 202–220.
11. Gefke D.A., Zatsepin P.M. Primenenie neyronnykh setey dlya klassifikatsii signalov zvukovogo diapazona [The application of neural networks for voice signal classification]. *Neiroinformatika, ee primenenie i analiz dannykh. XVII Vserossiysky seminar* [Neuroinformatic, its application and data analyze. XVII Russian conference]. Krasnoyarsk, 2009. pp. 37–40.
12. Haykin S. *Neural Networks – A Comprehensive Foundation*. 2nd ed. New Jersey, Pearson Education Inc., 1999. 823 p.
13. Rutkovskaya D., Pilinskiy M., Rutkovskiy L. *Neironnie sistemy, geneticheskie algoritmy i nechetkie sistemy* [Neural networks, genetic algorithms and fuzzy systems]. Moscow, Telekom, 2006. 452 p.
14. Makovkin K.A. Gibridnye modeli – Skrytye Markovskie Modely/ Mnogosloyny pertseptron – i ikh primenenie v sistemakh avtomaticheskogo raspoznavaniya rechi [The Hybrid models – Hidden Markov Models/Multi-Layer Perceptron – and its application in automatic speech recognition systems]. *Rechevye tekhnologii – Speech Technologies*, 2012, no. 3, pp. 58–83.
15. Sanders J., Kandrot E. *CUDA by Example: An Introduction to General-Purpose GPU Programming Code*. Boston, Addison-Wesley Professional, 2010. 312 p.
16. Boreskov A.V., Harlamov A.A. *Osnovy raboty s CUDA* [Basic usage of CUDA]. Moscow, DMK Publ., 2011. 232 p.
17. NVIDIA CUDA SDK 4.0. *NVIDIA Corporation*. Avaible at: http://www.nvidia.com/object/cuda_sdks.html (accessed 10 July 2013).
18. Gefke D.A., Zatsepin P.M. Primenenie tekhnology CUDA dlya dekodirovaniya Skrytykh Markovskikh Modeley [The application of CUDA technology for decoding Hidden Markov Models]. *Mnogoyadernye protsessory, parallelnoe programmirovanie, PLIS, sistemy obrabotki signalov. Sbornik statey III Regionalnoy konferentsii* [Multicore processors, parallel programming, PLD, digital signal processing systems. Paper collection of the III Regional science-practical conference]. Barnaul, 2012. pp. 45–51.
19. Gefke D.A., Zatsepin P.M. Primenenie tekhnology CUDA dlya obucheniya Skrytykh Markovskikh Modeley [The application of CUDA technology for education of Hidden Markov Models]. *Mnogoyadernye processory, parallelnoe programmirovanie, PLIS, sistemy obrabotki signalov. Sbornik statey III Vserossiyskoy konferentsii* [Multicore processors, parallel programming, PLD, digital signal processing systems. Paper collection of the III Russian science-practical conference]. Barnaul, 2013. pp. 30–39
20. Gefke D.A. Primenenie tekhnologii CUDA dlya chastotnogo razdeleniya kanalov shirokopolosnogo trakta [The application of CUDA technology for frequency-separation of wideband signal]. *Mnogoyadernye protsessory i parallelnoe programmirovanie. Regionalnaya nauchno-prakticheskaya konferentsiya* [Multicore processors and parallel programming. Regional science-practical conference]. Barnaul, 2011. pp. 12–15.