22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Chiba, Japan, Jan. 16-19, 2017

Towards Scalable and Efficient GPU-Enabled Slicing Acceleration in Continuous 3D Printing

Aosen Wang[1], Chi Zhou[2], Zhanpeng Jin[3] and Wenyao Xu[1]
[1]CSE of SUNY Buffalo
[2]ISE of SUNY Buffalo
[3]ECE of SUNY Binghamton

Outline

Outline

01 **Motivation**

02

03

04

05

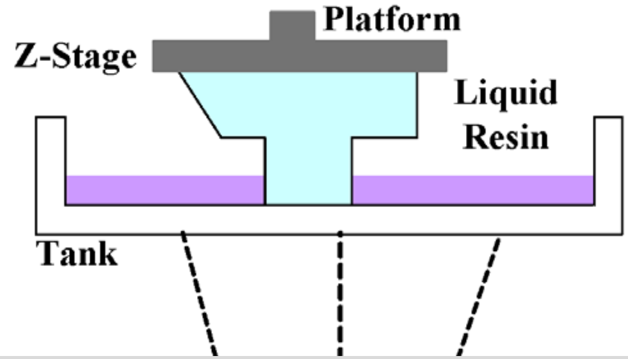
Continuous 3D Printing

- Continuous 3D printing is a recent technical breakthrough in additive manufacturing [2015]. (Carbon3D)

Two Orders of Magnitude on Printing Acceleration

* This picture comes from internet: https://techcrunch.com/2015/08/20/ with-100m-in-funding-carbon3d-will-make-3d-manufacturing-a-reality/

Principle of Continuous 3D Printing (Carbon 3D)

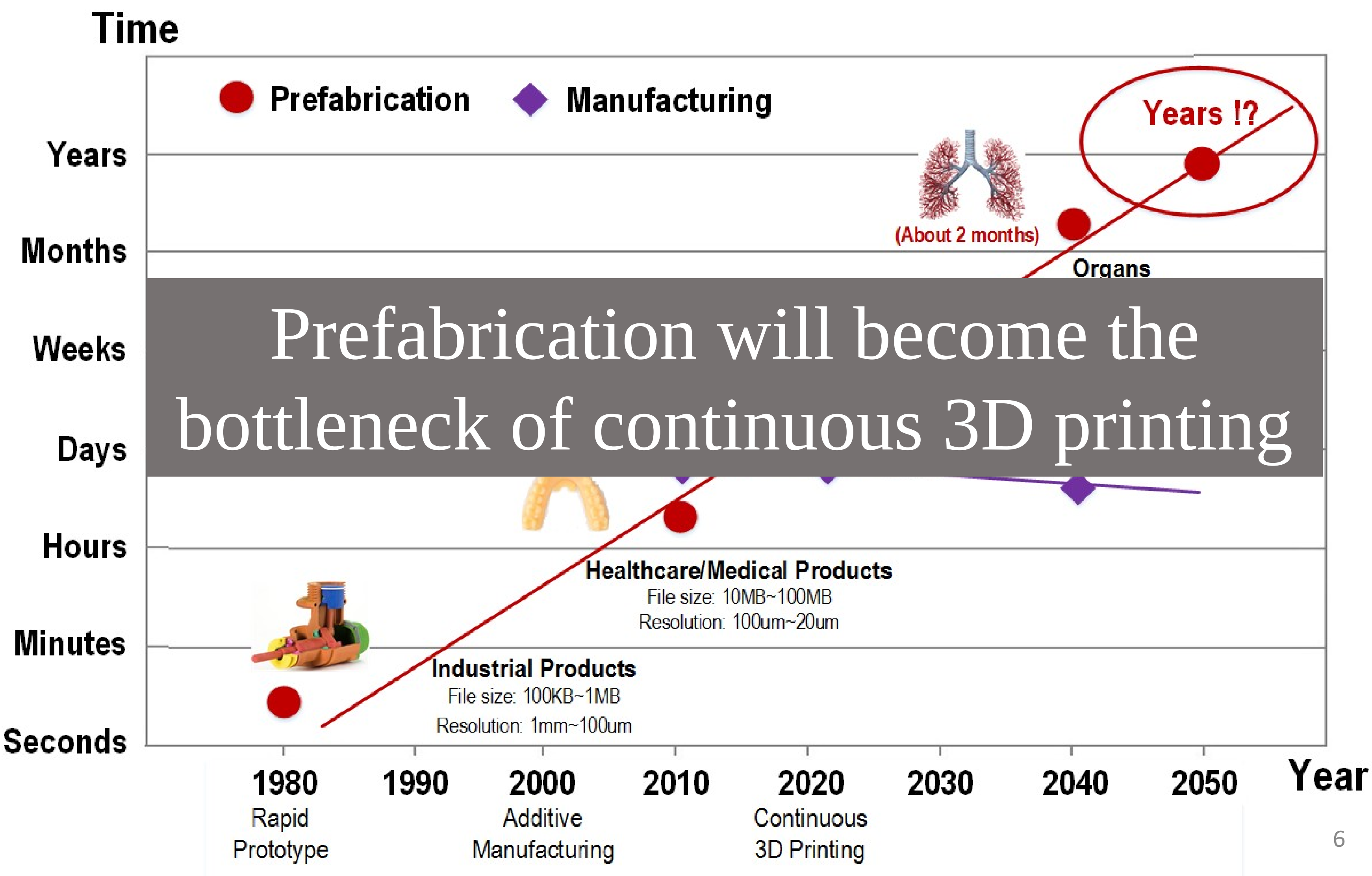


Speedup of Carbon 3D is mainly from Manufacturing (wet part)

- ✓ Dry Part (Prefabrication)
  Computing unit slices of the layer images.

- ✓ Wet Part (Manufacturing)
  Mechanical operations to fabricate 3D object from liquid materials.

Prefabrication V.S. Manufacturing



Prefabrication will become the bottleneck of continuous 3D printing

Outline

01

02 **Background - Slicing**

03

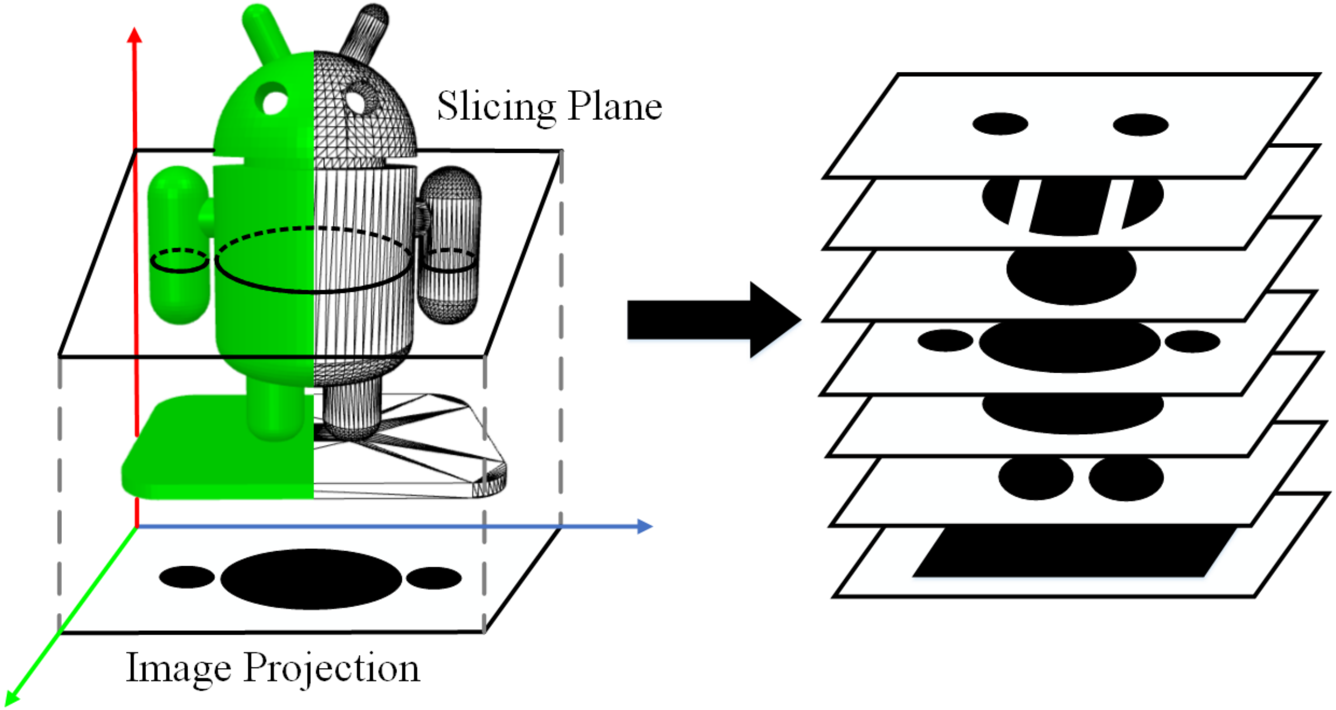
04

05

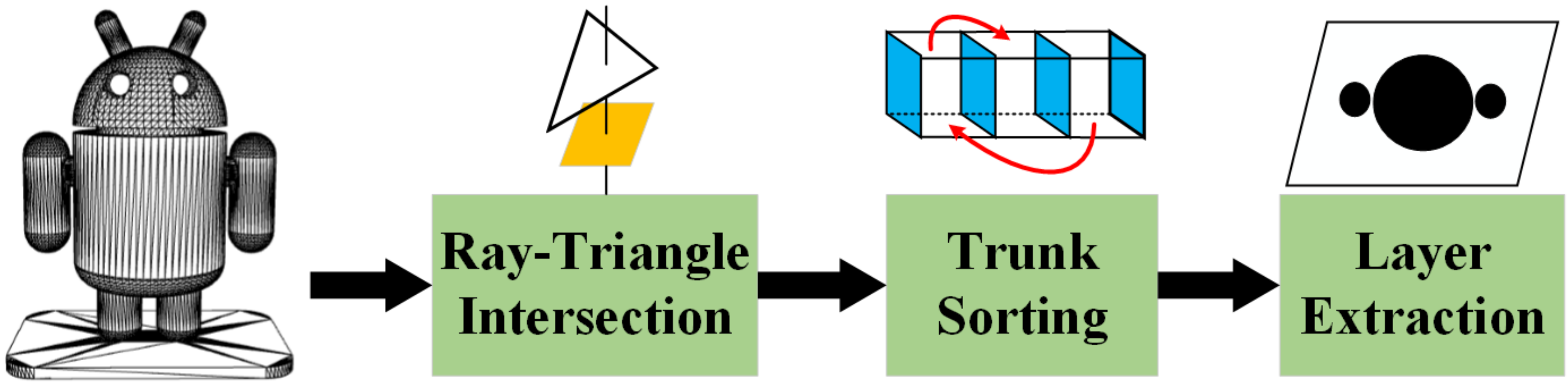
Background (Slicing)

- The task in prefabrication includes three sequential procedures, i.e., slicing, path planning and support generation. Slicing dominates time efficiency in “dry part”.

- In continuous 3D printing, image-mask-projection based slicing algorithm is employed. This pixel-independent processing enables massive parallel acceleration.



Methodology (Slicing Algorithm Analysis)



- The e… …triangle interse…

Trunk Sorting takes up a minor part of computation

  - … rays on image pixel center and the triangles from STL file.
  - The trunk sorting sorts the out-of-order intersection points by ascending order using the bubble sorting in the trunk of each pixel.
  - The binary value of each pixel on projected images is identified by incremental updating, so that the topology information is extracted for binary slicing image.

Outline

01

02

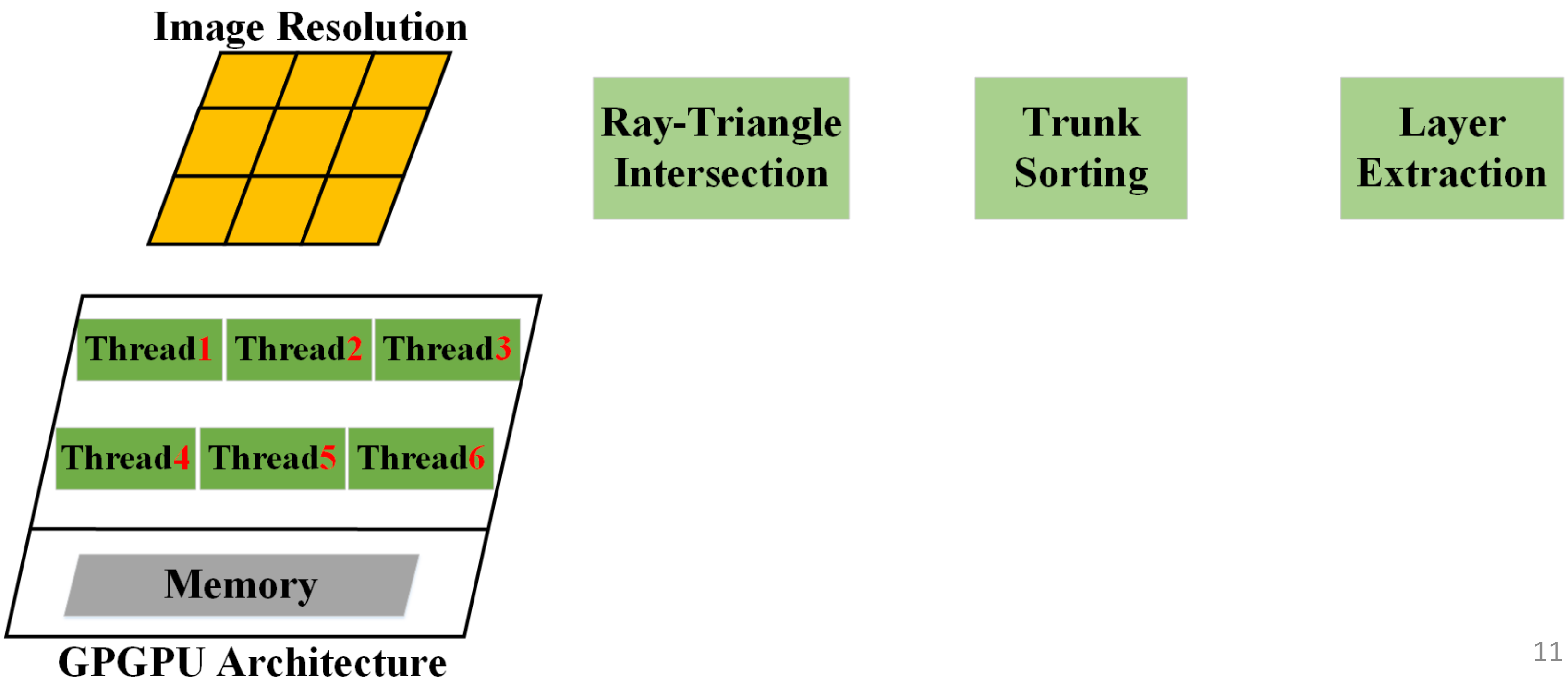
03 **GPU Acceleration**

04

05

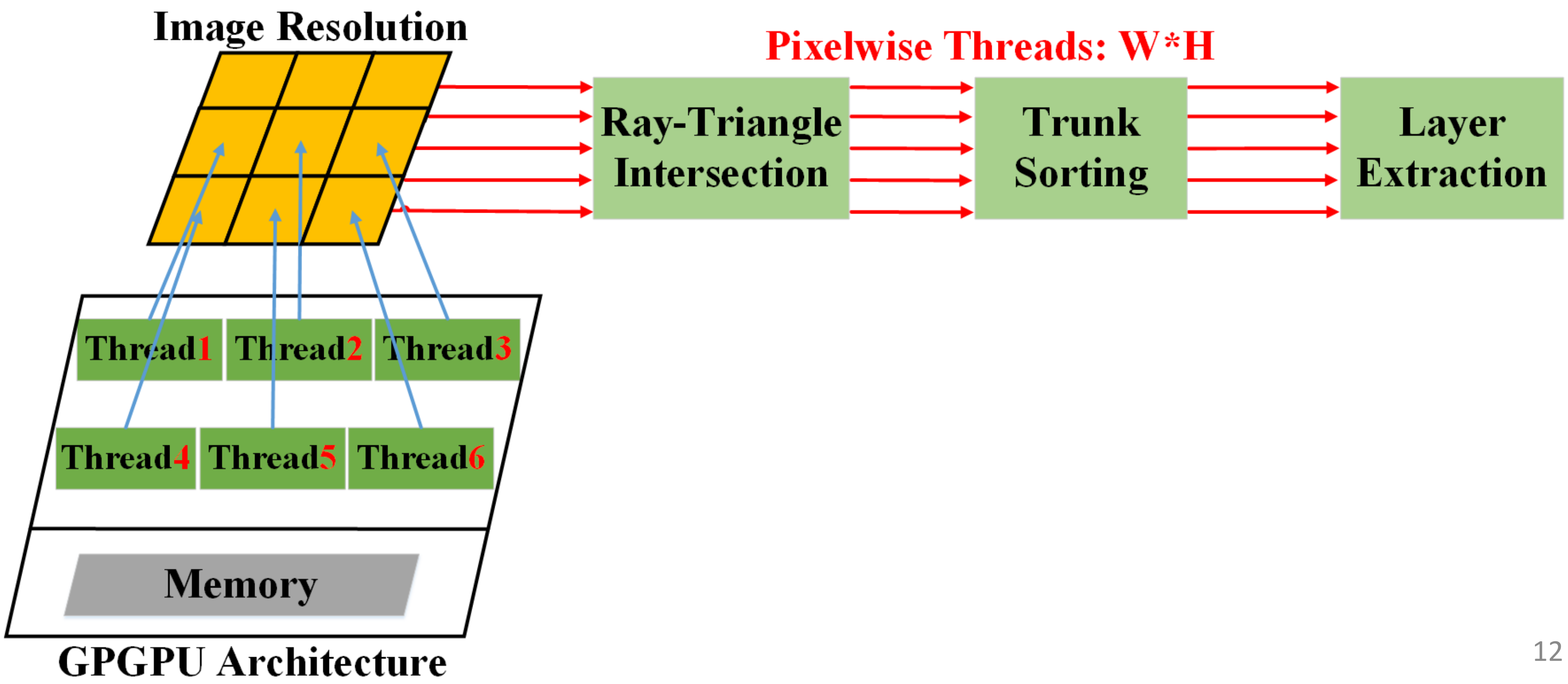
GPU-Enabled Slicing-I
(Pixelwise Parallel Slicing)

- By the sequential algorithm analysis, we exploit the pixelwise parallelism based on GPGPU architecture.



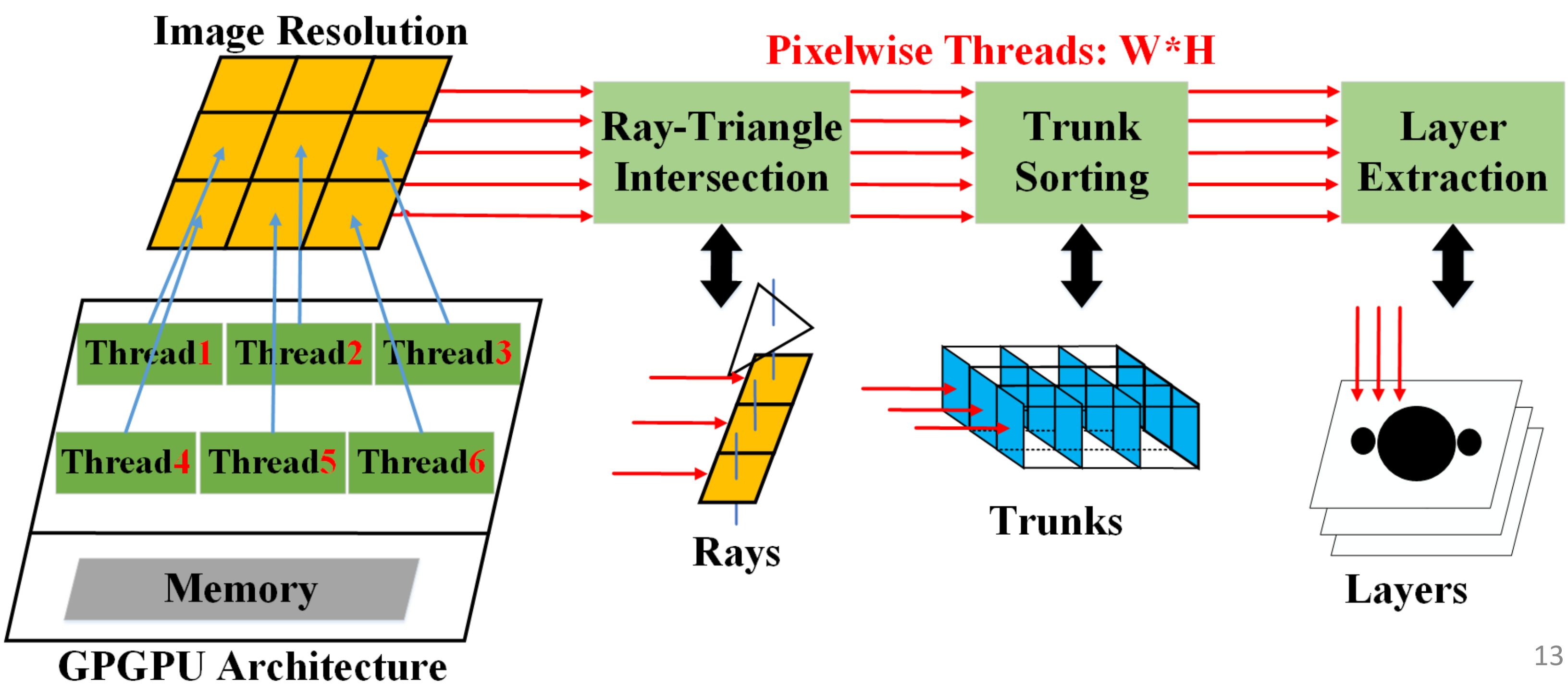
GPU-Enabled Slicing-I (Pixelwise Parallel Slicing)

- By the sequential algorithm analysis, we exploit the pixelwise parallelism based on GPGPU architecture.
- The entire processing in all three functional modules for one pixel is assigned to a specific thread.

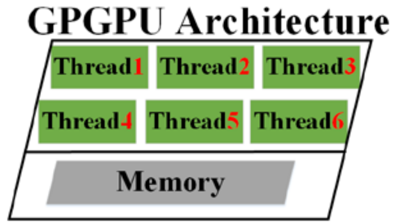


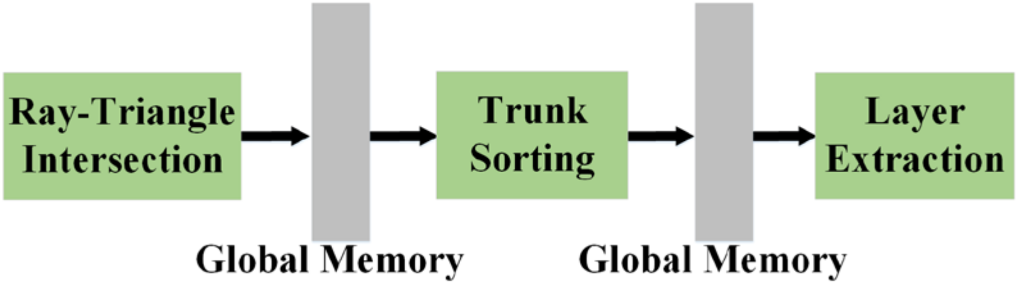
GPU-Enabled Slicing-I
(Pixelwise Parallel Slicing)

- By the sequential algorithm analysis, we exploit the pixelwise parallelism based on GPGPU architecture.
- The entire processing in all three functional modules for one pixel is assigned to a specific thread.
- Fully use of precious shared memory on GPU to reduce time-consuming global memory intersections.



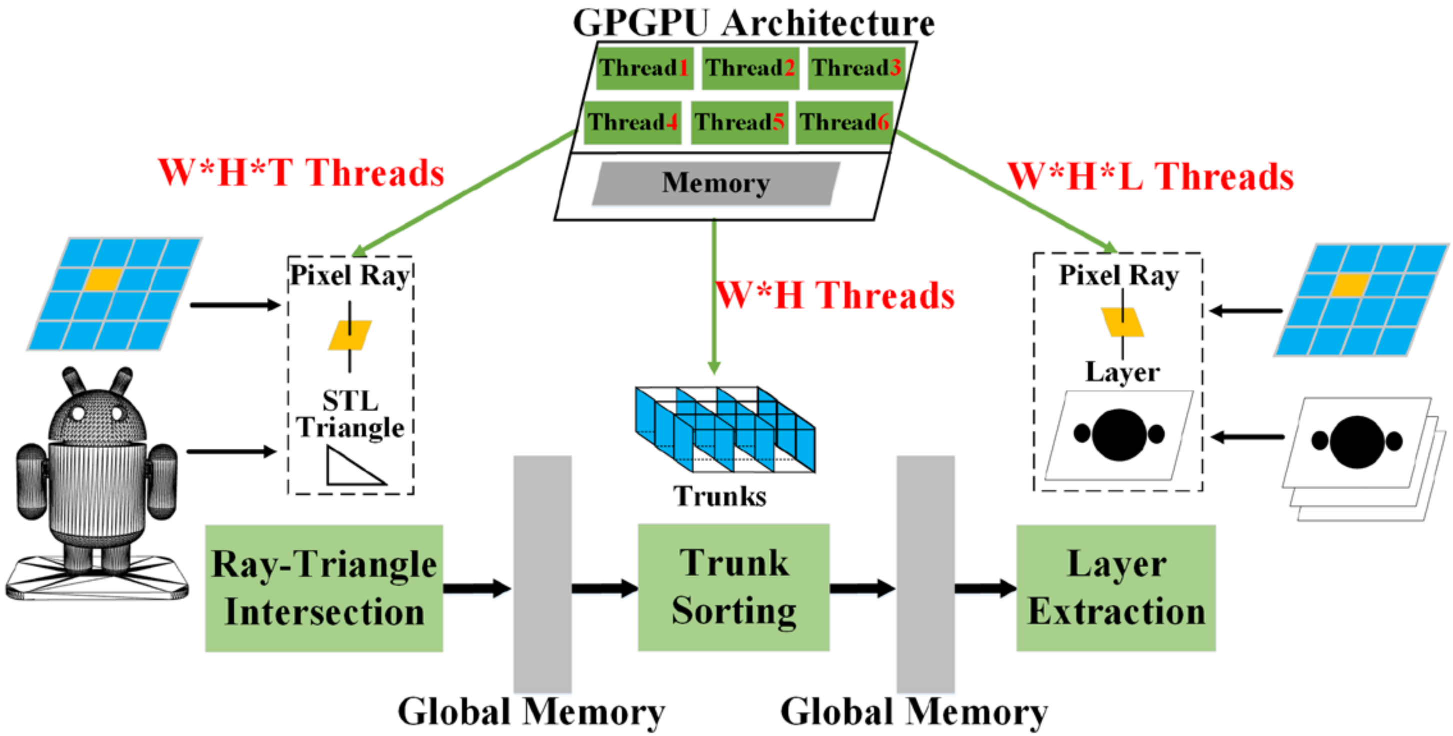
GPU-Enabled Slicing-II (Fully Parallel Slicing)





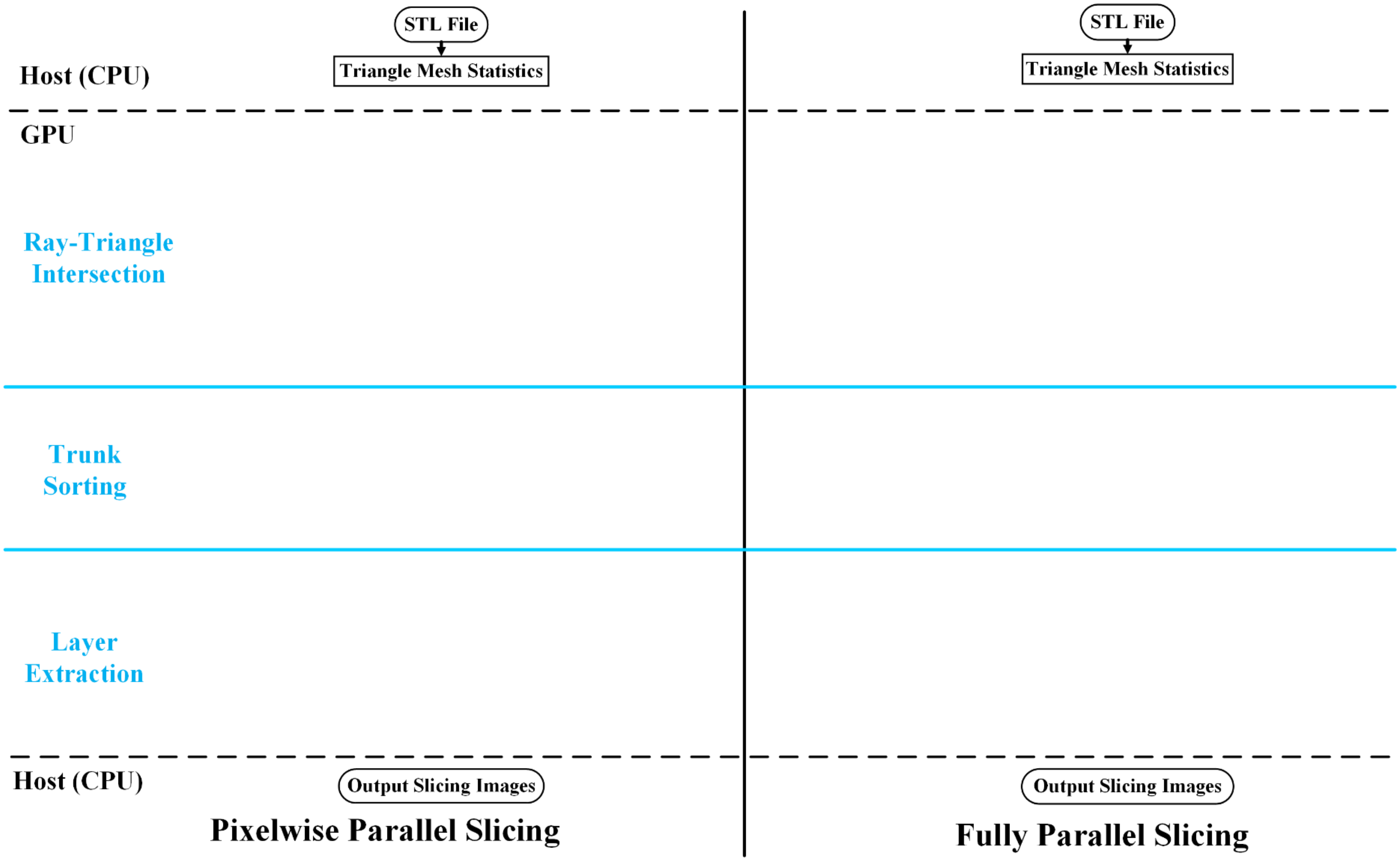
- PPS still has serial computing components.
- FPS explores the massive thread concurrency

GPU-Enabled Slicing-II (Fully Parallel Slicing)

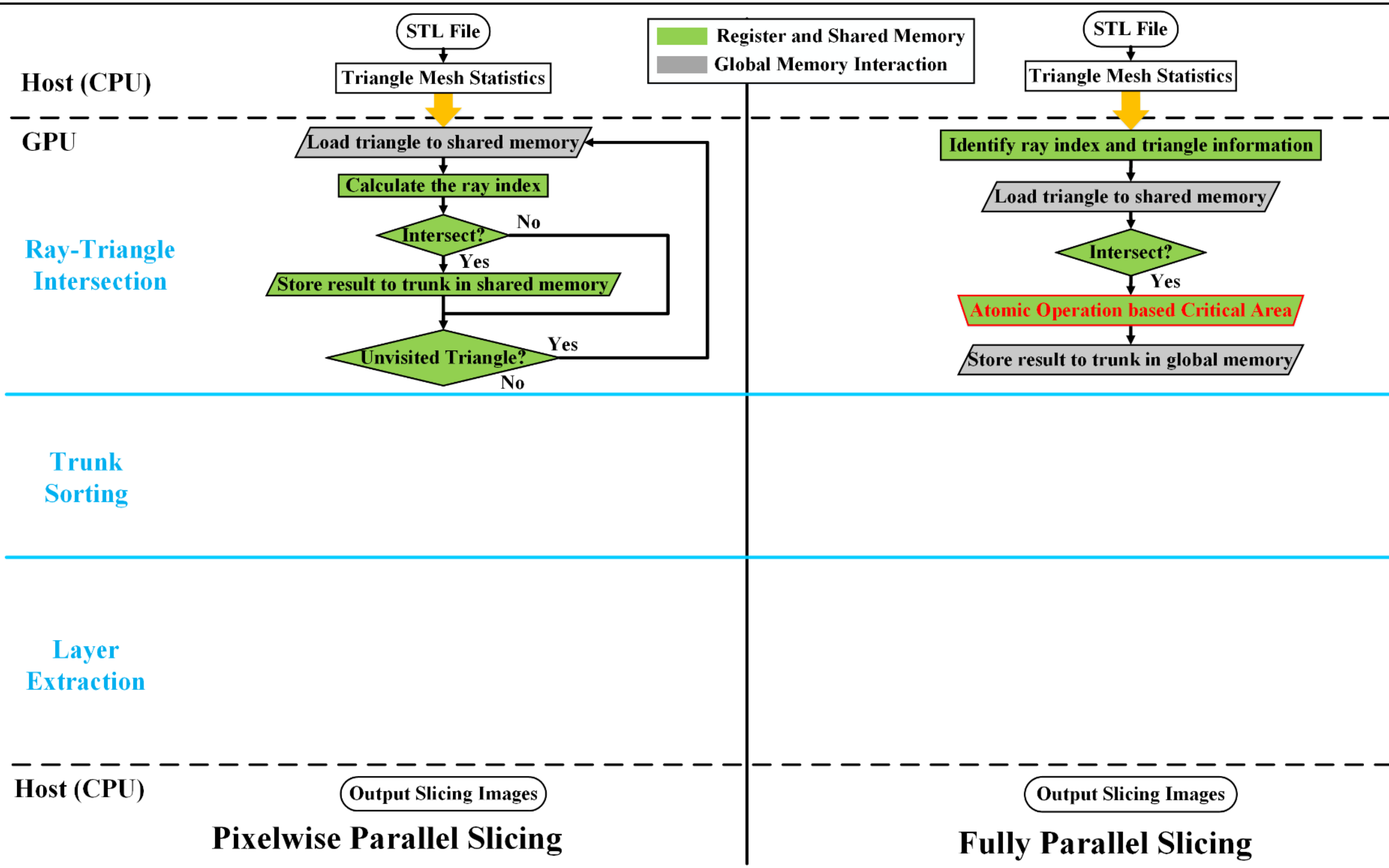


- PPS still has serial computing components.
- FPS explores the massive thread concurrency
- This method increases global memory accessing pattern, but is scalable for large-size problem.
- The issue of multi-thread memory writing conflict arises and can be addressed by atomic operation based critical area.

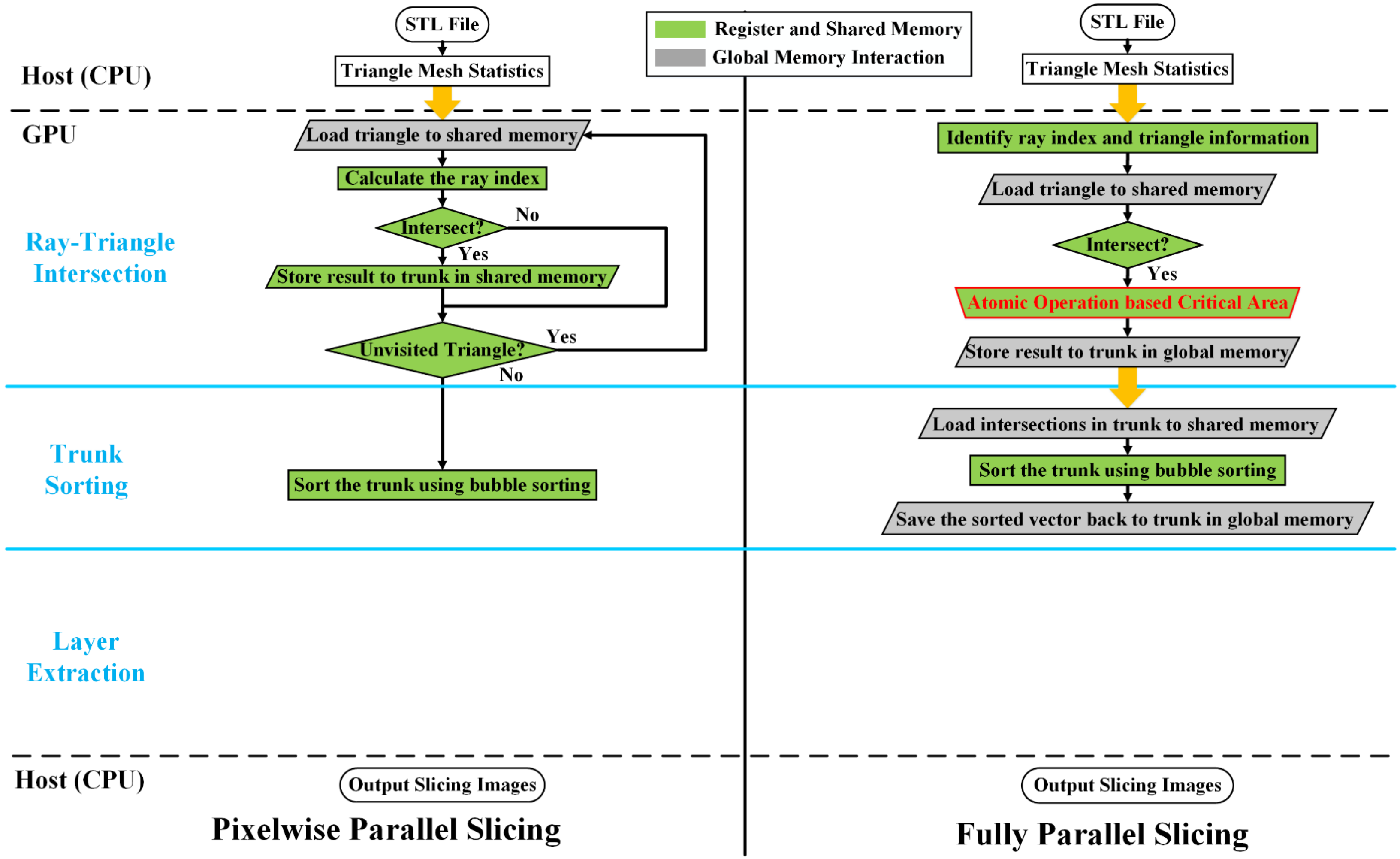
Comparison of Two GPU Implementations



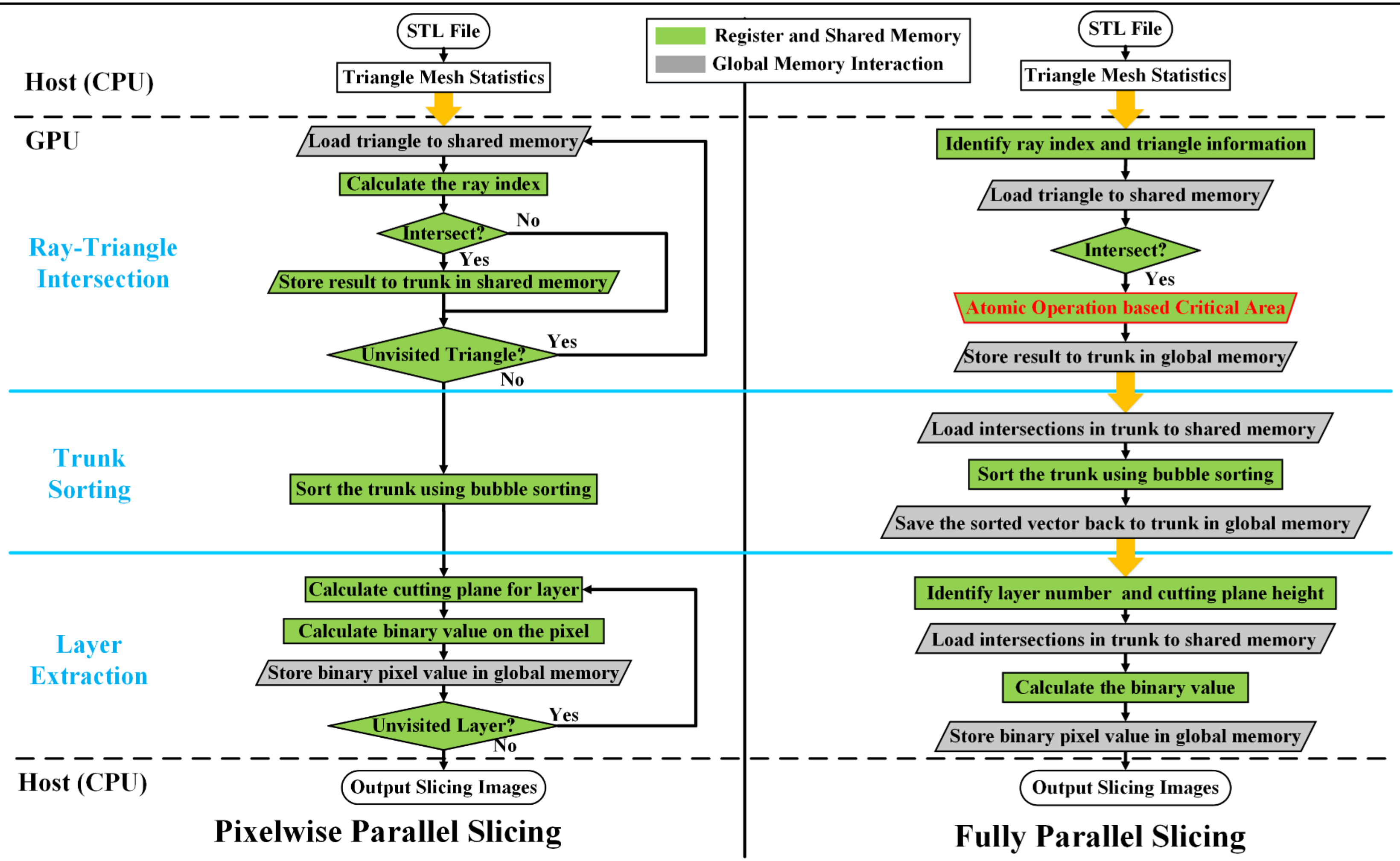
Comparison of Two GPU Implementations



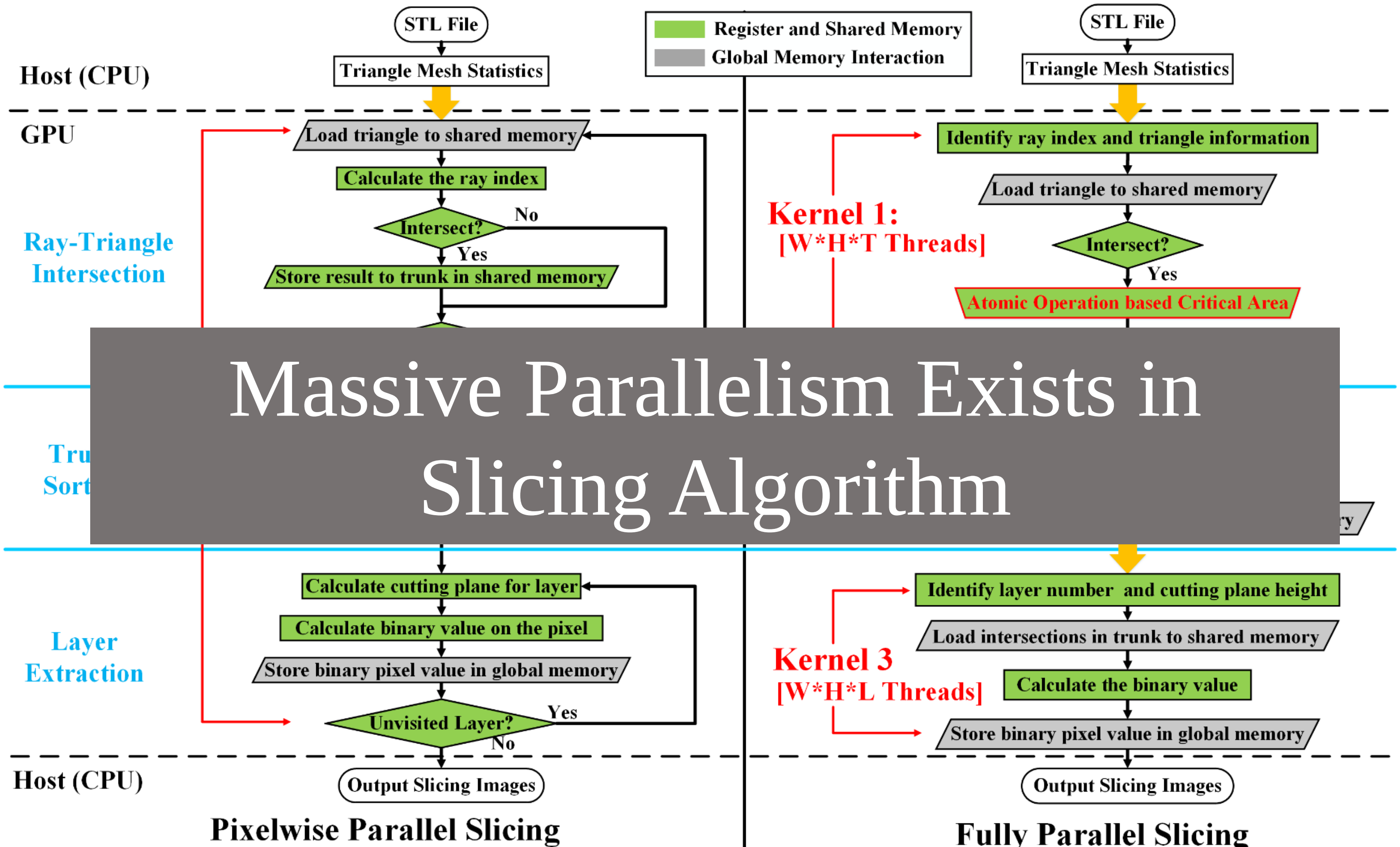
Comparison of Two GPU Implementations



Comparison of Two GPU Implementations



Comparison of Two GPU Implementations



Massive Parallelism Exists in Slicing Algorithm

- **PPS**: all tasks in fast shared memory, less global memory access, no multi-thread conflict.
- **FPS**: recycle-free processing, atomic operation based critical area to address conflict issue.

Outline

01

02

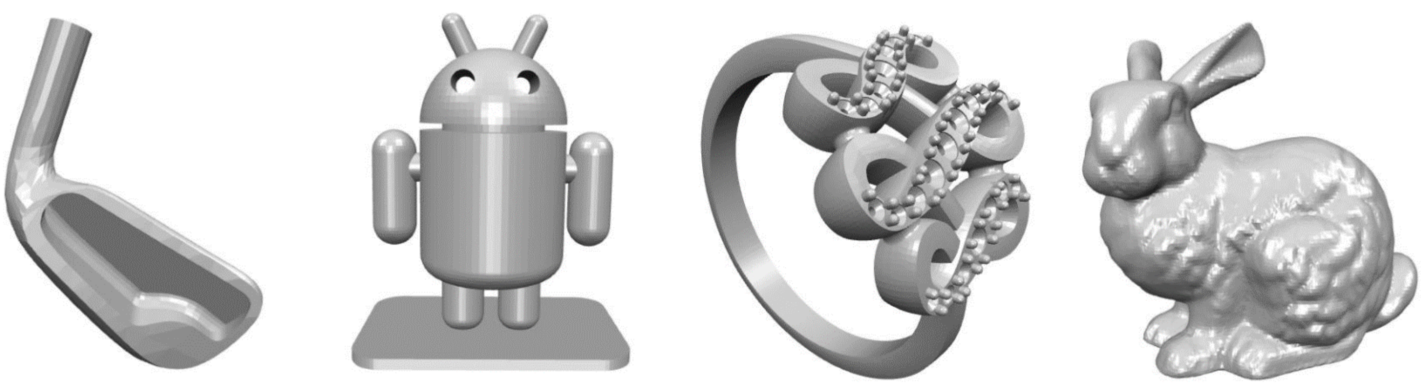
03

04 **Experiments**

05

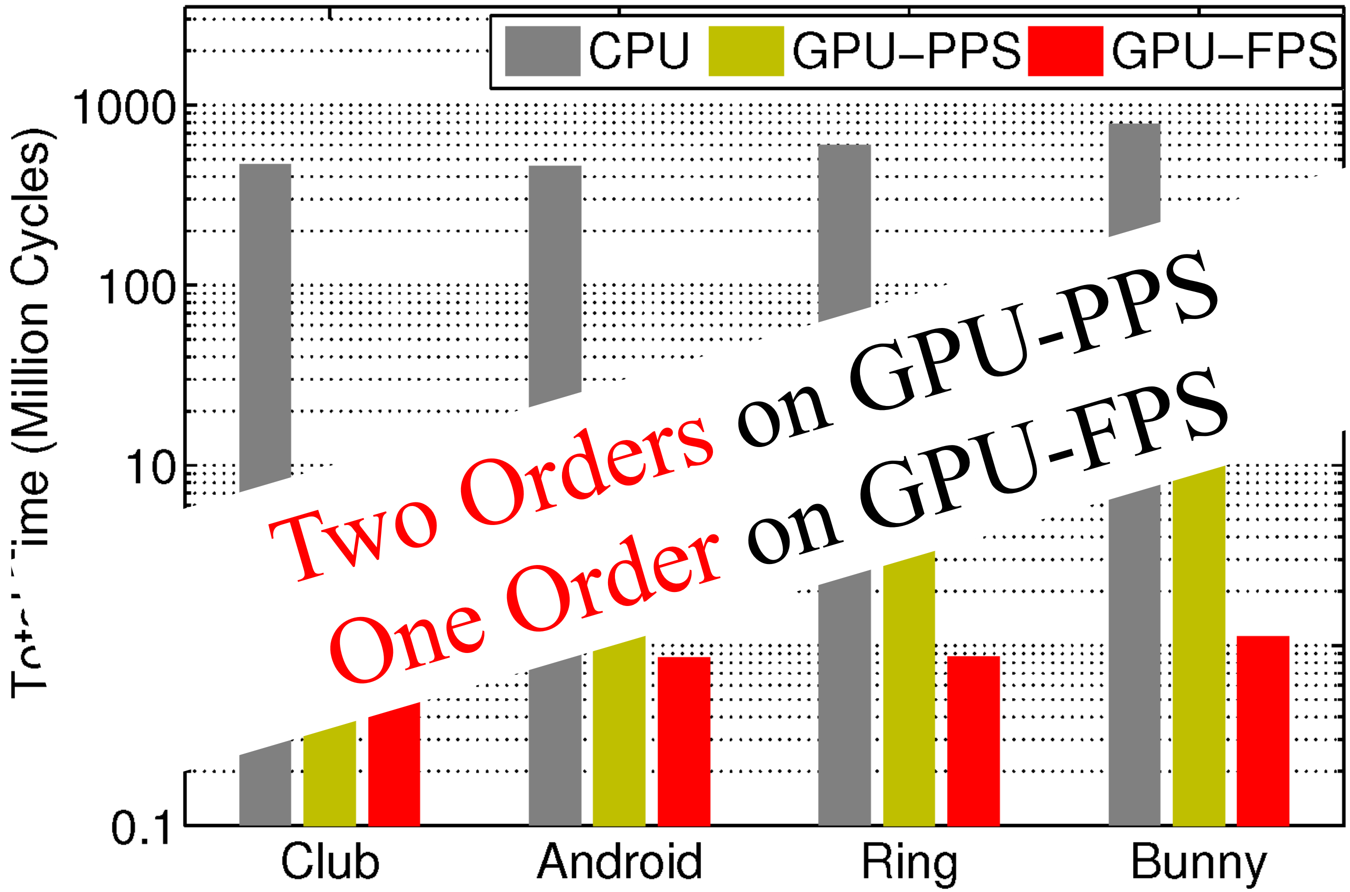
Experimental Setup

- We use cycle-accurate simulators for CPU and GPU computing platforms
- Sniper is a typical simulator for x86 architecture and GPGU-Sim is a good simulating tool to check statistics of GPGPU architecture.
- Sniper is configured as Intel Xeon X5550 with 2.66GHz frequency while GPGPU-Sim is configured as Nvidia Geforce GTX480 with 700MHz.
- We choose four representative 3D objects: Club, Android, Ring and Bunny. They have different triangle mesh size, as 3290, 10926, 33730 and 69664.



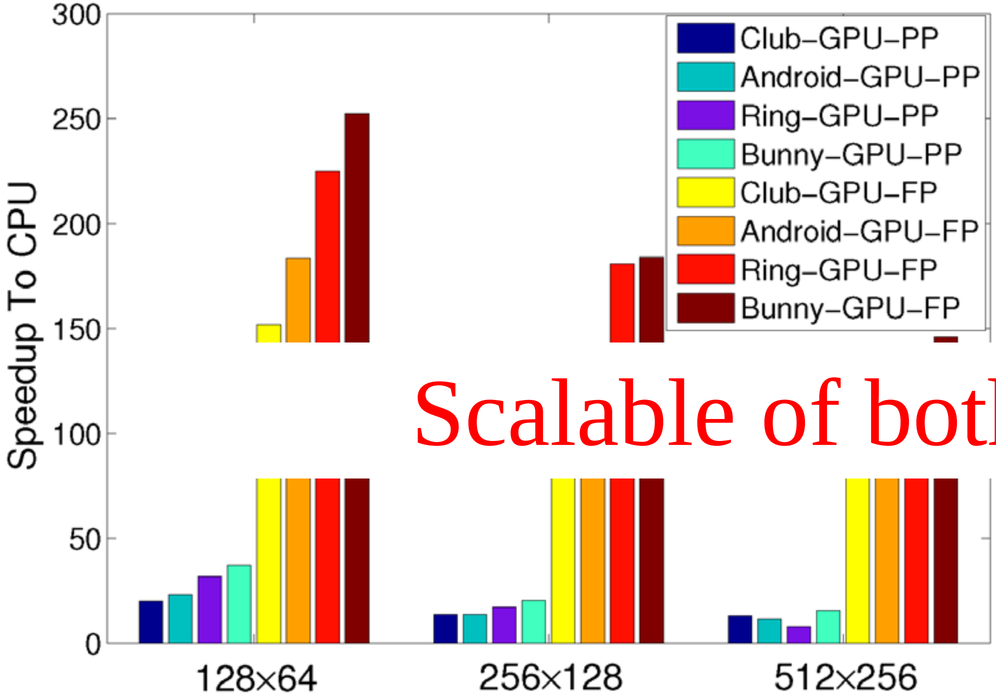
(a) Club (b) Android (c) Ring (d) Bunny

Experiment: Time Efficiency



- Fully parallel slicing achieves the best performance in three schemes.
- Considering the processing frequency difference, PPS gains one order of magnitude improvement and FPS even obtains two orders acceleration.

Experiment: Scalability



Time (Million Cycles)		Layer Number 10	100	1000
CPU	Ray-Triangle	517.38	517.34	517.51
	Trunk Sorting	12.06	11.99	12.08
	Layer Extract	**33.11**	**255.32**	**2459.17**
	Total	562.55	784.65	2988.76
[illegible]	Ray-Triangle	0.300	0.301	0.300
	Trunk Sorting	0.020	0.020	0.020
	[illegible]	[illegible]	[illegible]	**7.965**
	[illegible]	[illegible]	[illegible]	8.285

Scalable of both PPS and FPS

- We choose three image resolutions: 128*64, 256*128 and 512*256.
- PPS holds one order of magnitude speedup and FPS achieves about two orders time efficiency compared to CPU.

- We choose three layer numbers: 10, 100 and 1000.
- FPS scheme on GPU can achieve about two orders time efficiency compared with CPU case.
- As layer number increases, layer extraction dominates the entire runtime.
- Trunk Sorting takes a subtle proportion.

Outline

01

02

03

04

05 **Conclusions**

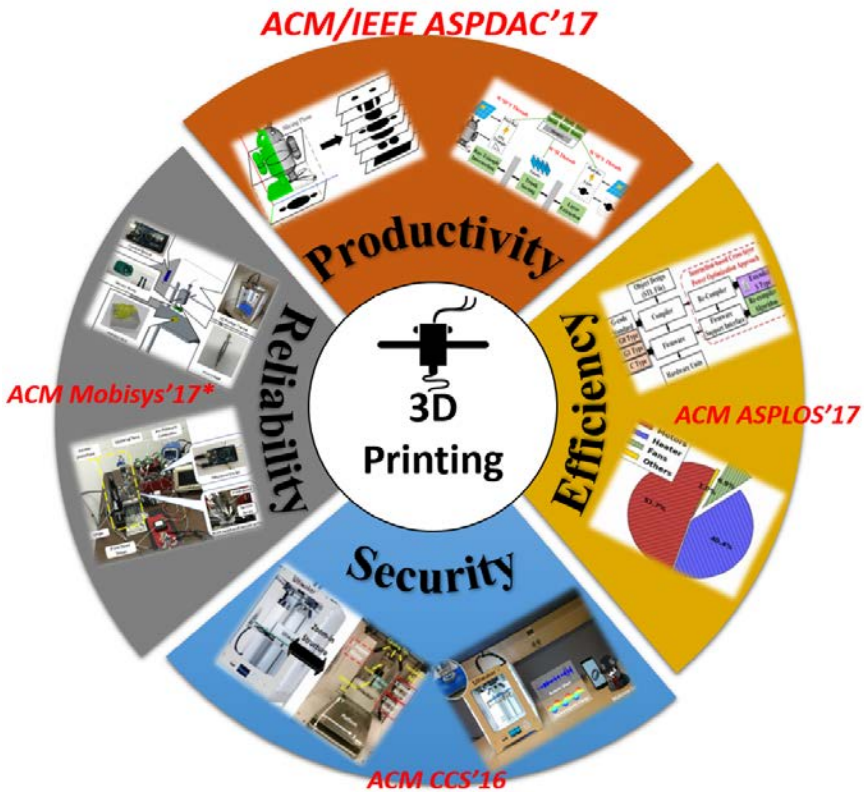
Conclusions

- We investigated slicing algorithm acceleration on GPGPU architecture for continuous 3D printing.
- We developed pixelwise parallel slicing and fully parallel slicing implementations.
- Experiments demonstrate the effectiveness and scalability of our implementation.

In the future:

- We will design new implementations on the new hardware platform, such as FPGA or more powerful GPU.
- We will exploit pipeline property between prefabrication and manufacturing.

Q & A



Thank you!

Address comments to wenyaoxu@buffalo.edu