Перенос вычислений на акселераторы NVIDIA в компиляторе GCC

Александр Монаков
amonakov@ispras.ru

Владислав Иванишин
vlad@ispras.ru

Евгений Кудряшов
kudryashov@ispras.ru

Институт системного программирования РАН

Языки программирования и компиляторы 2017

OpenMP

Прагмы и API для параллелизма на общей памяти

Пример

```
#pragma omp parallel for reduction(+:sum)
  for (i = 0; i < n; i++)
    sum += x[i] * y[i];
```

Прагмы

- parallel
- for, sections, task
- critical, barrier, atomic
- simd, ...

Клаузы

- shared, private, default
- reduction, lastprivate
- schedule, nowait
- if, ...

API

`omp_get_thread_num`, `omp_set_lock`, `omp_get_wtime`, ...

OpenMP Lowering

Пример

```
#pragma omp parallel
{
  int i;
#pragma omp for reduction(+:sum)
  for (i = 0; i < n; i++)
    sum += x[i] * y[i];
}
```

```
struct omp_ds_s {
  double *sum;
  int n;
  double *x, *y;
} omp_ds_o = {&sum, n,
GOMP_parallel(omp_fn_1,
```

```
void omp_fn_1(void *dat
{
  struct omp_ds_s *omp_
  int i;
  double sum;
  int thrI = omp_get_th
  int thrN = omp_get_nu
```

OpenMP Lowering

Пример

```
#pragma omp parallel
{
  int i;
#pragma omp for reduc
  for (i = 0; i < n;
    sum += x[i] * y[i
}
```

```
struct omp_ds_s {
  double *sum;
  int n;
  double *x, *y;
} omp_ds_o = {&sum, n, x, y};
GOMP_parallel(omp_fn_1, omp_ds_o);
```

```
void omp_fn_1(void *data)
{
  struct omp_ds_s *omp_ds_i = data;
  int i;
  double sum;
  int thrI = omp_get_thread_num();
  int thrN = omp_get_num_threads();
```

OpenMP Lowering

```
GOMP_parallel(omp_fn_1, omp_ds_o);
```

Пример

```
#pragma omp parallel
{
  int i;
#pragma omp for reduc
  for (i = 0; i < n;
    sum += x[i] * y[i
}
```

```
void omp_fn_1(void *data)
{
  struct omp_ds_s *omp_ds_i = data;
  int i;
  double sum = 0;
  int thrI = omp_get_thread_num();
  int thrN = omp_get_num_threads();
  int n0 = ..., n1 = ...;
  for (i = n0; i < n1; i++)
    sum += x[i] * y[i];
  omp_ds_i->sum /*atomic*/+= sum;
}
```

OpenMP в GCC

Поддержка на этапе компиляции:

- Языковые фронт-энды:
  поддержка прагм в C, C++, Fortran
- Ядро компилятора:
  трансляция прагм из AST в GIMPLE

Поддержка времени выполнения (libgomp):

- OpenMP API:
  `omp_get_thread_num()`, ...
- GOMP API:
  `GOMP_parallel`, ...

OpenMP 4.0

В OpenMP 4.0 добавлена поддержка акселераторов

Отличия акселераторов

- Свое пространство памяти
- Отдельная трансляция кода
- 3 уровня параллелизма в архитектуре

Новые прагмы/клаузы

- `omp target data`
- `omp target`
- `... map(from/to:...)`
- `omp teams distribute`

Уровни параллелизма в NVIDIA GPU

Иерархия нитей выполнения функций-ядер на GPU:

- Синхронные группы (32 контекста, warp)
  векторный параллелизм
- Блоки нитей (1-32 warp'а, CTA)
  быстрая синхронизация, общая память
- Сетки блоков (1...N CTA, grid)

Ограничения:

- Между блоками невозможна глобальная синхронизация
- В синхронных группах возможны взаимоблокировки

Параллелизм GPU в OpenMP

Уровни параллелизма GPU мотивируют стратегию отображения нитей в OpenMP:

- Блоки: OpenMP teams
- Синхронные группы: OpenMP threads
- Контексты выполнения: OpenMP SIMD lanes

Вне SIMD-регионов одна «логическая» нить — 32 GPU-нити

- Нужно обеспечить согласованность состояний
- Нужно организовать общие стеки
- Наблюдаемые эффекты должны наступать 1 раз (не 32)

Uniform SIMT

В синхронной группе нужно поддерживать:

- Согласованность: содержимое регистров одинаково
- Корректность: наблюдаемые эффекты наступают 1 раз

Почти все инструкции поддерживают эти свойства; исключения:

- Атомарные операции
- Внешние вызовы (`malloc`, `free`, `printf`)

```
atom.op.s32 dest, ...
```

Uniform SIMT

В синхронной группе нужно поддерживать:

- Согласованность: содержимое регистров одинаково
- Корректность: наблюдаемые эффекты наступают 1 раз

Почти все инструкции поддерживают эти свойства; исключения:

- Атомарные операции
- Внешние вызовы (`malloc`, `free`, `printf`)

```
atom.op.s32 dest, ...
```

```
@p  atom.op.s32  Rdst, ...
    shfl.idx.b32 Rdst, Rdst, Rm
```

Построим `p`, `Rm` так:

```
Rm = laneIdx & unisimt_mask;
p = (laneIdx == Rm)
```

General Design

Вне SIMD-регионов используются существующие стратегии организации параллельного счета:

- Поддержка всего многообразия OpenMP
- Требуется порт libgomp на архитектуру GPU

Для SIMD-регионов нужны специальные стратегии:

- Отображение на SIMT-параллелизм
- Желательна совместимость с классическим подходом

Пример – простой OpenMP-SIMD цикл

```
#pragma omp simd
for (i = START; i < END; i += STEP)
  BODY;
```

Пример – простой OpenMP-SIMD цикл

```
#pragma omp simd
for (i = START; i < END; i += STEP)
  BODY;
```

```
for (i = START + STEP * SIMT_LANE();
     i < END;
     i += STEP * SIMT_VF())
  BODY;
```

Пример – простой OpenMP-SIMD цикл

```
#pragma omp simd
for (i = START; i < END; i += STEP)
  BODY;
```

```
for (i = START + STEP * SIMT_LANE();
     i < END;
     i += STEP * SIMT_VF())
  BODY;
```

ompdevlow для CPU:

- SIMT_LANE() $\rightarrow$ 0
- SIMT_VF() $\rightarrow$ 1

Пример – safelen

```
#pragma omp simd safelen(SAFE)
for (i = START; i < END; i += STEP)
  BODY;
```

```
if (SIMT_LANE() < SAFE)
  for (i = START + STEP * SIMT_LANE();
        i < END;
        i += STEP * MIN(SAFE, SIMT_VF()))
    BODY;
```

Пример – lastprivate

```
#pragma omp simd lastprivate(v)
for (i = START; i < END; i += STEP)
  BODY;
```

```
for (i = START + STEP * SIMT_LANE();
     i < END;
     i += STEP * SIMT_VF())
  BODY;

i -= STEP * (SIMT_VF() - 1);
cond = !(i < END);
if (GOMP_SIMT_VOTE_ANY(cond)) {
    lane = GOMP_SIMT_LAST_LANE(cond);
    v = GOMP_SIMT_XCHG_IDX(v, lane);
}
```

Пример – reduction

```
#pragma omp simd reduction(+:v)
for (i = START; i < END; i += STEP)
  BODY;
```

```
for (i = START + STEP * SIMT_LANE();
     i < END;
     i += STEP * SIMT_VF())
  BODY;

for (t = 1; t < SIMT_VF(); t <<= 1)
  v += GOMP_SIMT_XCHG_BFLY(v, t);
```

Пример – reduction

```
#pragma omp simd reduction(+:v)
for (i = START; i < END; i += STEP)
  BODY;
```

```
for (i = START + STEP * SIMT_LANE();
     i < END;
     i += STEP * SIMT_VF())
  BODY;

for (t = 1; t < SIMT_VF(); t <<= 1)
  v += GOMP_SIMT_XCHG_BFLY(v, t);
```

Требуется коммутативность для операции редукции

Пример – reduction

```
#pragma omp simd reduction(+:v)
for (i = START; i < END; i += STEP)
  BODY;
```

```
for (i = START + STEP * SIMT_LANE();
     i < END;
     i += STEP * SIMT_VF())
  BODY;

for (t = 1; t < SIMT_VF(); t <<= 1)
  v += GOMP_SIMT_XCHG_BFLY(v, t);
```

Требуется коммутативность для операции редукции

Копирование NaN payloads

```
if (v != v) v = GOMP_SIMT_XCHG_IDX(v, 0);
```

Пример – ordered SIMD

```
#pragma omp simd
...
#pragma omp ordered
  STMT;
```

```
for (t = SIMT_LANE();
     GOMP_SIMT_VOTE_ANY(t >= 0);
     t--)
  if (GOMP_SIMT_ORDERED_PRED(t))
    STMT;
```

Результаты и планы

Текущий статус:

- Код включен в основную ветку GCC
- Была важна корректность и поддержка всех конструкций OpenMP
- Известны недочеты в производительности

Будущие работы:

- Работа с сообществом
- Поддержка OpenMP 4.5
- Улучшение производительности