Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет
"Высшая школа экономики"

Факультет компьютерных наук
Департамент программной инженерии"

ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Параллельное программирование

Разработчик программы
Аветисян А.И., член-корр. РАН, проф., д.ф.-.м.н. arut@ispras.ru

Одобрена на заседании департамента программной инженерии «___»_____ 2017 г.
Руководитель департамента Авдошин С.М. _____

Утверждена Академическим советом образовательной программы «___»_____ 2017 г.,
№ протокола_____

Академический руководитель образовательной программы Петренко А.К. _____

Москва, 2017

1 Область применения и нормативные ссылки

Настоящая программа учебной дисциплины устанавливает минимальные требования к знаниям и умениям студента и определяет содержание и виды учебных занятий и отчетности. Программа предназначена для преподавателей, ведущих данную дисциплину, учебных ассистентов и студентов образовательной программы «Системное программирование» направления подготовки 09.03.04 «Программная инженерия», изучающих дисциплину "Параллельное программирование". Программа разработана в соответствии с образовательным стандартом Национального исследовательского университета «Высшая школа экономики» по направлению 09.03.04 «Программная инженерия»

2 Цели освоения дисциплины

Цель курса – Целью курса является формирование у студентов теоретических знаний и навыков разработки, исследования производительности, оптимизации и отладки параллельных программ для современных архитектур процессоров и графических акселераторов.

Задачами данного курса являются:

- освоение студентами базовых современных достижений в технологиях центральных процессоров и акселераторов, языковых моделей параллельного программирования и их поддержке в компиляторах;
- формирование практических навыков написания кода, использующего различные методы распараллеливания, поиска ошибок и неэффективных участков.

3 Компетенции обучающегося, формируемые в результате освоения дисциплины

В результате освоения дисциплины студент должен:

1. Знать:

- фундаментальные понятия и основные законы параллельного программирования;
- современные проблемы соответствующих разделов системного программирования;
- основные трудности параллельного программирования, причины их возникновения и пути их преодоления;
- основные свойства соответствующего системного программного обеспечения;
- классы параллельных архитектур и особенности разработки параллельных программ для этих классов.

2. Уметь:

- понять поставленную задачу;
- использовать свои знания для разработки эффективных параллельных программ;
- оценивать корректность программы, ее масштабируемость;
- самостоятельно проектировать, реализовывать, отлаживать и сопровождать высокоэффективные параллельные программы.

3. Иметь навыки (приобрести опыт):

- навыками освоения большого объема информации и решения задач параллельного программирования (в том числе, сложных);
- навыками самостоятельной работы и освоения новых дисциплин;
- культурой постановки, анализа и решения математических и прикладных задач, требующих для своего решения использования математических подходов и методов параллельного программирования.

В результате освоения дисциплины студент должен обладать следующими компетенциями:

Системные компетенции:

- Способен рефлексировать (оценивать и перерабатывать, анализировать и синтезировать) освоенные научные методы и способы деятельности для применения на практике (СК-М1).
- Способен предлагать концепции, модели, создавать и апробировать новые способы и инструменты профессиональной деятельности для применения на практике (СК-М2).
- Способен к самостоятельному освоению новых методов исследований, изменению научного и производственного профиля своей деятельности (СК-М3).
- Способен анализировать, верифицировать, оценивать полноту информации, найденной и полученной из различных источников в ходе профессиональной деятельности, при необходимости восполнять и синтезировать недостающую информацию (СК-М6).

Инструментальные компетенции:

- Способен проводить анализ, синтез, оптимизацию решений с целью обеспечения качества объектов профессиональной деятельности (ИК-М1.2.НИД (ПИ)).
- Способен планировать, управлять и контролировать выполнение требований (ИК-М2.1.АД (ПИ)).
- Способен выполнять проектную деятельность в области программной инженерии на основе системного подхода , уметь строить и использовать модели для описания и прогнозирования различных явлений, осуществлять их качественный и количественный анализ (ИК-М3.1.ПД (ПИ)).
- Способен оценить и выбрать методологию проектирования объектов профессиональной деятельности (ИК-М3.3.ПД (ПИ)).
- Способен применять современные технологии разработки программных комплексов с использованием автоматизированных систем планирования и управления, осуществлять контроль качества разрабатываемых программных продуктов (ИК-М4.1.ПТД_ПИ2 (ПИ)).

4 Место дисциплины в структуре образовательной программы

Изучение данной дисциплины базируется на знаниях, полученных студентами при освоении учебных дисциплин:

- «Программирование»,
- «Информатика, математическая логика и теория алгоритмов»,
- «Построение и анализ алгоритмов»,
- «Архитектура вычислительных систем»,
- «Операционные системы».

5 Тематический план учебной дисциплины

№	Название раздела	Всего часов	Аудиторные часы		Самостоятельная работа
			Лекции	Практические занятия	
1.	Уровни параллелизма в современных	16	2	2	12

	компьютерах. Теоретические подходы: законы Амдаля, Густафсона. Оценки пиковой производительности. Memory wall. Performance/portability tradeoff.				
2.	Параллелизм в пределах одного контекста выполнения. Параллелизм на уровне команд. VLIW. SIMD. Delayed branches. Компиляторные преобразования повышающие ILP. Modulo scheduling.	24	2	2	20
3.	Векторный параллелизм (SIMD). Ограничения компиляторного анализа. Возможности и ограничения явной векторизации через интринсики. Компромисс между производительностью и переносимостью. Структурирование кода для облегчения компиляторного анализа.	14	2	2	10
4.	Вычисления с плавающей точкой. Принципы IEEE-754. Ограничения и возможности компиляторной трансляции.	20	2	2	16
5.	Иерархия кешей. Оптимизация доступа к памяти. Префетчинг. Cache-aware и cache-oblivious алгоритмы. Возможные компиляторные оптимизации.	26	2	4	20
6.	Параллелизм на многоядерных CPU. Кеши и поддержка согласованности памяти. False sharing.	22	2	4	16
7.	Низкоуровневые примитивы межконтекстного взаимодействия: атомарные операции, семафоры, futex. Поддержка на уровне языка Си и особенности компиляторной поддержки.	18	2	4	12
8.	OpenMP. Классическая функциональность: параллелизм на уровне нитей. Анализ и трансляция OpenMP в компиляторе.	26	2	4	20
9.	Анализ производительности на CPU. Основные инструменты разработчика: valgrind (cachegrind), perf, ocperf. Использование компиляторных возможностей.	6	2	4	
10.	Параллелизм на графических акселераторах. Явные интерфейсы программирования: CUDA, OpenCL. Инструменты разработки в CUDA.	6	2	4	
11.	Оптимизация в CUDA. Оптимизация доступа к памяти. Компромисс между ILP, регистровым давлением, TLP. Warp-synchronous programming. Сравнение с подходами, применяемыми на CPU.	6	2	4	

12.	OpenACC и OpenMP 4.0: параллелизм для акселераторов. Подходы к трансляции кода в различных реализациях. Специализация OpenMP-кода для акселераторов.	6	2	4	
	ИТОГО:	190	24	40	126

6 Формы контроля знаний студентов

Тип контроля	Форма контроля	Модуль				Параметры
		1	2	3	4	
Текущий	Контрольная работа	*				Письменная работа 60 минут
	Домашнее задание	*				Сдача не позднее, чем за 15 дней до экзамена
Итоговый	Экзамен		*			Устный экзамен.

Критерии оценки знаний, навыков

В рамках курса слушателям предлагается выполнить контрольную работу, домашнее задание, сдать промежуточный и итоговый экзамены. Оценки за контрольную работу, за домашнее задание и за каждый экзамен выставляются по 10-ти балльной шкале.

Порядок формирования оценок по дисциплине

Оценка за курс складывается из оценок за выполнение контрольной работы $O_{конт}$ (10 баллов), домашнего задания $O_{дом}$ (10 баллов) и оценки за итоговый экзамен $O_{экз}$ (10 баллов).
В диплом выставляется результирующая оценка по учебной дисциплине, которая формируется по следующей формуле:

$$O_{результ} = 0,3*O_{конт} + 0,3*O_{дом} + 0,4*O_{экз}$$

7 Содержание дисциплины

1. Уровни параллелизма в современных компьютерах. Теоретические подходы: законы Амдаля, Густафсона. Оценки пиковой производительности. Memory wall. Performance/portability tradeoff.

2. Параллелизм в пределах одного контекста выполнения. Параллелизм на уровне команд. VLIW. SIMD. Delayed branches. Компиляторные преобразования повышающие ILP. Modulo scheduling.

3. Векторный параллелизм (SIMD). Ограничения компиляторного анализа. Возможности и ограничения явной векторизации через интринсики. Компромисс между производительностью и переносимостью. Структурирование кода для облегчения компиляторного анализа.

4. Вычисления с плавающей точкой. Принципы IEEE-754. Ограничения и возможности компиляторной трансляции.

5. Иерархия кешей. Оптимизация доступа к памяти. Префетчинг. Cache-aware и cache-oblivious алгоритмы. Возможные компиляторные оптимизации.

6. Параллелизм на многоядерных CPU. Кеши и поддержка согласованности памяти. False sharing.

7. Низкоуровневые примитивы межконтекстного взаимодействия: атомарные операции, семафоры, futex. Поддержка на уровне языка Си и особенности компиляторной поддержки.

8. OpenMP. Классическая функциональность: параллелизм на уровне нитей. Анализ и трансляция OpenMP в компиляторе.

9. Анализ производительности на CPU. Основные инструменты разработчика: valgrind (cachegrind), perf, ocperf. Использование компиляторных возможностей.

10. Параллелизм на графических акселераторах. Явные интерфейсы программирования: CUDA, OpenCL. Инструменты разработки в CUDA.

11. Оптимизация в CUDA. Оптимизация доступа к памяти. Компромисс между ILP, регистровым давлением, TLP. Warp-synchronous programming. Сравнение с подходами, применяемыми на CPU.

12. OpenACC и OpenMP 4.0: параллелизм для акселераторов. Подходы к трансляции кода в различных реализациях. Специализация OpenMP-кода для акселераторов.

8 Оценочные средства для текущего контроля и аттестации студента

8.1 Примеры заданий для контрольной работы

Напишите функцию, вычисляющую индекс максимального элемента в массиве с эффективным использованием векторных инструкций.

Укажите ошибки работы с общими данными в приведенном OpenMP-коде.

Предложите возможные способы повышения производительности приведенной программы, использующей интерфейс CUDA.

8.2 Примеры контрольных вопросов для экзамена

1. Параллелизм в пределах одного процессорного ядра и контекста выполнения: архитектурные возможности и компиляторные подходы к повышению параллелизма на уровне команд.

2. SIMD-расширения: поддержка в ОС и компиляторах, доступные возможности использования (ассемблер, расширения языка, автовекторизация).

3. Вычисления с плавающей точкой. Принципы IEEE-754. Ограничения и возможности компиляторной трансляции.

4. Организация кеш-памяти. Автоматический и явный префетчинг. Cache-aware и cache-oblivious алгоритмы. Инструмент pahole.

5. Поддержка согласованности кешей на многоядерных процессорах. Эффект false sharing.

6. Низкоуровневые средства взаимодействия параллельных потоков: атомарные операции, futex. Поддержка атомарных операций в языке Си.

7. OpenMP. Основные принципы (fork-join параллелизм на общей памяти, аннотации кода в виде прагм). Компиляторная трансляция OpenMP-конструкций.

8. Профилирование программ: через инструментирование кода, на модельном процессоре (Cachegrind, Callgrind). Профилирование с помощью аппаратных счетчиков.

9. Вычисления на графических акселераторах. Иерархия параллелизма вычислительных модулей и ее отражение в модели программирования.

9 Учебно-методическое и информационное обеспечение дисциплины

Необходимое программное обеспечение: дистрибутив Linux с инструментами разработки (GCC, GDB, linux-perf, Valgrind), пакет CUDA Toolkit.

Основная литература.

1. Paul McKenney. Is Parallel Programming Hard, And, If So, What Can You Do About It?

Дополнительная литература.

2. William Kahan. Why do we need a floating-point arithmetic standard?
3. David Kirk, Wen-mei Hwu. Programming Massively Parallel Processors, Second Edition: A Hands-on Approach
4. David Paterson. Computer Architecture, Fifth Edition: A Quantitative Approach.
5. Calvin Lin, Lawrence Snyder. Principles of Parallel Programming, 1st Edition.
6. Brendan Gregg. Systems Performance: Enterprise and the Cloud, 1st Edition.
7. Barbara Chapman, Gabriele Jost, Ruud van der Pas. Using OpenMP: Portable Shared Memory Parallel Programming.
8. Michael Quinn. Parallel Programming in C with MPI and OpenMP.
9. Randy Allen, Ken Kennedy. Optimizing Compilers for Modern Architectures: A Dependence-based Approach, 1st Edition.