Лекция 6: Использование графических процессоров для ускорения вычислений. Решение некоторых задач

Н.Д. Смирнова

Санкт-Петербургский государственный Политехнический университет

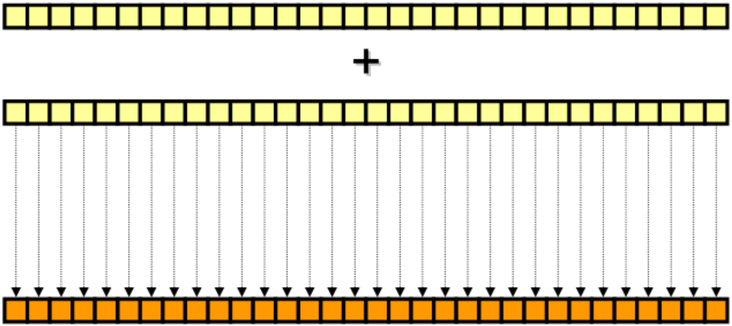
15.11.2011

Содержание

Содержание

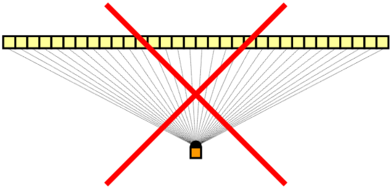
Сложение, вычитание векторов

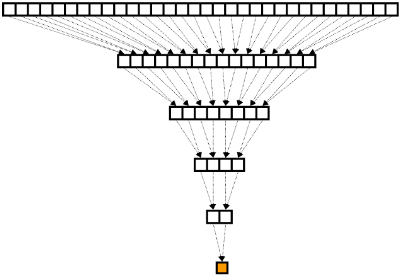
- тривиальные поэлементные операции



Произведение элементов, норма,...

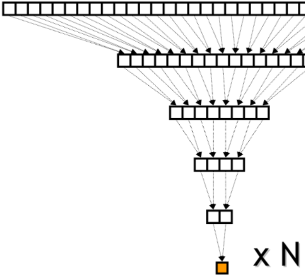
- интуитивная реализация может привести к одному потому без распараллеливания
- надо делать редукцию





Произведение вектора на матрицу

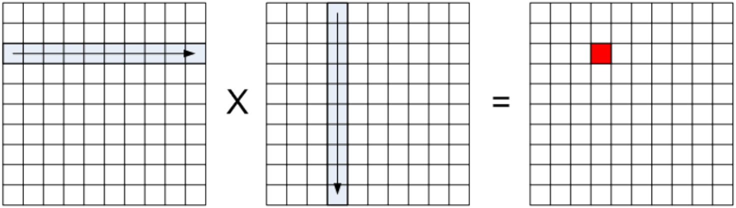
- делаем N редукций параллельно



Содержание

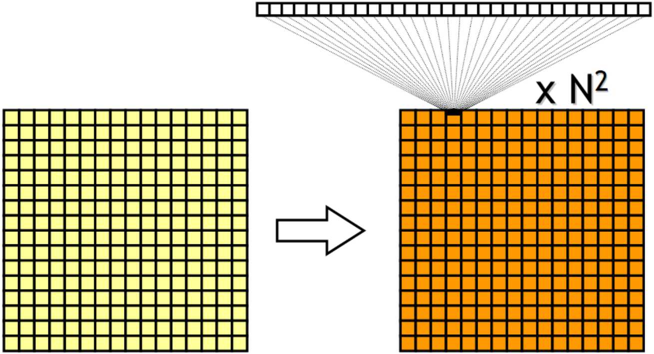
Перемножение матриц

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)
  {
    c[i][j] = 0;
    for (k=0; k<N; k++)
      c[i][j] += A[i][k] * B[k][j];
  }
```



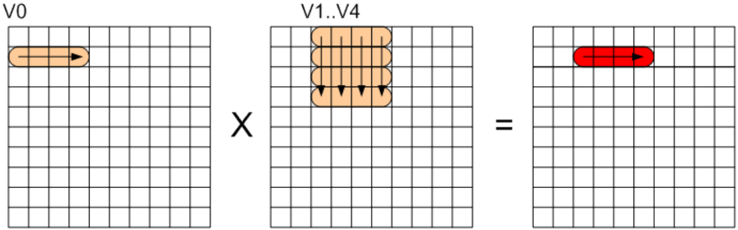
Перемножение матриц

- а вот тут возможно скалярное произведение двух векторов в одном потоке
- N^2 скалярных произведений векторов



Оптимизация

- вспомним про SIMD
- используем векторную арифметику
- за одну операцию обрабатываем сразу несколько групп данных



ВАЖНО: подходит не для всех GPU (NVidia CUDA-процессоры скалярны)

Содержание

Методы решения СЛАУ

Предподчтение отдается итерационным методам

- легче реализуются для GPU (необходимо лишь умножение матрицы на вектор)
- не требуют обратной связи (в противоположность методу Гаусса)
- дают больший прирост относительно CPU
- минимальная коммуникация между нитями
- но встречается и использование LU разложений

Методы

- метод Якоби (выбирается из-за простоты)
- метод Гаусса-Зейделя
- метод сопряженных градиентов

Условие окончания итераций

- можно просто выбрать конечное число итераций (для приложений реального времени)
- сделать k итераций, проверить невязку

Как и какую норму используют?

- первая норма $||A||_1 = \max_j \sum_i |a_{ij}|$
- бесконечная норма $||A||_\infty = \max_i \sum_j |a_{ij}|$
- считаем редукцией

Содержание

Постановка задачи

Задача: - расчет динамики поведения N тел

Основные данные

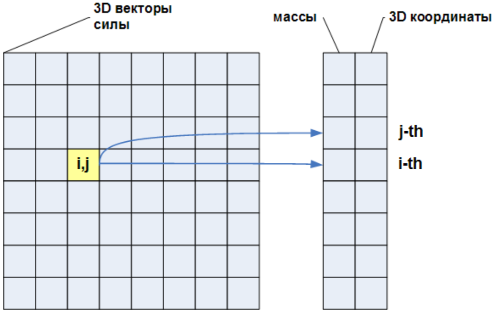
- массив 3D координат тел
- массив 3D скоростей тел
- массив масс

Алгоритм

- расчитать силы для всех пар
- расчитать сумму сил для каждого объекта
- пересчитать позицию, скорость

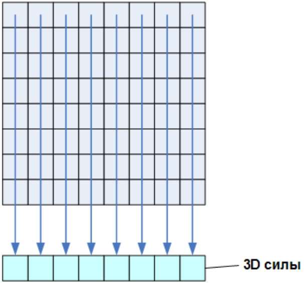
Расчет сил для каждой пары объектов

- рачет сил гравитационного взаимодействия для каждой пары тел i,j $F_{ij} = G\frac{M_i M_j}{d^2(\vec{p}_i, \vec{p}_j)}$

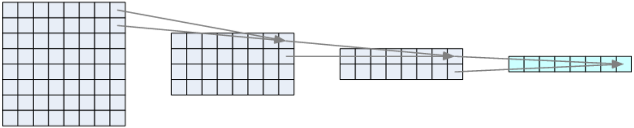


Суммирование сил для каждого объекта

- надо просуммировать:

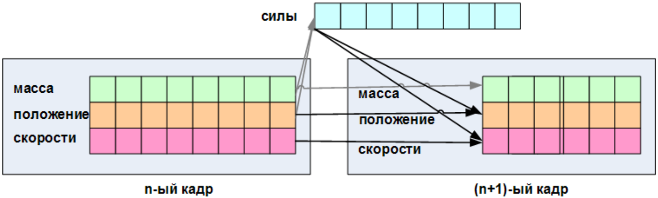


- на деле используем для этого reduce:



Расчет новых положений и скоростей

- для каждого тела расчет положения и скорости
  - $\vec{v}^{(n+1)} = \vec{v}^{(n)} + \vec{f}^{(n)}\Delta t$
  - $\vec{p}^{(n+1)} = \vec{p}^{(n)} + \vec{v}^{(n)}\Delta t$
  - расчет методом Эйлера (для простоты)



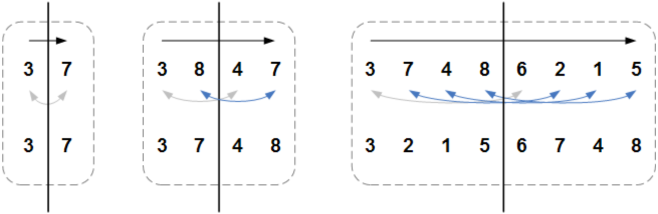
Содержание

Принципы

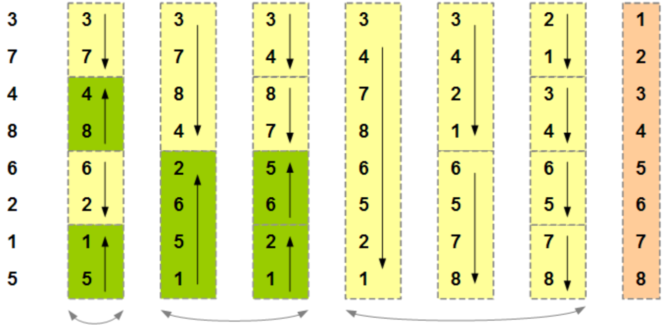
- использует понятие *битоническая последовательность* (bitonic sequence)
- битоническая последовательность состоит из двух монотонных подпоследовательностей: неубывающей и невозрастающей
- $A : \{3, 4, 7, 8\}$
  $B : \{6, 5, 2, 1\}$
  $AB : \{3, 4, 7, 8, 6, 5, 2, 1\}$ - битоническая последовательность
- используется подход "разделяй и властвуй"
- теорема о битонической последовательности
  - пусть $A = \{a_i | i = \overline{0, 2n-1}\}$ - битоническая
  - тогда можно разделить A на 2 битонические подпоследовательности $A_1 = \{a_i | i = \overline{0, n-1}\}$ $A_2 = \{a_i | i = \overline{n, 2n-1}\}$ так, чтобы $\forall a_* \in A_1, \forall a_{**} \in A_2 : a_* \leq a_{**}$

Ядро алгоритма

- сортируем левую половину по неубыванию
- сортируем правую половину по невозрастанию
- на полученной последовательности выполняем *битоническое слияние* (bitonic merge), после этого битоническое слияние выполняется рекурсивно для каждой половины последовательности



Пример


3 7 4 8 6 2 1 5
3 7 4 8 6 2 1 5
3 7 8 4 2 6 5 1
3 4 8 7 5 6 2 1
3 4 7 8 6 5 2 1
3 4 2 1 6 5 7 8
2 1 3 4 6 5 7 8
1 2 3 4 5 6 7 8

Сложность

- $O(\log^2 n)$ проходов
- каждый проход требует n сравнений/перестановок
- полная сложность $O(n \log^2 n)$

ПРОДОЛЖЕНИЕ СЛЕДУЕТ...