Mixed Precision

Christoph Höppke, Daniel Thomaschewsik

TU Dortmund

Version:16. Mai 2016

technische universität dortmund fakultät für mathematik

Content

Content

Definition

Definition:

An Algorithm that uses different precisions in its computation

Example: double(a) + float(b)

Goal:

Obtain the same accuracy by using high precision
but better performance by utilizing low precision computations

Performance Gains

- **Bandwidth bound algorithm**
  - 64 bit = 1 double = 2 floats = 4 halfs
  - More variables per bandwidth
  - More variables per storage
  - Applies to all memory levels: network, disc, main, device, local, register
- **Computation bound algorithm**
  - 1 double addition $\approx$ 2 float additions $\approx$ 4 half additions (linear)
  - 1 double multip. $\approx$ 4 float multip. $\approx$ 16 half multip. (quadratic)
  - $\rightarrow$ up to 16 times better computational efficiency

Roundoff and Cancellation

Definition Machine Precision

The smalles positive number ϵ for wich a floating point calculations evaluates the expression $1 + \epsilon > 1$ to be true.

Examples:

- $\epsilon_{double} \approx 2.220446049250313 \cdot 10^{-16}$
- $\epsilon_{float} \approx 1.1920929 \cdot 10^{-7}$
- $\epsilon_{half} \approx 9.765625 \cdot 10^{-4}$
- So more precision is usually better

Cancellation

additive roundoff	$a = 1 + 0.00000004 = 1.00000004$		$=_{fl} 1$
multiplicative roundoff	$b = 1.0002 \cdot 0.9998 = 0.99999996$		$=_{fl} 1$
cancellation	$c \in \{a, b\}$	$\pm 4 = (c - 1) \cdot 10^8$	$=_{fl} 0$
oder of operations	$1 + 0.00000004 - 1 =_{fl} 0$		
	$1 - 1 + 0.00000004 =_{fl} 0.0000004$		

Computational Precision vs Accuracy of Result

Instructive Example [S.M. Rump, 1988]

$f(x, y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + 0.5x/y$
$x_0 = 77617, y_0 = 33096$

float s23e8	1.1726
double s52e11	1.17260394005318
quad s63e15	1.172603940053178631

The correct result is:

-0.827396059946821368141165095479816 29

DataError and Truncation

- Data error occurs when the exact value has to be truncated for storage in the binary format
  - π , $\sqrt{2}$, $sin(2)$, e^2, $1/3$
  - every rational number with a denominator that has a prime factor other than 2
- How can float be better than double?
  - There is no data error in the operands
  - The errors can cancel out themselves favorably

Floating Point Operations. A deeper analysis

- Number representation
  - half s10e5 a=|1 bit sign s_a | 10 bit mantissa m_a | 5 bit exp. e_a
  - float s23e8 b = |1 bit sign s_b | 23 bit mantissa m_b | 8 bit exp. e_b
  - double s52e11 c = |1 bit sign s_c | 52 bit mantissa m_c | 11 bit exp. e_c
- Multiplication $float(a) \cdot float(b)$
  - Operations: $s_a \cdot s_b$,$m_a \cdot m_b$,$e_a \cdot e_b$
  - Excat format: s46e9 = s23e8 ·s23e8
  - Main error: Mantissa truncated from 46 bit to 23 bit
- Addition $float(a) + float(b)$
  - Operations: $e_{diff} = e_a - e_b$, $m_a + (m_b >> e_{diff})$, normalize
  - Exact format: s278e8 = s23e8+s23e8
  - Main error: Maintissa truncated from 278 bit to 23 bit

Content

History of GPGPU

1999: NVIDIA GeForce 256

- Term GPU was coined during the launch
- GPGPU calculations were only achievable by "Hacking the GPU"(verry cumbersome and error-prone)

2003-2007 First wave of GPU computing

- Floating point support
- Performance improvements
- Geforce 8800GT Release (29.October 2007)
  first CUDA enabled Consumer GPU
- Double precision was not available on the GPU
- Mixed Precision was the ONLY way to utilize GPU horesepower without precision compromises
- Mixed Precision lead to a speedup of factor 3-5 in FEM applications

Why Mixed Precision is difficult

- Data has to be transferred to the GPU over the relativly slow PCIe interface
- A lot of time is wasted while waiting for data

Unconventional compuation Hardware

NVIDIA Jetson TK1 Bord

- CPU and GPU share the same memory and use the same memory interface
- Copying data from CPU to GPU and vice versa is much faster

Content

Example Calculation

Problem

Solve the Poission problem $-\Delta u = 1,\ x \in \Omega$ with dirichlet boundry conditions $u \equiv 0,\ x \in \partial\Omega$ using conforming quadrilateral elements for the finite-element discretization of the unit square $\Omega = (-1, 1)^2$

- Method 1: Using a CG solver
- Method 2 : Using a more sophisticated MG solver

Testresults pending

Imagine a nice Plot