ИТМО

# Высокопроизводительные вычисления с помощью HIP

# Серия «Суперкомпьютерное Образование», Издательство «МАКС Пресс»

ИТМО


# Язык программирования HIP

- Комплекс параллельного программирования с открытым исходным кодом
- Включает в себя:
  - Среду выполнения программного интерфейса API C++
  - Базовый язык и компилятор
  - Инструментарий
  - Прикладные библиотеки
- Поддерживает выполнение программ на платформах GPU от AMD и Nvidia (в отличие от CUDA)
- Работает в среде выполнения ROCm и ОС Windows
- Широкий набор библиотек: hipBLAS, hipFFT, hipRAND, hipSPARSE, . . . . . .

# Мотивация создания этого учебника

ИТМО

- Отсутствие книг по программированию на языке HIP
- Помощь программистам в использовании вычислительных мощностей графических процессоров AMD
- Представление профессиональному сообществу языка программирования, который может работать на графических процессорах от разных производителей (в отличие от CUDA)
- Рассмотрение в деталях концепций построения аппаратных и программных средств, являющихся ключевыми для обеспечения высокой производительности
- Предоставление широкого набора примеров программирования для разных классов приложений
- Информирование о возможностях интегрированных утилит ROCm и инструментов сторонних разработчиков

# Введение в программирование с использованием HIP

ИТМО

- **HIP — Heterogeneous Interface for Portability или интерфейс программирования гетерогенных вычислений для переносимости (представлен в 2016 году)**
- Приоритетный язык для запуска параллельных приложений на графических процессорах AMD
- HIP предоставляет широкий набор утилит API для взаимодействия и запуска вычислительных задач на GPU
- Синтаксис HIP очень похож на OpenCL и CUDA, что упрощает освоение программистами, использующими GPU
- Среда выполнения HIP API обеспечивает богатый набор сервисов, включающий:

  hipMalloc, hipMemcopy, hipDeviceSynchronize, hipFree, … …

# Стек программного обеспечения ROCm

ИТМО

**Фреймворки**
TensorFlow, PyTorch, Kokkos

**Библиотеки**
MIOpen, библиотеки roc*

**Модели программирования**
HIP, OpenCL , OpenMP

**Промежуточные среды выполнения/компиляторы**
Clang на основе LLVM (HIP-Clang)

**Среды выполнения**
ROCm

**Программные и системные инструменты**
- отладка
- профилирование

# Структура пособия по программированию на языке HIP

304 страницы

- 11 глав и 3 приложения
- Предисловие к изданию на русском языке
- Вступительное слово профессора Джека Донгарра
- Содержит описание интерфейса программирования HIP и синтаксиса программирования на HIP
- Описывает архитектуру графических процессоров AMD (MI100) и код ассемблера CDNA
- Приводится детальная информация о библиотеках и инструментарии ROCm, разработанных в AMD
- Добротное описание инструментов среды ROCm от сторонних разработчиков

# Содержание (1)



# Содержание (2)



# HIP: Использование нескольких GPU

ИТМО

- Системы с множеством GPU преобладают в гетерогенных вычислительных кластерах классических суперкомпьютеров и систем искусственного интеллекта
- Один графический процессор в вычислительной системе не может удовлетворить требования к производительности современных высокопараллельных приложений
- ROCm и HIP обеспечивают возможности программирования  и запуска приложений на системах с несколькими GPU без каких-либо дополнительных усилий и инструментов


# Общий подход к распараллеливанию задач в HIP

ИТМО

- Использование функций HIP API для управления несколькими GPU и диспетчеризации задач в параллельном режиме
- Существуют и поддерживаются несколько подходов:
  - Использование потоков HIP
  - Использование тредов CPU
  - Использование MPI
  - Использование RCCL
- Программист может комбинировать эти подходы в одной задаче:
  - Например, использовать два треда CPU для управления четырьмя GPU и использовать потоки HIP внутри каждого GPU
- В учебном пособии описаны базовые концепции программирования систем с несколькими GPU, чтобы дать возможность программистам начать работать на такого рода системах

# Использование потоков для программирования нескольких GPU

ИТМО

- Ключевая идея: Задачи распределяются в разные потоки, которые могут выполняться параллельно
- Базовый алгоритм:

1. Создать поток для каждого GPU
2. Выполнить резевирование памяти и копирование данных, если это необходимо
3. Запустить асинхронно программные ядра в потоках
4. Синхронизировать вычисления в конце или где необходимо

- Пример кода в учебном пособии: Выполнение сложения векторов с несколькими GPU, где каждый GPU работает с собственным фрагментом входных данных

# Сложение векторов с использованием потоков и нескольких GPU (1)

ИТМО

```
// Each GPU works on "length_per_gpu" elements of data
for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  // Create streams
  // Allocate memory on GPU
  // Copy data from the host to the memory allocated on the GPU
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  vec_add<<<num_block_per_gpu, block_size, 0, streams i>>>(
      d_a[i], d_b[il, d_c[i], length_per_gu);
}
for (int i = 0; i < num_gpus; i++){
  hipStreamSynchronize(streams[i]);
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  hipMemcpy(..)
}
```

Резервирование буферов в нескольких GPU
Назначение потоков на GPUs

# Сложение векторов с использованием потоков и нескольких GPU (2)

```
// Each GPU works on "length_per_gpu" elements of data
for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  // Create streams
  // Allocate memory on GPU
  // Copy data from the host to the memory allocated on the GPU
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  vec_add<<<num_block_per_gpu, block_size, 0, streams i>>>(
      d_a[i], d_b[il, d_c[i], length_per_gu);
}
for (int i = 0; i < num_gpus; i++){
  hipStreamSynchronize(streams[i]);
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  hipMemcpy(..)
}
```

Резервирование буферов в нескольких GPU
Назначение потоков на GPUs

Запуск программных ядер в каждом GPU через потоки без синхронизации

# Сложение векторов с использованием потоков и нескольких GPU (3)

ИТМО

```
// Each GPU works on "length_per_gpu" elements of data
for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  // Create streams
  // Allocate memory on GPU
  // Copy data from the host to the memory allocated on the GPU
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  vec_add<<<num_block_per_gpu, block_size, 0, streams i>>>(
      d_a[i], d_b[il, d_c[i], length_per_gu);
}
for (int i = 0; i < num_gpus; i++){
  hipStreamSynchronize(streams[i]);
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  hipMemcpy(..)
}
```

Резервирование буферов в нескольких GPU
Назначение потоков на GPUs

Запуск программных ядер в каждом GPU через потоки без синхронизации

Синхронизация всех потоков

# Сложение векторов с использованием потоков и нескольких GPU (4)

ИТМО

```
// Each GPU works on "length_per_gpu" elements of data
for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  // Create streams
  // Allocate memory on GPU
  // Copy data from the host to the memory allocated on the GPU
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  vec_add<<<num_block_per_gpu, block_size, 0, streams i>>>(
      d_a[i], d_b[il, d_c[i], length_per_gu);
}
for (int i = 0; i < num_gpus; i++){
  hipStreamSynchronize(streams[i]);
}

for (int i = 0; i < num_gpus; i++) {
  hipSetDevice(i);
  hipMemcpy(..)
}
```

Резервирование буферов в нескольких GPU
Назначение потоков на GPUs

Запуск программных ядер в каждом GPU через потоки без синхронизации

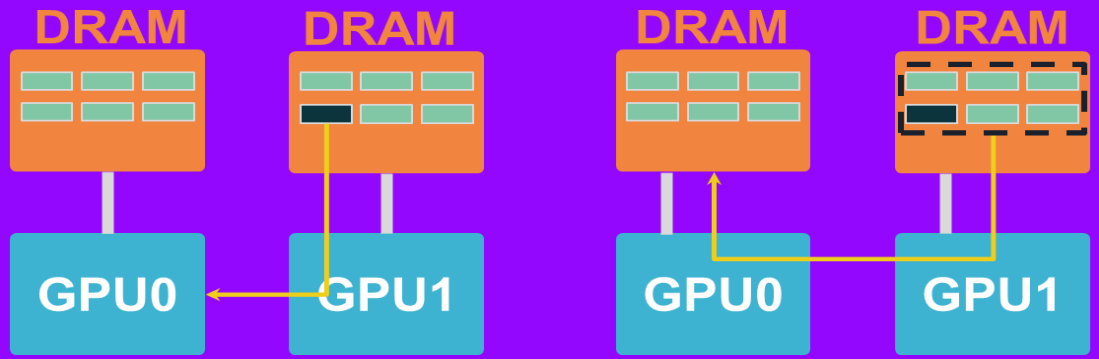
Синхронизация всех потоков

Копирование данных из памяти каждого GPU

# Передача данных в системе с несколькими GPU

ИТМО

- ROCm поддерживает разные способы передачи данных между GPU
- Главные способы включают:
  - Явные операции копирования данных в памяти от одного GPU к другому с использованием hipMemcpy
  - Этот способ может включать либо прямой одноранговый доступ (P2P), либо через память хоста используя host memcopies
  - Прямой доступ к кэш-памяти
  - Поддержка передачи данных через унифицированную память (не описано в первом издании, но включено во второе издание с ROCm 6.x)



# Копирование памяти в сравнении с прямым доступом к кэш-памяти

ИТМО

Хорошо  Средне  Затруднительно

| Режим | Доступ к малораз-мерным данным | Использо-вание пропускной способности | Легкость програм-мирования | Совмещение передачи данных и вычислений |
|---|---|---|---|---|
| Копирование памяти | | | | |
| Прямой доступ к кэш-памяти | | | | |

# Копирование памяти в сравнении с прямым доступом к кэш-памяти

ИТМО

Хорошо Средне Затруднительно

| Режим | Доступ к малоразмерным данным | Использование пропускной способности | Легкость программирования | Совмещение передачи данных и вычислений |
|---|---|---|---|---|
| Копирование памяти | | | | |
| Прямой доступ к кэш-памяти | | | | |

Учебное пособие включает примеры кода по использованию этих способов при передаче данных между GPU

# ROCm Collective Communications Library (RCCL)

ИТМО

- RCCL является библиотекой для передачи данных в системе с несколькими GPU
  - Интерфейс в стиле MPI

Общепринятые примитивы для передачи данных между GPU: All Reduce, Gather, Reduce, Broadcast
  - Можно использовать в комбинации с MPI

- Библиотека учитывает топологию связности и оптимизирована для серии GPU AMD Instinct
- В пособии есть примеры использования RCCL:
  - Передать одни и те же данные нескольким GPU (Broadcast)
  - Вычислить среднее арифметическое результатов, вычисленных несколькими GPU (All Reduce)

# Пример с использованием RCCL + MPI (1)

ИТМО

```
ncclUniqueId id;
ncclComm_t comm;
float *sendbuff, *recvbuff;
hipStream_t s;

if (myRank == 0) ncclGetUniqueId(&id);
MPI_Bcast((void*)&id, sizeof(id), MPI_BYTE, 0, MPI_COMM_WORLD);

//Allocate buffers and create streams
hipSetDevice(myRank);
hipMalloc(&sendbuff, size * sizeof(float));
hipMalloc(&recvbuff, size * sizeof(float));

// Initializing RCCL
hipStreamCreate(&s);
ncclCommInitRank(&comm, nRanks, id, myRank);

// communicating using RCCL
ncclAllReduce((const void*)sendbuff, (void*)recvbuff, size, ncclFloat, ncclSum,
comm, s);

// Completing RCCL operation by synchronizing on the HIP stream
hipStreamSynchronize(s);
```

Резервирование буферов и другие декларации

Передача идентификаторов от ранга 0 ко всем остальным рангам

# Пример с использованием RCCL + MPI (2)

ИТМО

```
ncclUniqueId id;
ncclComm_t comm;
float *sendbuff, *recvbuff;
hipStream_t s;

if (myRank == 0) ncclGetUniqueId(&id);
MPI_Bcast((void*)&id, sizeof(id), MPI_BYTE, 0, MPI_COMM_WORLD);

//Allocate buffers and create streams
hipSetDevice(myRank);
hipMalloc(&sendbuff, size * sizeof(float));
hipMalloc(&recvbuff, size * sizeof(float));

// Initializing RCCL
hipStreamCreate(&s);
ncclCommInitRank(&comm, nRanks, id, myRank);

// communicating using RCCL
ncclAllReduce((const void*)sendbuff, (void*)recvbuff, size, ncclFloat, ncclSum,
comm, s);

// Completing RCCL operation by synchronizing on the HIP stream
hipStreamSynchronize(s);
```

Резервирование буферов и другие декларации

Передача идентификаторов от ранга 0 ко всем остальным рангам

Создание потоков и инициализация передачи данных

# Пример с использованием RCCL + MPI (3)

ИТМО

```
ncclUniqueId id;
ncclComm_t comm;
float *sendbuff, *recvbuff;
hipStream_t s;

if (myRank == 0) ncclGetUniqueId(&id);
MPI_Bcast((void*)&id, sizeof(id), MPI_BYTE, 0, MPI_COMM_WORLD);

//Allocate buffers and create streams
hipSetDevice(myRank);
hipMalloc(&sendbuff, size * sizeof(float));
hipMalloc(&recvbuff, size * sizeof(float));

// Initializing RCCL
hipStreamCreate(&s);
ncclCommInitRank(&comm, nRanks, id, myRank);

// communicating using RCCL
ncclAllReduce((const void*)sendbuff, (void*)recvbuff, size, ncclFloat, ncclS
comm, s);

// Completing RCCL operation by synchronizing on the HIP stream
hipStreamSynchronize(s);
```

Резервирование буферов и другие декларации

Передача идентификаторов от ранга 0 ко всем остальным рангам

Создание потоков и инициализация передачи данных

Выполнение операций RCCL по передаче данных и последующая синхронизация

# Поддержка развития HIP

ИТМО

- Первая книга по программированию на HIP
  - Рост сообщества программистов, использующих HIP
  - Расширение экосистемы открытого кода ROCm (инструменты и библиотеки)
  - Распространение знаний об архитектуре графических процессоров AMD и гетерогенных вычислениях
  - Продвижение технологий настройки и повышения производительности вычислений в суперкомпьютерах и системах искусственного интеллекта
  - Учебник доступен в печатном и электронном виде
- Поддержка преподавания HIP в университетах и техникумах
  - Англоязычные слайды в разработке и будут доступны https://www.amd.com/en/corporate/university-program.html
  - Видеолекции от авторов на английском языке https://www.youtube.com/playlist?list=PLB1fSi1mbw6IKbZSPz9a2r2DbnHWnLbF-
  - Готовы сотрудничать в создании русскоязычной версии слайдов и видео

# Дальнейшее развитие проекта

- Начался перевод учебного пособия на китайский язык с отдельными изданиями в Шанхае и Тайбэе
- Второе издание «Accelerated Computing with HIP» готовится к печати и будет опубликовано в четвертом квартале 2024 г.
  - Содержит модель архитектуры и производительности GPU с детальной информацией по новым процессорам MI200/MI300
  - Новая версия ROCm 6.x, включающая новые инструменты и библиотеки
  - Добавлены новые инструменты от сторонних разработчиков, интегрированные с ROCm
- Подготовка перевода второго издания на русский язык может быть начата уже в декабре 2024 г.
- Нужны российские спонсоры этого проекта, требуется примерно $10K+ для оплаты работы переводчика, издательства и базового тиража
  - Первое издание является результатом сотрудничества с DXC Luxoft, сотрудник которого обеспечил качественный перевод и подготовку издания

# Спасибо
# за внимание!

ITMO more than a UNIVERSITY