ИСП РАН

Производительность расчетов в OpenFOAM с использованием GPU

Александр Монаков

amonakov@ispras.ru

Институт Системного Программирования

5 декабря 2014 г.

Гетерогенные системы

- Наращивание вычислительной мощности суперкомпьютеров: увеличение количества узлов/процессоров
- Повышение производительности процессоров затруднено:
  - Ограничения по тепловыделению
  - Приоритет на производительности однопоточного кода

Возможное решение: гибридная архитектура

- Дополнить центральный процессор акселератором
- Архитектура акселератора оптимизируется под массивно-параллельные задачи
- GPU: NVIDIA Tesla, AMD FirePro
- Intel MIC

GPU

- Специализированная массивно-параллельная архитектура
- 1-4 TFLOPS, 100-300 GB/s на одном устройстве
- Требуют задач с высоким параллелизмом

Преимущества

1. Производительность
2. Энергоэффективность (10x производительность при 2x энергопотреблении)
3. Доступность
   - В Top500: 62 системы с акселераторами, 44 NVIDIA
   - В персональных компьютерах

Использование в OpenFOAM

50-80% времени счета: решение СЛАУ

- PCG: Метод сопряженных градиентов
- AMG: Многосеточный метод

Рассматриваем PCG

- Производительность ограничена:
  - Пропускной способностью памяти
  - Скоростью сети
- Сложности при переносе на GPU
  - Предобуславливание
  - Поддержка MPI
  - Минимизация накладных расходов

Предобуславливание на GPU

OpenFOAM: обычно используется DILU

- Подготовка/применение: решение треугольных систем
- Эффективный, но не обладает высоким параллелизмом
- Переупорядочивание может повысит паралелизм за счет снижения эффективности

Диагональное преобуславливание:

- Подготовка/применение: деление на диагональные коэффициенты
- Высокий параллелизм, низкая эффективность

Предобуславливание на GPU

Наш выбор для GPU: AINV

$$W^T AZ = D + e$$

- Подготовка: вычисление явного разложения приближенной обратной матрицы
- Применение: домножение на две разреженные матрицы
- Хорошо подходит для GPU
- Затраты на подготовку соизмеримы с решением СЛАУ

Асинхронное вычисление AINV

Предположим:

1. Что связность сетки не меняется
   $\Rightarrow$ Портрет разреженной матрицы СЛАУ фиксирован
2. СЛАУ для последних шагов по времени имеют близкие коэффициенты
   $\Rightarrow$ Можно переиспользовать предобуславливатели

Тогда:

1. Предобуславливатели вычисляются асинхронно в отдельном потоке
2. На GPU используется последний посчитанный предобуславливатель

Поддержка MPI

Схема параллельного счета в OpenFOAM:

- Разделение сетки (без перекрытия или “гало”)
- Межпроцессорные границы – частный случай граничных условий на сетке
- Процессоры обмениваются коэффициентами при домножении на матрицу системы
- Предобуславливание выполняется без MPI-коммуникаций

Поддержка на GPU:

- Сводится к поддержке произвольных граничных условий
- Параллельно с домножением на матрицу на GPU:
  1. Скопировать граничные коэф-ты в память CPU
  2. Вызвать методы обновления граничных условий
  3. Просуммировать обновленные коэф-ты на GPU

Снижение накладных расходов

1. Преобразование матрицы в GPU-формат только на первом шаге
   На последующих шагах – обновлять ненулевые коэффициенты

2. Автоматическая настройка
   - Оптимизация умножения разреженных матриц на GPU
   - Разреженность предобуславливателя AINV

3. Минимизация CPU-GPU коммуникаций
   - Исключение лишних копирований
   - CUDA-aware MPI
   - Конвейеризованные варианты метода сопряженных градиентов

Конвейеризованные методы сопряженных градиентов

$$
\begin{aligned}
s &= M^{-1}r \\
\delta &= (s, r) \\
\hline
\beta &= \delta_i/\delta_{i-1} \\
p &= s + \beta p \\
s &= Ap \\
\gamma &= (s, p) \\
\hline
\alpha &= \delta_i/\gamma \\
x &= x + \alpha p \\
r &= r - \alpha s
\end{aligned}
$$

$$
\begin{aligned}
m &= M^{-1}w \\
n &= Am \\
\beta &= \gamma_i/\gamma_{i-1} \\
\alpha &= \gamma_i/(\delta - \beta\gamma_i/\alpha) \\
z &= n + \beta z \\
q &= m + \beta q \\
s &= w + \beta s \\
p &= u + \beta p \\
x &= x + \alpha p \\
r &= r - \alpha s \\
u &= u - \alpha q \\
w &= w - \alpha z \\
\gamma_i &= (r, u) \\
\delta &= (w, u)
\end{aligned}
$$

Зависимость производительности от размера сетки

Модельная задача: icoFoam, cavity3d; 12 ядер, 2 GTX Titan

- 373K: 16 с на CPU, 18 с на GPU
- 1M: 91 с на CPU, 60 с на GPU (общее время)
  73 с на CPU, 43 с на GPU (ускоренная часть)
- 2M: 239 с на CPU, 113 с на GPU (общее время)
  199 с на CPU, 71 с на GPU (ускоренная часть)
- 8M: 2160 с на CPU, 563 с на GPU (общее время)
  1824 с на CPU, 300 с на GPU (ускоренная часть)

Текущие работы

- Включение в релиз-версии FOAM-Extend
  текущая версия доступна на Github:
  https://github.com/amonakov/ispm-sparse-lib
  https://github.com/amonakov/openfoam-extend-foam-extend-3.1
- Итерационные методы для асимметричных систем
- Решение блочных систем
- Дальнейшие оптимизации

Вопросы?

Спасибо!