Introduction to GPU programming: GPU accelerated Convex Hull computation

Computational geometry

FAGUREL Petro

Why

vvna

MergeHu

BottomUp MergeHul

Parallel MergeHu

Results

Reference

Computational geometry Introduction to GPU programming: GPU accelerated Convex Hull computation

FAGUREL Petro

December 2021

FAGUREL Petro

Why

...

vvnat

iviergen

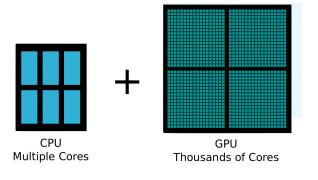
BottomU MergeHul

Parallel MergeHu

Results

Reference

Why



• Laverage the parallel power

[1]

Why

How

.

BottomU

MergeHul

Parallel MergeHul

Results

Reference

How

OpenCL MergeHull:

 https://github.com/pfagurel/OpenCL-ConvexHull/ blob/main/GPUHull.pdf

CUDA QuickHull:

 https://timiskhakov.github.io/posts/ computing-the-convex-hull-on-gpu

OpenCL QuickHull code:

 https://github.com/pfagurel/OpenCL-ConvexHull/ tree/VsProject

Interface code:

 https://github.com/pfagurel/OpenCL-ConvexHull/ tree/Interface VVh

How

What

Wicigoria

BottomUp MergeHull

Parallel MergeHul

Results

Reference

What

- MergeHull
- BottomUp MergeHull
- Parallel MergeHull
- Results

Convex Hull

Computational

FAGUREL Petro

Wh

Hov

MergeHull

BottomUp MergeHul

Parallel MergeHul

Results

Reference

MergeHull Algorithm

Algorithm CH

```
Input. A set S = \{a_1, \ldots, a_n\}, where a_i \in E^d and x_1(a_i) < x_1(a_i) \Leftrightarrow i < j \text{ for } i, j = 1, \ldots, n.
Output. The convex hull CH(S) of S.

Step 1. Subdivide S into S_1 = \{a_1, \ldots, a_{\lfloor \frac{1}{2}n \rfloor}\} and S_2 = \{a_{\lfloor \frac{1}{2}n \rfloor}, \ldots, a_n\}.

Step 2. Apply recursively Algorithm CH to S_1 and S_2 to obtain CH(S_1) and CH(S_2).

Step 3. Apply a merge algorithm to CH(S_1) and CH(S_2) to obtain CH(S) and halt.
```

- First presented in Preparata's and Hong's paper Convex Hull of a Finite Set of Points in Two and Three Dimensions
- Divide and Conquer technique

Wh

How

M/ha

MergeHull

BottomU

MergeHul

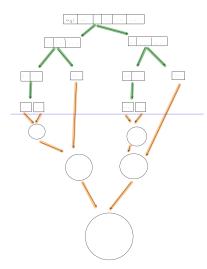
MergeHu

Results

References

MergeHull

General View



MergeHull

BottomU

MergeHul

MergeHul

Results

References

```
bool done = 0;
while (!done)
{
    done = 1;
    if(m>1)
    while (side(W[ix2], V[ix1], V[(ix1 + 1) % m])>= 0)
        ix1 = (ix1 + 1) % m;
    if(n>1)
    while (side(V[ix1], W[ix2], W[(n + ix2 - 1) % n])<= 0)
    {
        ix2 = (n + ix2 - 1) % n;
        done = 0;
    }
}</pre>
```

- Finding of upper tangent points
- Similar for lower

FAGUREL Petro

Why

LI.

M/h a

MergeHull

BottomUp MergeHull

Parallel MergeHull

Results

References

FAGUREL Petro

Wh

Wha

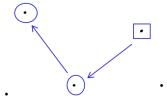
MergeHull

BottomUp MergeHull

Parallel

Reculte

References



Computational geometry Introduction to GPU programming: GPU accelerated Convex Hull

FAGUREL Petro

Wh

vvna

MergeHull

BottomUp MergeHull

Parallel MergeHui

Results

Reference



FAGUREL Petro

Wh

Wha

MergeHull

BottomUp MergeHull

Parallel

Reculto

References



Computational geometry Introduction to GPU programming: GPU accelerated Convex Hull

FAGUREL Petro

Wh

Hov

1 / / la a

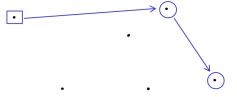
MergeHull

BottomUp MergeHull

Parallel

Recults

References



Why

How

MergeHull

BottomUp MergeHull

Parallel MergeHul

Results

Reference

Parallel MergeHull Problems

We want to decrease the number of merge operations. **Original MergeHull problems**:

- riginai iviergenuii problems
- The smallest subset size is not fixed.
 It can only be bound.
- Two arbitrary sets can not be merged.
 They have to be convex hulls.

Why

How

\//h =

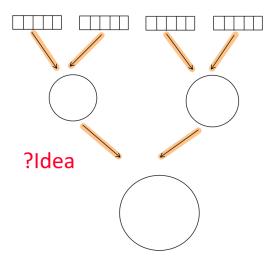
MergeH

BottomUp MergeHull

iviergeHuii

References

BottomUp MergeHull



computation **FAGUREL** Petro

BottomUp MergeHull

BottomUp MergeHull

Algorithm

```
for (int sz = d_size; sz < size; sz = sz + sz)
        for (int lo = 0; lo < size - sz; lo += sz + sz)
```

No merge yet.

Convex Hull

.

3.0.71

Managhla

BottomUp MergeHull

Parallel

Results

Reference

BottomUp MergeHull

Algorithm

OK.

First compute convex hulls. (e.g Jarvis March)

Wh

Но

3.0.71

Managh

BottomUp MergeHull

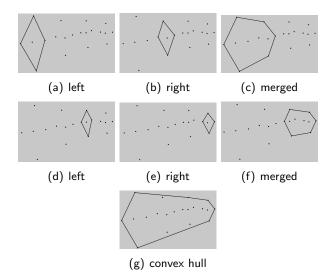
Parallel

Results

References

BottomUp MergeHull

Example



geometry
Introduction
to GPU
programming:
GPU
accelerated
Convex Hull
computation

Computational

FAGUREL Petro

\//by

iviergerit

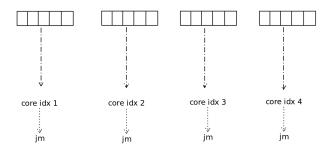
MergeHul

Parallel MergeHull

Results

Reference

Parallel MergeHull



Convex Hull

computation **FAGUREL**

Petro

Parallel MergeHull

Parallel MergeHull **GPUA**

```
jm_gpua();
for (int sz = d_size; sz < size; sz = sz + sz)
        for (int lo = 0; lo < size - sz; lo += sz + sz)
                         merge ( . . . );
```

Why

Нои

vvnat

MergeH

BottomUp MergeHul

Parallel

Results

References

Results Preliminary

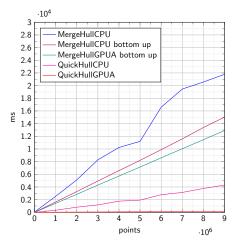


Figure: Execution on I7 4790k GTX980

FAGUREL Petro

Why

LI.

Wha

MergeH

BottomU

Davallal

Results

References

Results

Parameter change

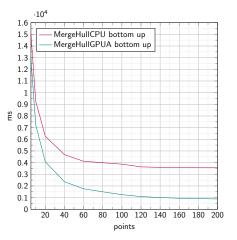


Figure: dsize change over 9 million points

FAGUREL Petro

Why

How

VVhat

MergeH

BottomU_I

Parallal

Results

Reference

Results

MergeHull vs QuickHull

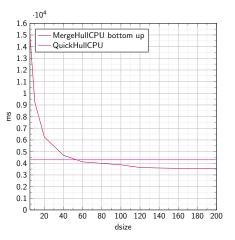


Figure: dsize threshold

FAGUREL Petro

\//by

\//ba

Mergel

BottomU

Parallel

MergeHu

Results

References

Results

MergeHull vs QuickHull

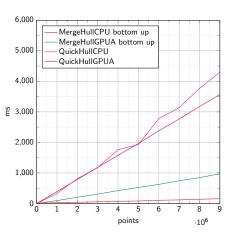


Figure: Execution on I7 4790k GTX980 with dsize=160

FAGUREL Petro

Why

. .

\//b

MergeH

BottomU

MergeHul

Parallel MergeHu

Result

References

- [1] MYO NeuralNet. Learning MNIST with GPU Acceleration A Step by Step PyTorch Tutorial. 2017. URL: http://makeyourownneuralnetwork.blogspot.com/2017/05/learning-mnist-with-gpu-acceleration.html.
- [2] F.P. Preparata and Se Hong. "Convex Hull of a Finite Set of Points in Two and Three Dimensions". In:

 Communications of the ACM 20 (Feb. 1977), pp. 87-93.

 URL: https://www.researchgate.net/publication/
 234809559_Convex_Hull_of_a_Finite_Set_of_
 Points_in_Two_and_Three_Dimensions.