Давхар интеграл бодоход CUDA програмчлал ашиглах нь

Бямбасүрэнгийн Янжмаа
Электроникийн IV түвшин, МУИС, ХШУИС
Даланбаярын Болормаа, профессор, МУИС, ХШУИС

Хураангуй – Шинжлэх ухааны тооцооллыг супер компьютер ашиглахгүй хийх боломж нь массив параллель архитектур ашиглах бөгөөд үүний хамгийн хялбар, хямд өртөгтэй шийдэл нь GPU-г тооцоолол ашиглах явдал юм. Уг судалгааны ажлаар CUDA програмчлал ашиглан давхар интеграл бодоход интегралын алхмаас бодолтын хугацаа болон нарийвчлал хэрхэн хамаарахыг судаллаа.

Түлхүүр үгс: CUDA програмчлал, давхар интерал, трапецийн арга

1. Оршил

Тооцоолон бодох машины хамгийн чухал параметр нь хурдан ажиллагаа. Процессорыг хөгжүүлж ирсэн чиг хандлага нь үргэлж хурдан ажиллагааг нэмэгдүүлэхэд чиглэсэн байдаг.

Хурдан ажиллагааг нэмэгдүүлэхийн тулд олон цөмт архитектур бий болсон ч програмыг N процессорт хуваасnаар Амдалын хуулиар хурдан ажиллагаа:

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

болно. Энд- P- программын N тооны процессорт хуваах боломжтой хэсэг. Процессорын тоо ихсэхэд хурдан ажиллагаа $1/(1-P)$ рүү тэмүүлэх буюу програмын алгоритм болон арга нь параллель гүйцэтгэлд сайн тохирсон байх шаардлагатай болно.

CPU-г инструкцүүдийг цуваагаар нэг процесс байдлаар хамгийн өндөр бүтээмжтэйгээр гүйцэтгэхээр зохион байгуулсан байдаг. Харин GPU-г олон процессыг параллелиар хурдан гүйцэтгэхээр зохион байгуулсан байдаг. Өөрөөр хэлбэл график картын ажиллагаа нь анхнаасаа параллелиар ажиллах processing unit-уудтай. Их өгөгдөл дээр нэг ижил математик үйлдлийг дараалан хийхэд GPU ашиглах нь олон цөмт процессор ашиглахаас ч хурдан байна.

CUDA архитектур програмчлалын C болон C+ хэлийг дэмждэг нь GPU-г параллель тооцоололд ашиглахад нэмэлтээр прогамчлалын хэлний хүндрэл гарахгүй байх давуу талыг бий болгосон.

Уг судалгааны ажлаар CUDA програмчлал ашиглан давхар интеграл бодох аргуудыг судлан трапецийн аргаар бодох C+ кодыг туршлаа. Орчин үед нэг электронт транзистор, супер торт бүтэц гэх мэт геометрийн хязгаарлалттай элементүүд гарч ирсэн нь электроникийн загварчлалд давхар интеграл хэрэглэн бодох бодлогуудыг ихээр бий болгож байна.

2. Давхар интеграл бодох аргууд

Электроник, материал судлал гэх мэт шинжлэх ухааны загварчлалд давхар интеграл бодох шаардлага их гардаг. Эдгээр бодлогын үр дүн нь интегралыг завсрын мужуудад хуваасан тооноос хамааралтай байдаг. Давхар интегралыг дараах түгээмэл аргуудаар:

Монте - Карлогийн арга

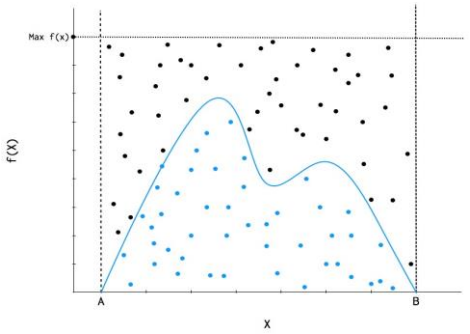
Тэгш өнцөгтийн арга

Трапецийн арга

Симпсоний аргаар ихэвчлэн боддог.

2.1. Монтекарлогийн арга

Энэхүү арга нь ху хавтгайн муж дотроос санамсаргүйгээр тодорхой тооны цэгүүд сонгон авч, эх функцад орлуулан, цэг бүрт харгалзах утгын нийлбэрүүдийг олсноор тухайн дүрсийн нийт гадаргуугийн талбайг олдог.

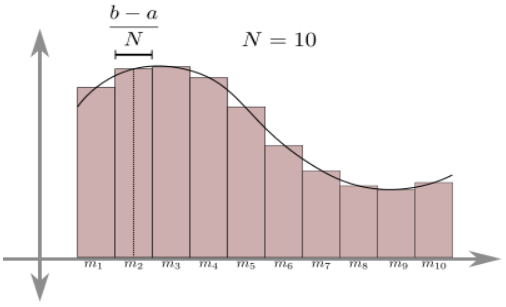


Зураг 1. Монтекарлогийн арга
[http://pymcmc.readthedocs.org/en/jss-gp/]

Энэ аргын давуу тал нь тухайн функцийн муж дотроос цэгүүд нь түүвэрлэгдсэн тохиолдолд хамгийн хялбар хурдан үнэн зөв тооцоолол хийх боломжтой боловч эсрэгээр тухайн функцийн мужийн гаднаас цэгүүд түүвэрлэгдсэн тохиолдолд хамгийн их алдаатай тооцоолол хийдэг сул талтай арга юм. Энэ аргыг график дүрслэлийг зураг 1-д харууллаа.

2.2 Тэгш өнцөгтийн арга

Энэхүү арга нь тухайн функц болох муруйн мужийг олон жижиг тэгш өнцөгтүүдэд хуваан тэдгээр тэгш өнцөгтүүдийн талбайг олно. Үүний дараа олсон талбай бүрийг нэмснээр тухайн функцийн нийт талбайг олж байгаа арга юм.

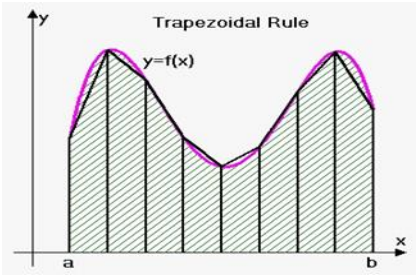


Зураг 2. Тэгш өнцөгтийн арга
[http://www2.bakersfieldcollege.edu/resperic/Math6A/Lectures/ch5/]

Энэ аргын давуу тал нь олон тэгш өнцөгтүүдэд хувааж хийж байгаа нь тооцооллыг хялбар бөгөөд алдаа багатай тооцоолол хийх боломжийг бий болгож байгаа ч тэгш өнцөгтүүд нь тухайн функцийн хязгаараас өнцгөөрөө илүү гарсан үед тооцоолол буруу хийгдэх сул талтай арга юм. Зураг 2-т энэ аргын график дүрслэлийг харууллаа.

2.3 Трапецийн арга

Энэхүү арга нь тухайн функц болох муруйн мужийг олон жижиг трапецуудад хуваан тэдгээр трапецуудын талбайг олон, олсон талбай бүрийг нэмснээр тухайн функцийн нийт талбайг олж байгаа арга юм.

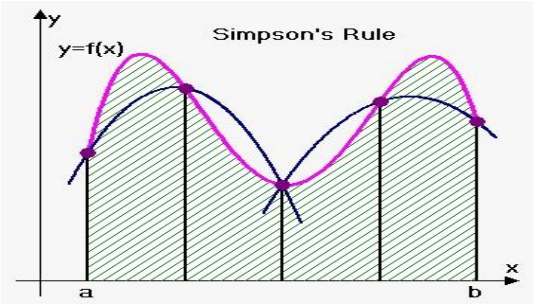


Зураг 3: Трапецийн арга
[http://www.emathhelp.net/images/cal]c/]

Энэ аргын давуу тал нь олон жижиг трапецад хуваах нь тооцооллыг хамгийн алдаа багатай хийдэг. Харин мужуудаа олон жижиг интервалд хуваагаагүй тохиолдолд алдаа их гарах сул талтай арга юм. Энэ аргын график дүрслэлийг зураг 3-т харууллаа.

2.4 Симпсоний арга

Энэхүү арга нь тухайн функц болох муруйн мужийг олон жижиг хэсгүүдэд хувааж өгөх бөгөөд гол онцлог нь хуваалтууд нь парабол хэлбэрийн байхаар маш олон жижиг хэсгүүдэд хуваагддаг онцлог шинж чанартай. Энэ аргын давуу тал нь алдаа гарах магадлал хамгийн бага боловч хугацаа их шаарддагдаг сул талтай (зураг 4).



Зураг 4. Симпсоний арга
[https://ecourses.ou.edu/ebook/math/ch08/sec084/]

3. ГҮЙЦЭТГЭЛ

Уг судалгааны ажилд давхар интеграл бодох трапецийн аргыг сонгон авсан. Энэ арга нь хурдан хугацаанд алдаагүй хариуг олж чаддаг учраас эн аргыг сонгон авсан.

Энэ аргын сул тал болох интегралын интервалын тоо цөөдөх асуудал параллель аргаар GPU ашиглан бодоход давуу тал болох боломжтой.

Трапецийн арга нь дараах томьёогоор илэрхийлэгддэг:

$$\iint_R f(x,y)dA = \int_a^b \int_c^d f(x,y)dydx \approx$$

$$\frac{1}{4}kh\{(f[a,c] + f[b,c] + f[a,d] + f[b,d]) +$$

$$(2\sum_{i=1}^{m-1} f[X_i,c] + 2\sum_{i=1}^{m-1} f[X_i,d]) +$$

$$(2\sum_{j=1}^{n-1} f[a,Y_j] + 2\sum_{j=1}^{n-1} f[b,Y_j]) +$$

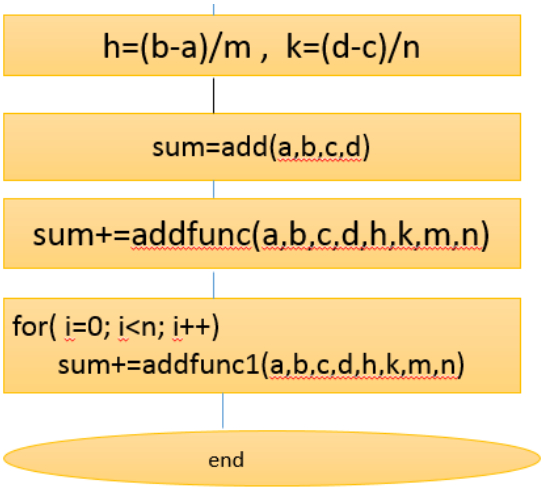
$$4\sum_{j=1}^{n-1}(\sum_{i=1}^{m-1} f[X_i,Y_j])\}$$

Энд улаанаар тодруулсан хэсэг нь a , b, c, d цэгийн утгуудыг олж өгч байгаа бол цэнхрээр тодруулсан хэсэг нь хуурмаг хэсгийн трапецийн хэсгийн нийлбэрийг олж байгаа. Харин ногоонoор тодруулсан хэсэг нь x координатын дагуух утгуудын нийлбэрийг олж байгаа.

Дээрх трапецийн аргын дагуу алгоритмаа зохиож кодыг CUDA C хэл дээр бичиж амжилттай ажиллуулсан.

Параллелиар тооцоолол хийхэд хамгийн гол анхаарах зүйл нь device -ийн санах ойн зохион байгуулалтыг зөв хийж өгөх хэрэгтэй. Энэхүү судалгааны ажилд санах ойн зохион байгуулалт хийхдээ нэг блоконд байх трейдийг матриц

хэлбэртэй байхаар зохион байгуулсан бөгөөд энэ нь олон жижиг хэсгийн нийлбэрийг нэгэн зэрэг хийх боломжтой болгож байгаа юм. Доор зураг 5-д трапецийн аргын параллель алгоритмыг үзүүлэв.



Зураг 5. Трапецийн аргын параллель алгоритм

add(a, b, c, d) функц нь улаанаар тодруулсан хэсгийн хийж байга хост функц юм. Харин addfunc(),addfunc1() функцүүд нь глобал санах ойг ашиглаж байгаа device-ийн функцүүд юм.

4. Үр дүн

Кодыг зөв эсэхийг шалгахын тулд f(x,y)=6xy2 R=[1 2][2 4] гэсэн интегралыг 384 Cuda core бүхий GTX 745 GPU карттай i7 процессор дээр бодуулж үзсэн. Бодлогын үр дүн болон хугацааг доор зургаар харууллаа. Уг бодлогыг уламжлалт аргаар нь бодоход 84 гарсан. Харин параллелиар куда картан дээр бодоход интервалын хуваалтын тоо ихсэх тусам 84 лүү хариу нь дөхөж байсан. Бодлогын үр дүн хугацааны хамаарлыг доор хүснэгт 4.1-т харууллаа. Энд N-ээр интегралыг завсрын утгуудад хуваасан хуваалтын тоог тэмдэглэсэн.

Хүснэгт 1. GTX 745 GPU карттай i7 процессор дээрх нарийвчлал

N хэмжээ	Үр дүн CPU	Үр дүн/ GPU
512	84.07	84.07
1024	84.05	84.05
2048	83.89	84.03
4096	82. 88	84.01
8192	64.01	84.00
16384	16.01	84.00
32768	4.01	84.00
65536	1.0	84.00
131072	0.25	84.00
262144	0.06	6.3

Хүснэгт 2. GTX 745 GPU карттай i7 процессор дээрх бодолтын хугацаа

N хэмжээ	CPU хугацаа	GPU хугацаа
512	0s	0 s
1024	0s	0.52s
2048	0.03s	0.97s
4096	0.13s	1.95s
8192	0.55s	3.92s
16384	2.23s	8.71s
32768	9.3s	19.47s
65536	35.761	50.05s
131072	143.55s	136.95s
262144	574.28s	434.96s

5. Дүгнэлт

Хүснэгт 1, 2-оос харахад GTX 745 GPU карттай i7 процессор дээр цуваа код хуваалтын тоо 8192 байхаас эхлэн алдаатай бодож байхад параллель код (GPU дээр) хуваалтын тоо 262144 бодоход алдаж эхэлж байна. Мөн бодолтын хугацаа (хүснэгт 2) GPU дээр бага байгаа нь харагдаж байна.
Давхар интеграл бодсон кодоо хялбар функц дээр шалгасан ч энэ нь 2 хэмжээст электрон хий буюу хагас дамжуулагч гетеро бүтцийн электрон тээвэрлэлтийн тэгшитгэлийг бодохоос зарчмын хувьд ялгаа гарахгүй. Иймээс дээрх кодыг гетеро бүтцийн болон 2D нанотранзисторын электрон тээвэрлэлтийн бодлогод ашиглах боломжтой.

6. Ашигласан материал

[1].http://www.mpia.de/~mordasini/UKNUM/integration.pdf

[2].http://pages.pacificcoast.net/~cazelais/187/simpso.pdf

[3].http://pages.cs.wisc.edu/~amos/412/lecturenotes/lecture1.pdf

[4].http://www.cs.nyu.edu/courses/fall06/G22.2112001/Montecarlo.pdf