Mixed Precision

Christoph Höppke, Daniel Tomaschewski

TU Dortmund

Version:22. Mai 2016

technische universität dortmund fakultät für mathematik

Content

Content

Definition

Definition:

An Algorithm that uses different precisions in its computation

Goal:

Obtain the **same accuracy** by using high precision

but **better performance** by utilizing low precision computations

Performance Gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halfs
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, main, device, local, register

Performance Gains for compuation bound algorithms

- 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

Definition

Definition:

An Algorithm that uses different precisions in its computation

Goal:

Obtain the **same accuracy** by using high precision

but **better performance** by utilizing low precision computations

Performance Gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halfs
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, main, device, local, register

Performance Gains for compuation bound algorithms

- 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

Definition

Definition:

An Algorithm that uses different precisions in its computation

Goal:

Obtain the **same accuracy** by using high precision

but **better performance** by utilizing low precision computations

Performance Gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halfs
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, main, device, local, register

Performance Gains for compuation bound algorithms

- 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

Definition

Definition:

An Algorithm that uses different precisions in its computation

Goal:

Obtain the **same accuracy** by using high precision

but **better performance** by utilizing low precision computations

Performance Gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halfs
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, main, device, local, register

Performance Gains for compuation bound algorithms

- 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	$1 + 0.00000004 - 1 =_{fl} 0$		
	$1 - 1 + 0.00000004 =_{fl} 0.0000004$		

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	$c \in \{a, b\} \qquad \pm 4 = (c-1) \cdot 10^8$	$=_{fl} 0$
oder of operations	$1 + 0.00000004 - 1 =_{fl} 0$	
	$1 - 1 + 0.00000004 =_{fl} 0.0000004$	

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	$c \in \{a, b\} \qquad \pm 4 = (c-1) \cdot 10^8$	$=_{fl} 0$
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

The correct result is:

-0.82739605994682136814116509547981629

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

The correct result is:

-0.827396059946821368141165095479816291

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

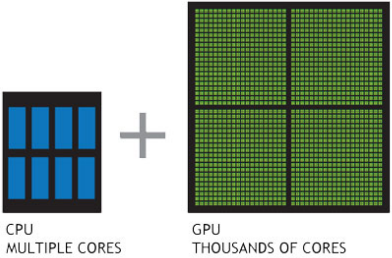
Number representation→ almost all numbers have to be truncated

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		
Precision	Exactformat	Mantissa truncation:
half(a) · half(b)	s20e6	from 20 to 10 bit
float(a) · float(b)	s46e9	from 46 to 23 bit
double(a) · double(b)	s104e12	from 104 to 52 bit
Addition		
Precision	Exactformat	Mantissa truncation:
half(a) · half(b)	s41e5	from 41 to 10 bit
float(a) · float(b)	s278e8	from 278 to 23 bit
double(a) · double(b)	s2099e11	from 2099 to 52 bit

Why Mixed Precision is difficult

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



A brief history overview

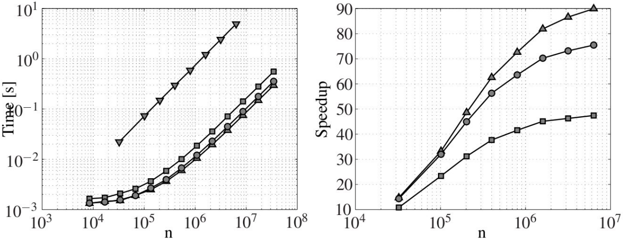
Trends:

- Memoryclock speeds are increasing
  - GTX 980 Ti Memoryclock speed: 2x 1753 MHz
  - GTX 1080 Memoryclock speeds: 4x 2500 MHz (+185%)
- Unconventional computation Hardware
  - APU's
  - SOC's such as the NVIDIA Tegra K1
- New technologys
  - Unified memory
  - NVIDIA pushing the use of half precision by implementing support for half and half2 in CUDA Toolkit version 7.5

→ Sharing memory between CPU and GPU is becoming easyer and more effective

A brief history overview

Past achievements:



(a) Absolute timings. (b) Speedup relative to CPU (single thread).

Fig. 3 Scalability tests and performance comparisons on Tesla C2050 in single precision (−▲−), double precision (−■−), mixed precision (− ● −), and CPU (single thread) code (−▼−). Sixth order spatial discretization employed. The iterative defect correction method has been left-preconditioned with a Gauss-Seidel V-cycle multigrid strategy on each architecture.

[1]

→ about 35% performance increase by using mixedprecision

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 double precision
- Method 2 : Using iterative refinement

Testresults pending

Imagine a nice Plot

Bibliography

Madsen M. Glimberg S. L., Engsig-Karup A. P.
A fast gpu-accelecrated mixed-precision strategy for full nonlinear water wave computation.
Technical report, ResearchGate, January 2013.