
WGK
2011

Multithreaded rendering using OpenGL in Task Pooled Environment

Karol Gasiński

2011

Agenda

- Introduction
  - Multicore is now
  - Everything is about streaming
- Idea of multithreaded rendering
  - Changing state
  - Loading resources
  - Compiling shaders
  - Synchronization
- Future research
- Summary

Agenda

- **Introduction**
  - **Multicore is now**
  - **Everything is about streaming**
- Idea of multithreaded rendering
  - Changing state
  - Loading resources
  - Compiling shaders
  - Synchronization
- Future research
- Summary

Introduction

Multicore is now

- At the beginning everything was simple
- One CPU, one process, one thread with one game loop..

Introduction

Multicore is now

- But PC's are evolving and now almost every CPU has several cores, that support several threads…
- So good old times are over and we need to think into the future with our designs.
- Putting rendering on second thread, physics and sound mixing on another ones sounds like a good idea to utilize additional cores..

Introduction

Multicore is now

- ...but in fact it works fine only with machines that have up to four cores.
- Even then, synchronization of such solution is a big challenge and can change into a nightmare.

Introduction

Multicore is now

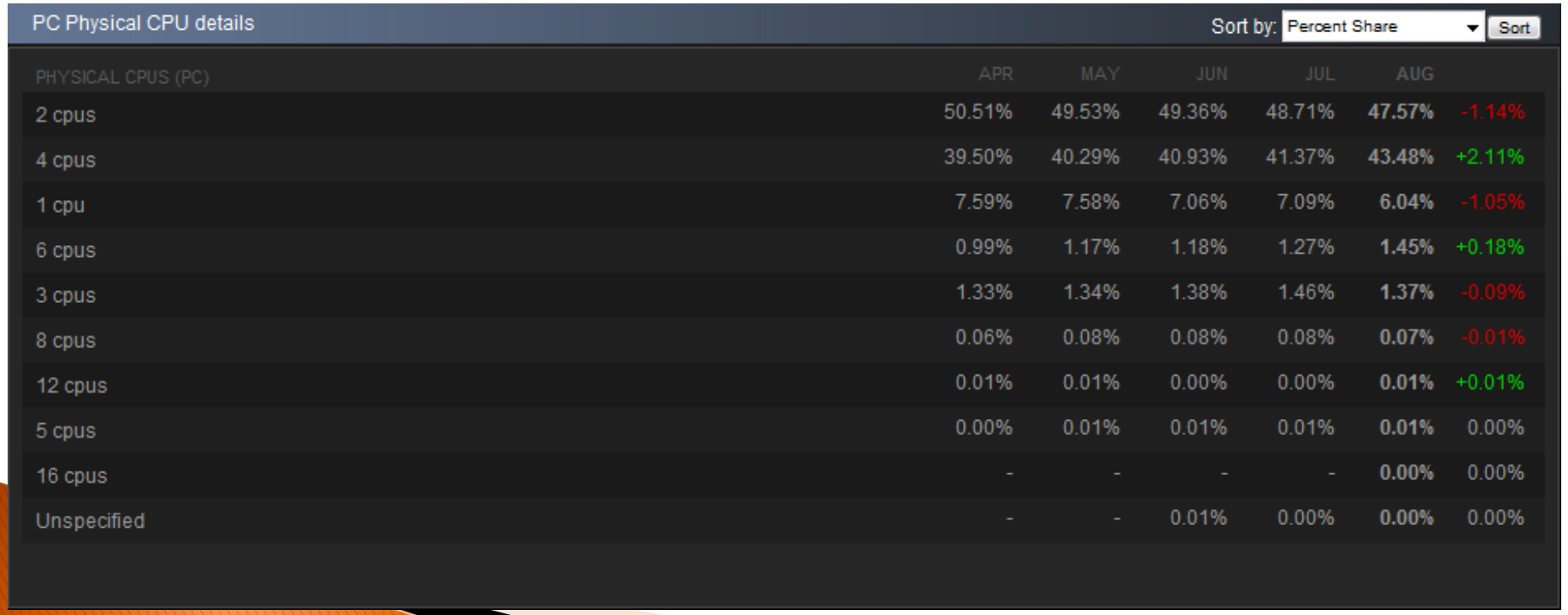
- So, how to prepare for upcoming many core era?
- Old guy's from mainframes already know that the solution is evolution:
  - from scheduling processess
  - through scheduling threads
  - **to scheduling tasks !**

- **Tasks Pooling is future!**

Introduction

Multicore is now

- In conclusion writing single threaded application is like riding a horse on a highway! (insane!)



PC Physical CPU details

Sort by: Percent Share Sort

PHYSICAL CPUS (PC)	APR	MAY	JUN	JUL	AUG	
2 cpus	50.51%	49.53%	49.36%	48.71%	**47.57%**	-1.14%
4 cpus	39.50%	40.29%	40.93%	41.37%	**43.48%**	+2.11%
1 cpu	7.59%	7.58%	7.06%	7.09%	**6.04%**	-1.05%
6 cpus	0.99%	1.17%	1.18%	1.27%	**1.45%**	+0.18%
3 cpus	1.33%	1.34%	1.38%	1.46%	**1.37%**	-0.09%
8 cpus	0.06%	0.08%	0.08%	0.08%	**0.07%**	-0.01%
12 cpus	0.01%	0.01%	0.00%	0.00%	**0.01%**	+0.01%
5 cpus	0.00%	0.01%	0.01%	0.01%	**0.01%**	0.00%
16 cpus	-	-	-	-	**0.00%**	0.00%
Unspecified	-	-	0.01%	0.00%	**0.00%**	0.00%

Introduction

Everything is about streaming

- Second important thing is fact of streaming resources.
- Games became so complicated that they cannot and they won't be able anymore to hold all required resources loaded at the same time.
- Therefore dynamic streaming of resources becomes a must have feature in upcoming years.

Agenda

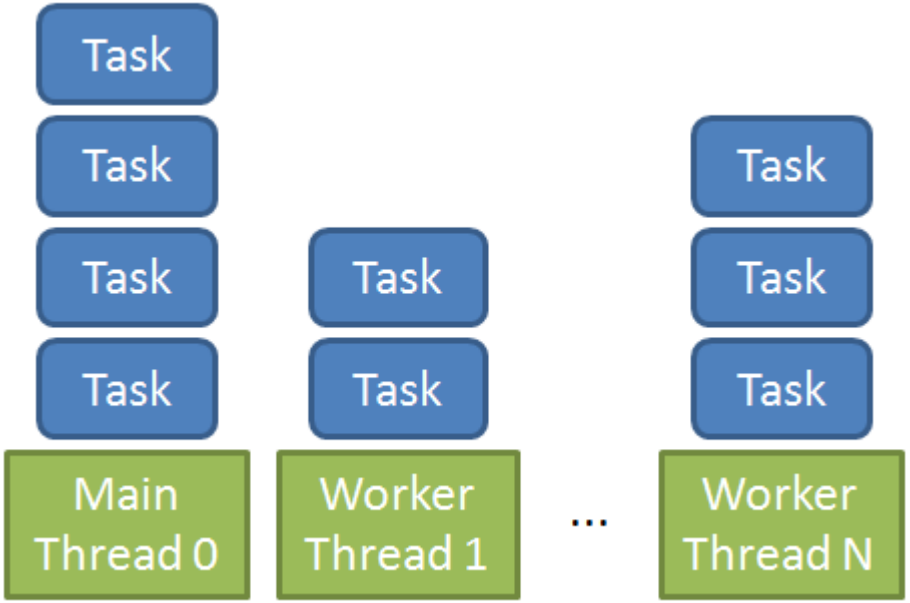
- Introduction
  - Multicore is now
  - Everything is about streaming
- **Idea of multithreaded rendering**
  - **Changing state**
  - **Loading resources**
  - **Compiling shaders**
  - **Synchronization**
- Future research
- Summary

Idea of multithreaded rendering

- Task Pooling system is base of the engine.
- It can be written from scratch (like in my approach) or other solution can be used, like Intel TBB library.

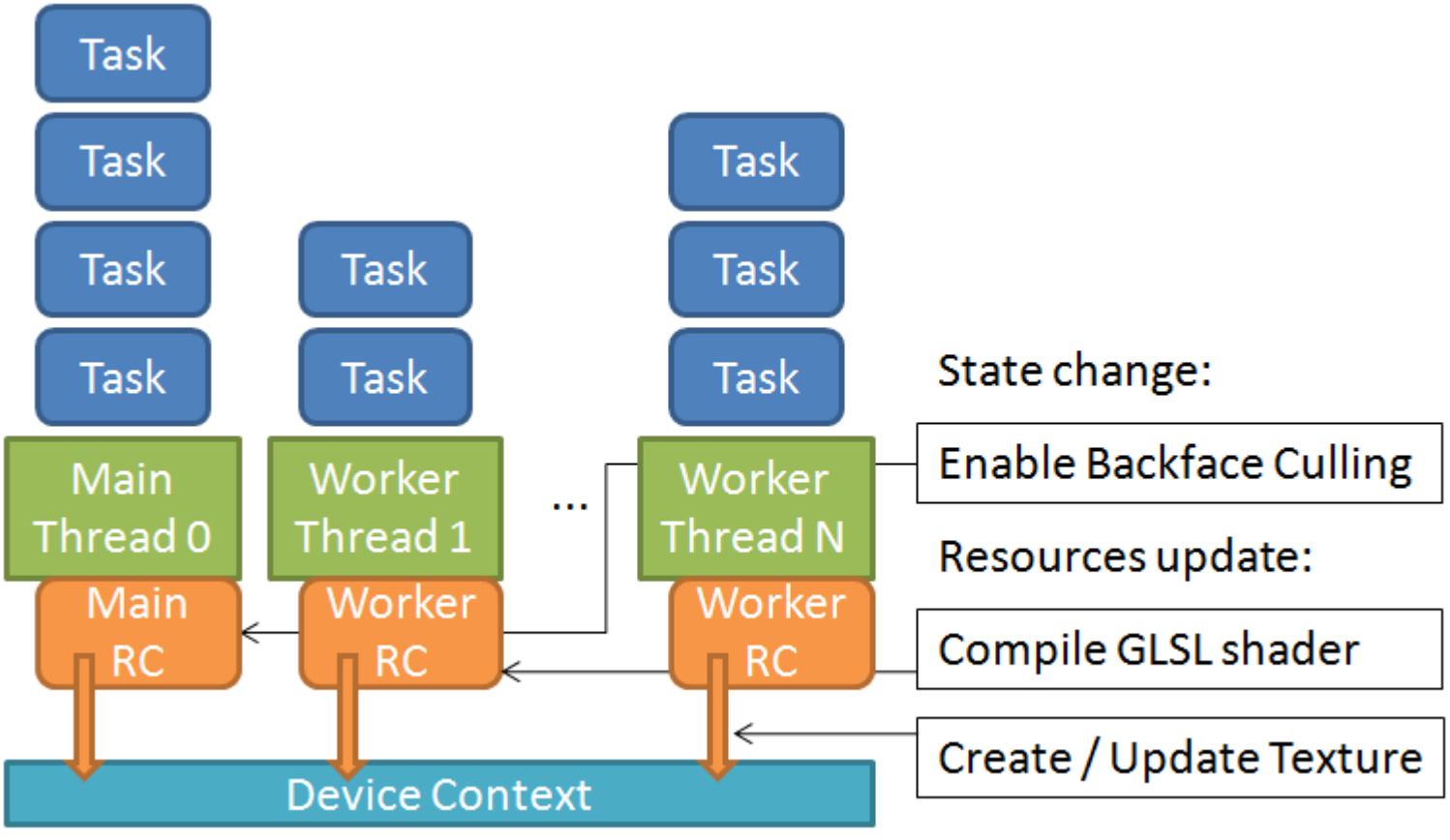
Idea of multithreaded rendering

- Base of Task Pooling system is idea of worker threads created per CPU core that pitch work from the queue and execute it one by one.



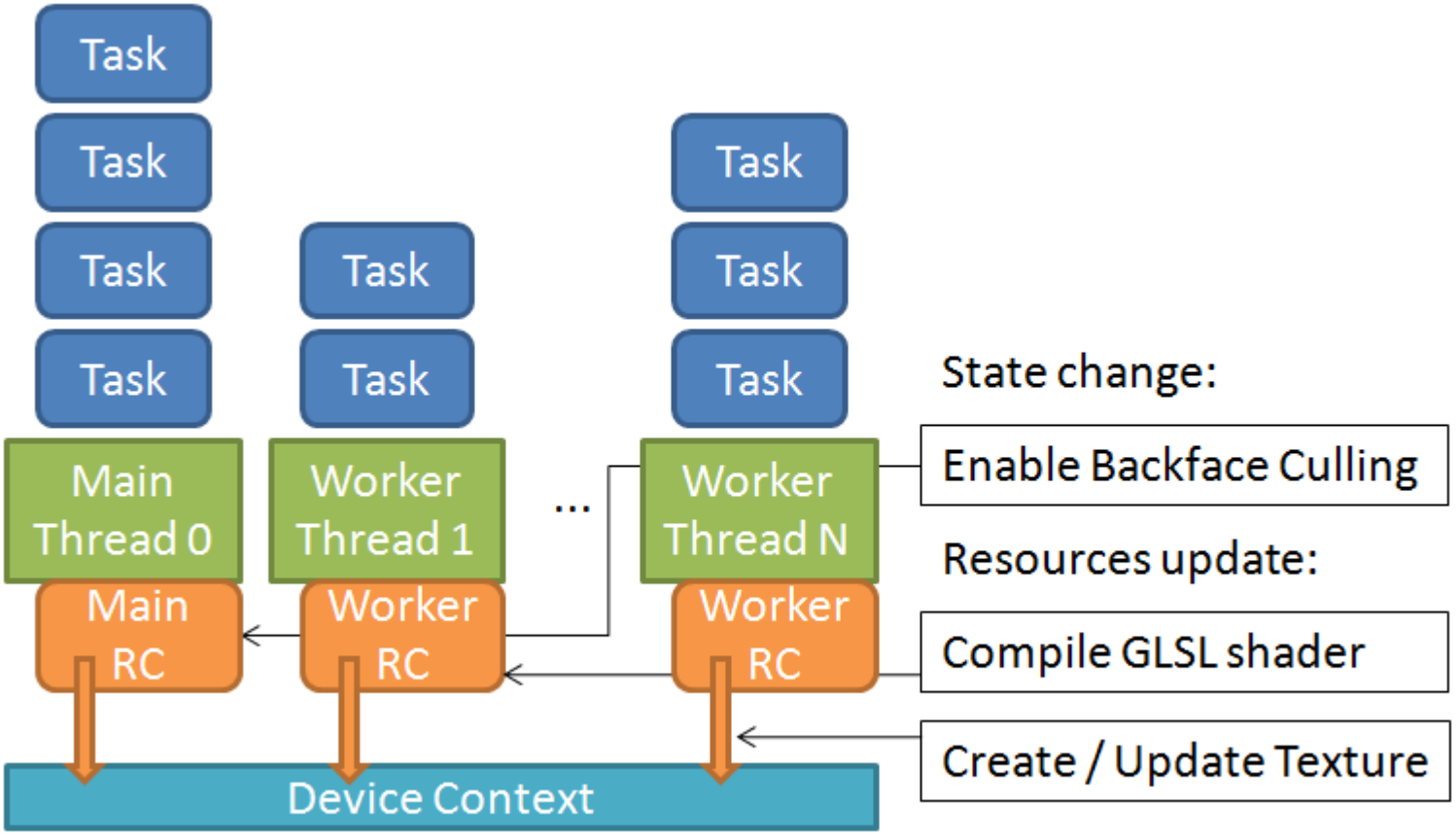
Idea of multithreaded rendering

- Multithreaded rendering aproach presented in this lecture is extending this architecture.



Idea of multithreaded rendering

- Similiar to worker threads we create main RC and worker RC's.



Idea of multithreaded rendering

Changing state

- Because we operate in Task Pooled environment we need to have freedom to composite state from all cores at the same time.
- In DX11 we have Deffered Commands that are agregated on worker threads and then executed on main thread.
- We want to have the same functionality in OpenGL ☺

Idea of multithreaded rendering

Changing state

- Solution to this problem is software implementation that emulates DX11 like behavior.
- For each object we create bitfield Key that represents most important state changes required to render it.
- We also cache shader variables values in data structures on application side.

Idea of multithreaded rendering

Changing state

- Then we perform compositing stage in which all <key,pointer> pairs are sorted like normal integer numbers.
- On high bits we have information about most time consuming changes, and on low bits that less important ones.
- After sorting phase is finished, we can iterate through the list of keys and perform rendering.
- It will have guaranteed minimum set of state changes.

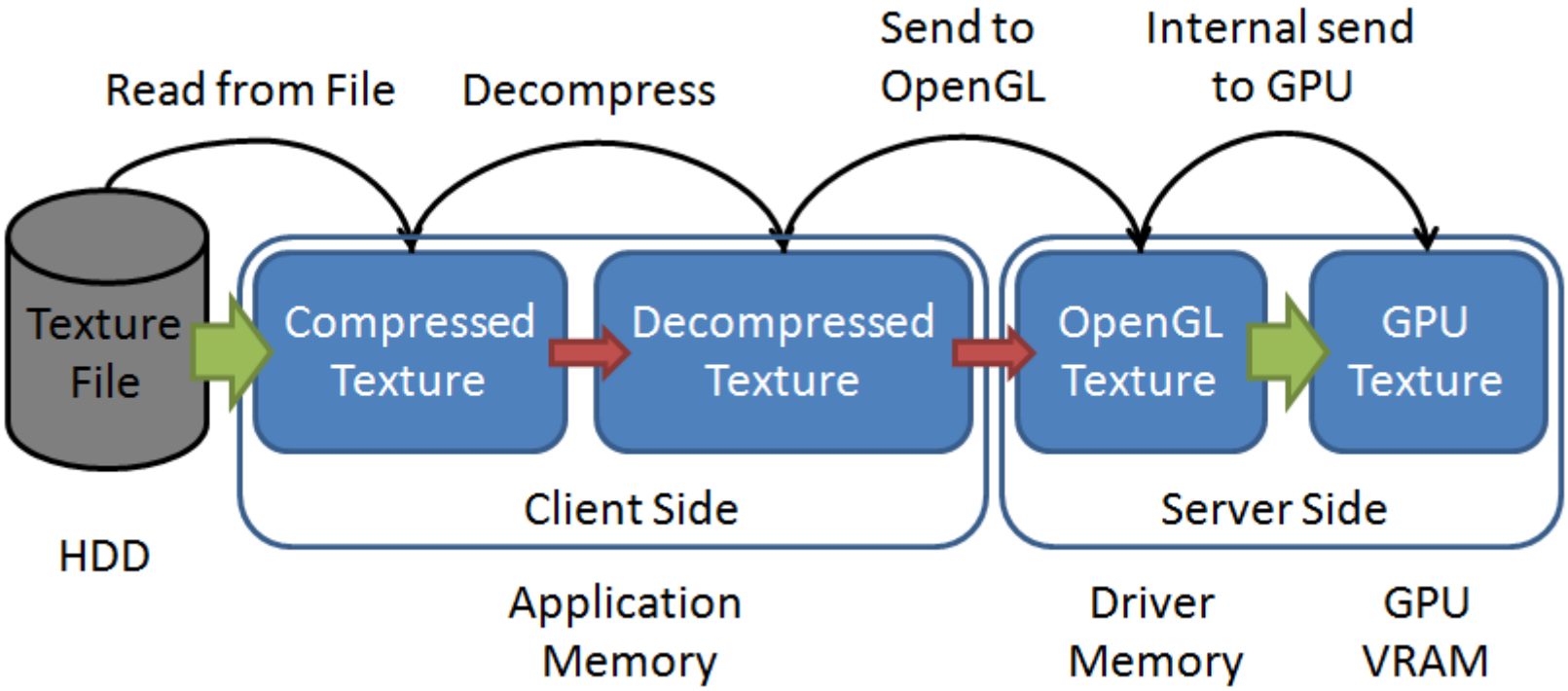
Idea of multithreaded rendering

Loading resources

- When we are able to manage state on all worker threads then it is time for most important part.
- Distributing long term operations on separate worker threads.
- First such behavior that comes in mind is loading resources (textures and geometry).
- This operations are complicated and time consuming so they are ideal for splitting.

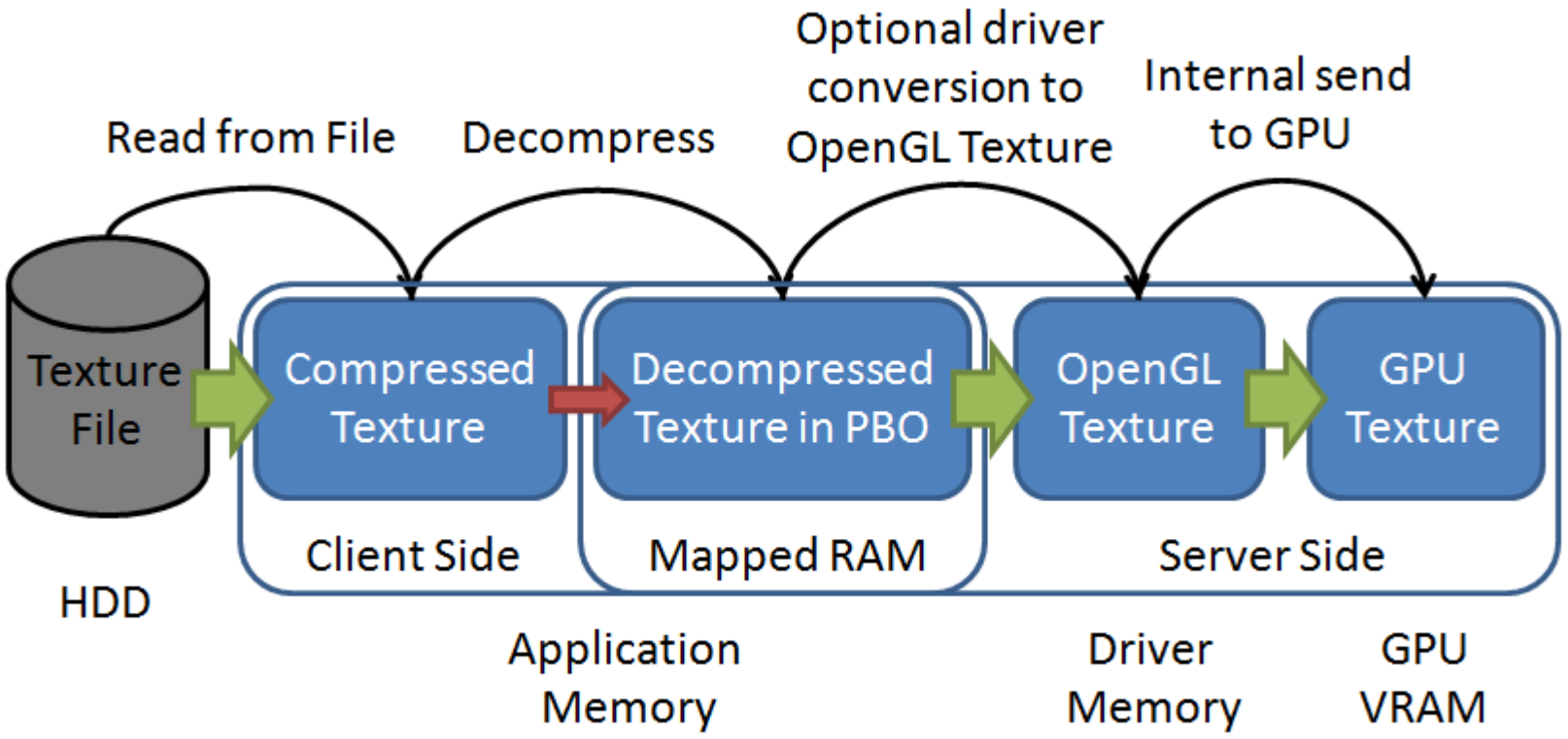
Idea of multithreaded rendering

Loading resources



Idea of multithreaded rendering

Loading resources



Idea of multithreaded rendering

Compiling Shaders

- Compiling shaders is second thing that can be distributed to worker Rendering Contexts.
- Shaders consumes small amount of memory in opposition to textures and geometry, but they consume huge amount of time to compile.
- We don’t want to stall main thread on shaders compilation.

Idea of multithreaded rendering

Synchronization

- After distributing all work to different threads there is time for synchronization of asynchronous work :P
- It isn't easy but OpenGL help us in this by giving mechanism of sync obcject.
- We need to qery driver to ask if loading is finished. It's not a problem taking into notice that loading and compiling are very rare operation in comparision to state changes.

Agenda

- Introduction
  - Multicore is now
  - Everything is about streaming
- Idea of multithreaded rendering
  - Changing state
  - Loading resources
  - Compiling shaders
  - Synchronization
- **Future research**
- **Summary**

Future research

- Create complete rendering system working in Task Pooled environment
- Implement resources streaming based on Virtual Texturing
- Currently working in small group on playable tech demo of this solution!

Summary

- Karol Gasiński
- Graphic Software Engineer
- www.mrkaktus.org

- Don’t hesistate to send me some questions at:
- kuktus@gmail.com

- Thanks for listening ☺