



第九届中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2017

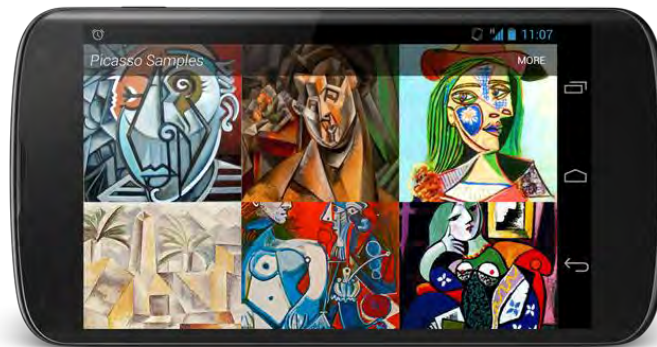
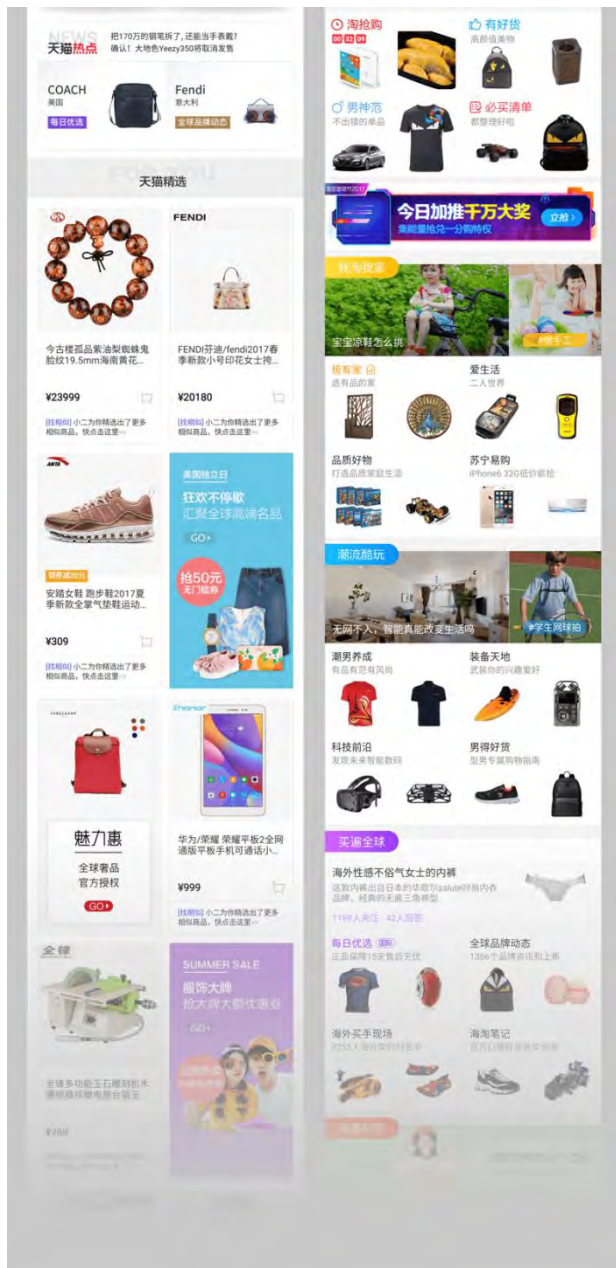
移动端图像加载优化与增强

基于图形硬件特性

阿里巴巴猫客技术部-默燧 (方超)
indianpapa@163.com

AGENDA

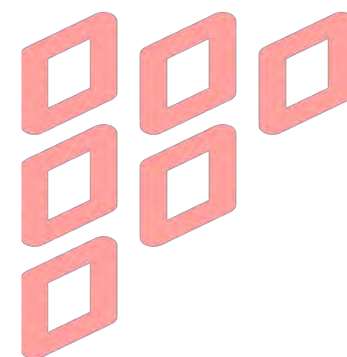
- 图片内容业务和行业现状
 - 图片内容业务
 - 现有解决方案和不足
- 技术背景和关键特性
- 框架设计与实现



本图由Picasso官方网站提供 | Source: <http://square.github.io/picasso>



glide



- **CDN解析度分级**
- 文件头精简
- 长链接

网络
效率

- **Ashmem**
- 位图复用

内存
利用

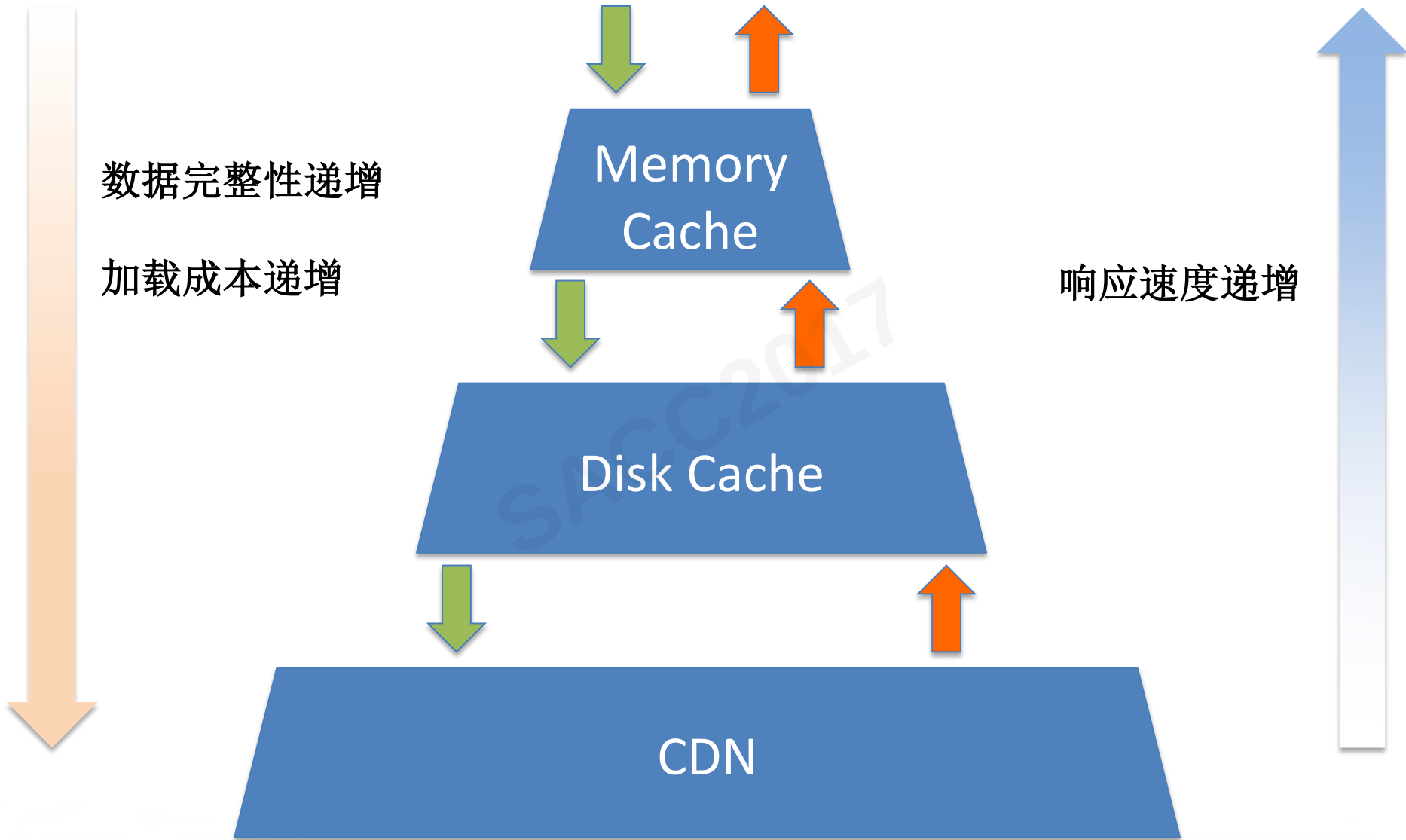
- 缓存分级
- 淘汰策略
- 缩略图预加载

响应
速度

- 异构并行处理
- 占位图

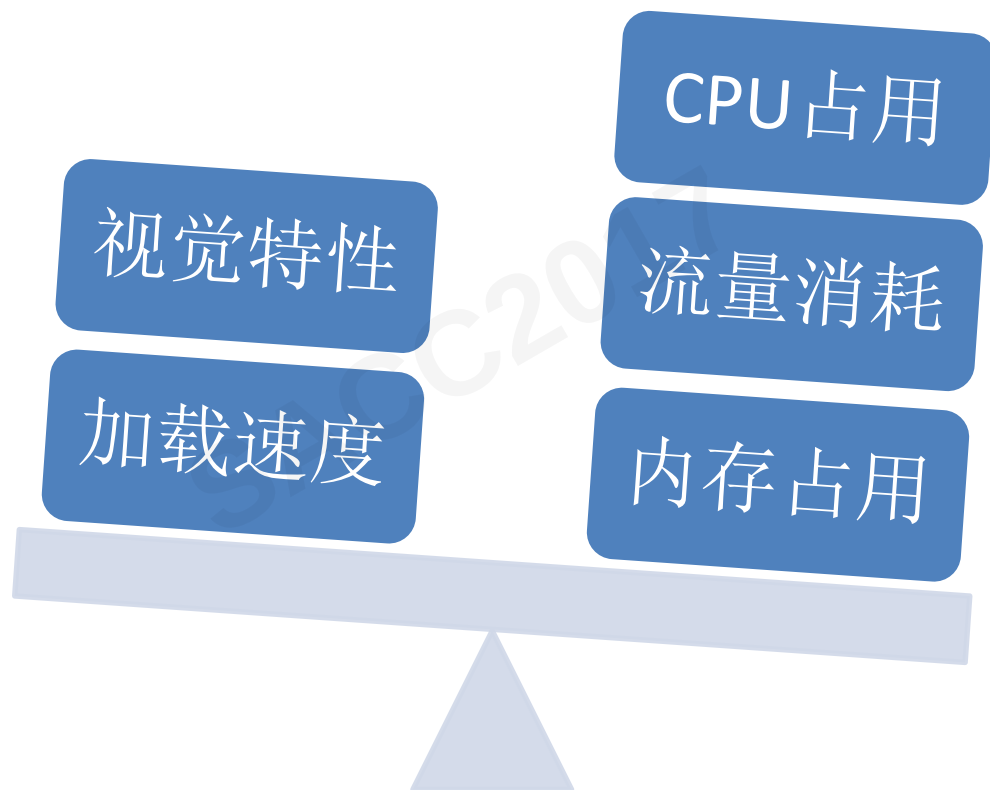
显示
特性

Request Return



图片性能

系统限制



AGENDA

- 图片内容业务和行业现状
- 技术背景和关键特性
 - 移动图形硬件的发展
 - 可编程渲染管线和图像渲染
 - 纹理缓存
 - 渲染线程和OpenGL上下文共享
 - 基于时序的缓存预测
- 框架设计与整合

Evolution of Mobile Graphics

2010: TrueForce



Hardware: Galaxy S2
GPU: Mali-400MP4
API support: OpenGL ES 2.0
Primitives per frame: 16k
Cycles per pixel: 3.7
Draw calls per frame: 50

2013: Trollheim



Hardware: Nexus 10
GPU: Mali-T604
API support: OpenGL ES 3.0,
OpenCL 1.1
Primitives per frame: 150k
Cycles per pixel: 16
Draw calls per frame: 60

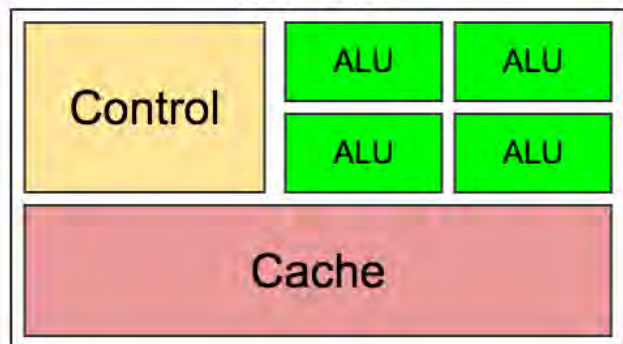
2016: Lofoten



Hardware: Galaxy S7
GPU: Mali-T880MP12
API support: Vulkan 1.0
Primitives per frame: 600k
Cycles per pixel: 40
Draw calls per frame: 500

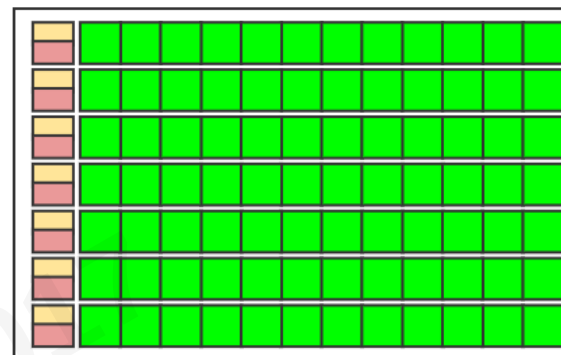
本图由ARM提供 | Source: <http://community.arm.com>

CPU



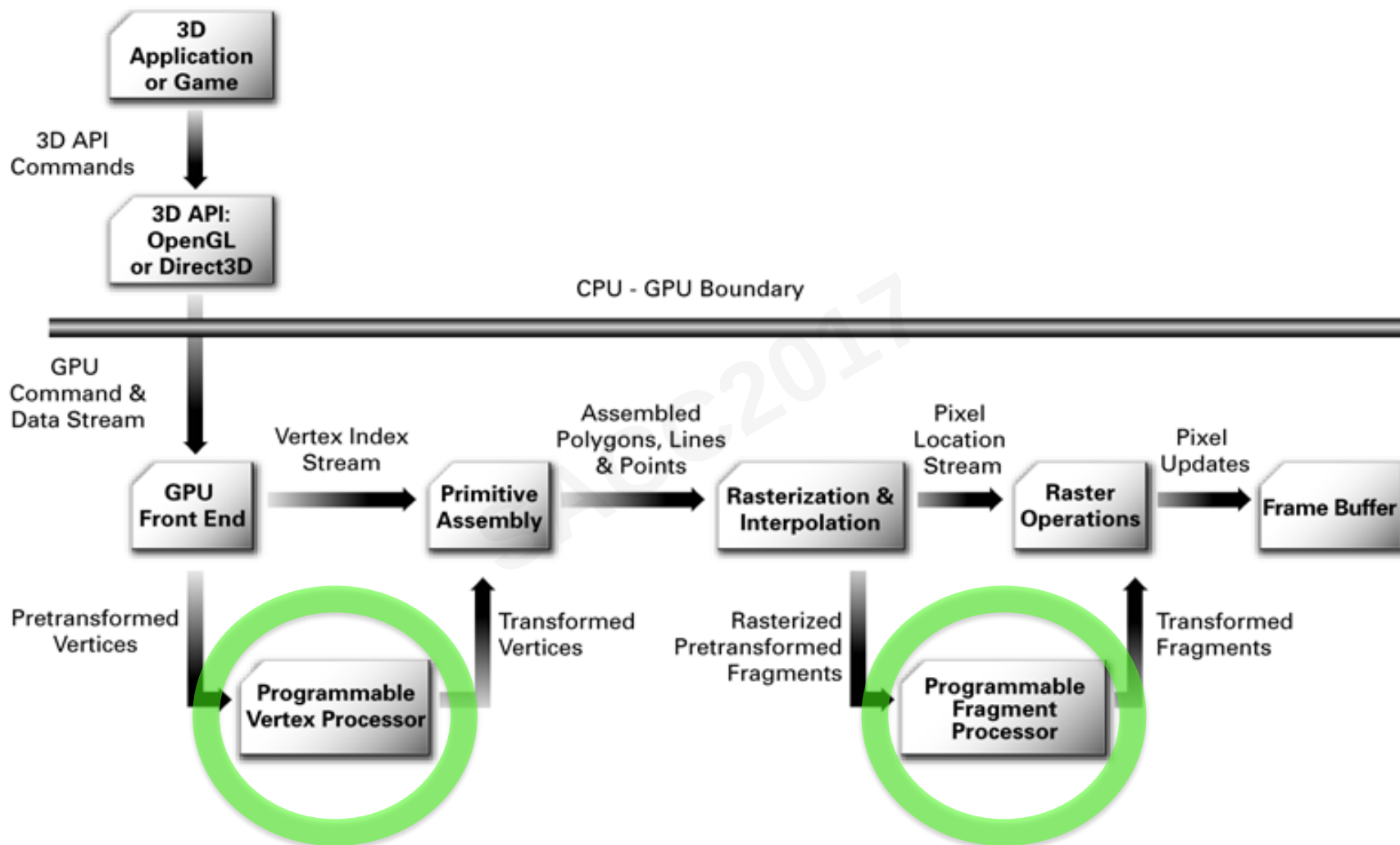
- 低运算单元密度
- 复杂控制逻辑
- 大缓冲存储器
- 串行操作优化
 - 高时钟频率
 - 少逻辑运算单元(ALUs)
- 短流水线（一般少于30个阶段）

GPU



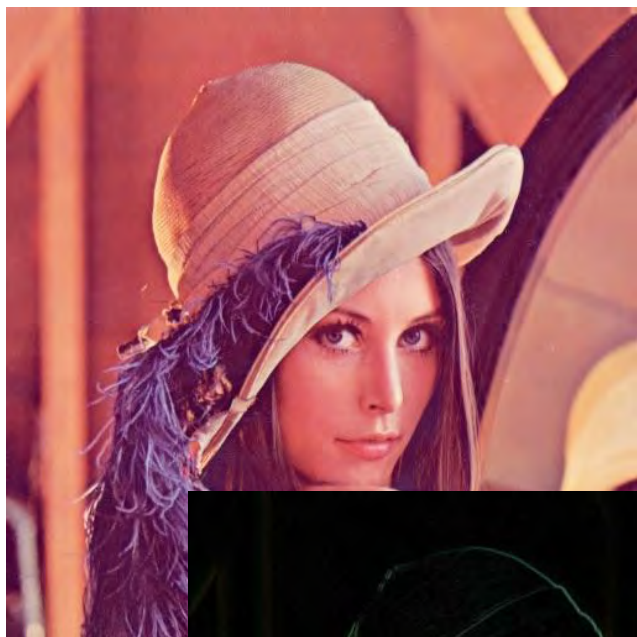
- 高运算单元密度
- 高计算内存访问
- 基于并行计算设计
 - 大量并行逻辑运算单元
 - 向量计算优化
- 深度流水线设计
- 高数据吞吐量

可编程图形渲染管线



本图由Nvidia提供 | Source: <http://developer.nvidia.com>

基于片段汇编的图像处理



Fragment Shader Program

```
precision mediump float;
uniform vec2 u_pixsize;
uniform sampler2D tex_origin;
varying vec2 v_texcoord;

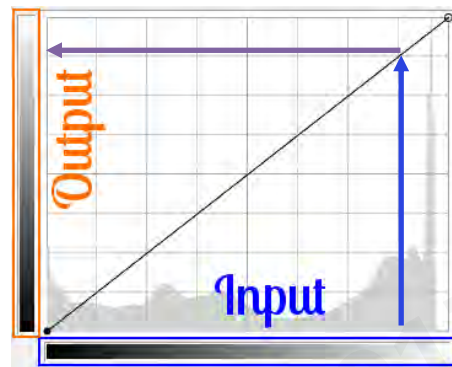
void main() {
    float dx = u_pixsize.x * u_param_0;
    float dy = u_pixsize.y * u_param_0;

    vec4 sam1 = texture2D(tex_origin, vec2(v_texcoord.x - dx, v_texcoord.y - dy));
    vec4 sam2 = texture2D(tex_origin, vec2(v_texcoord.x, v_texcoord.y - dy));
    vec4 sam3 = texture2D(tex_origin, vec2(v_texcoord.x + dx, v_texcoord.y - dy));
    vec4 sam4 = texture2D(tex_origin, vec2(v_texcoord.x - dx, v_texcoord.y));
    vec4 sam5 = texture2D(tex_origin, vec2(v_texcoord.x + dx, v_texcoord.y));
    vec4 sam6 = texture2D(tex_origin, vec2(v_texcoord.x - dx, v_texcoord.y + dy));
    vec4 sam7 = texture2D(tex_origin, vec2(v_texcoord.x, v_texcoord.y + dy));
    vec4 sam8 = texture2D(tex_origin, vec2(v_texcoord.x + dx, v_texcoord.y + dy));

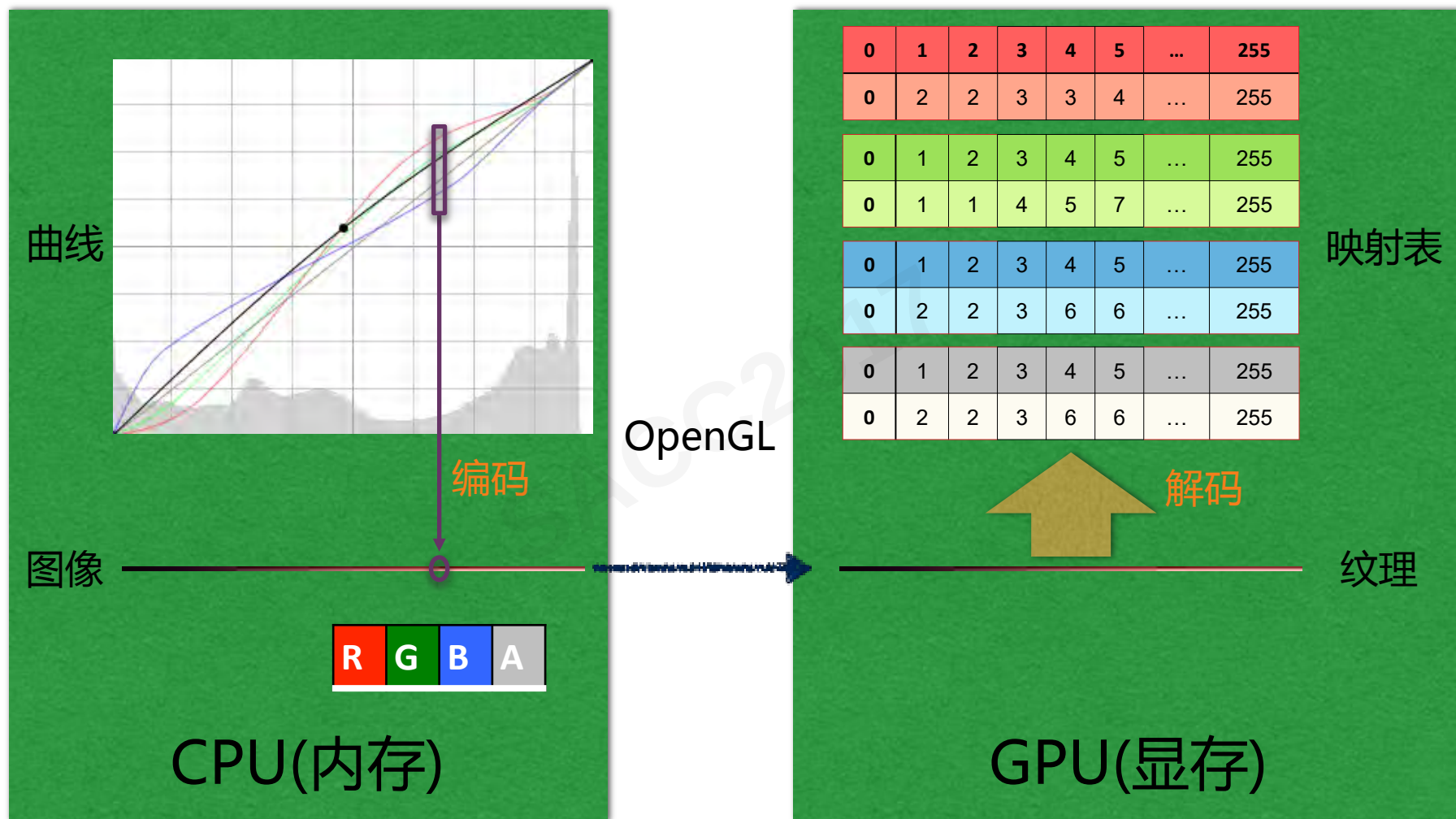
    vec4 gx = (sam3 - sam1) + (sam5 - sam4) * 2 + (sam8 - sam6);
    vec4 gy = (sam6 - sam1) + (sam7 - sam2) * 2 + (sam8 - sam3);

    gl_FragColor = dot(gx, gx) + dot(gy, gy);
}
```

曲线调节效果



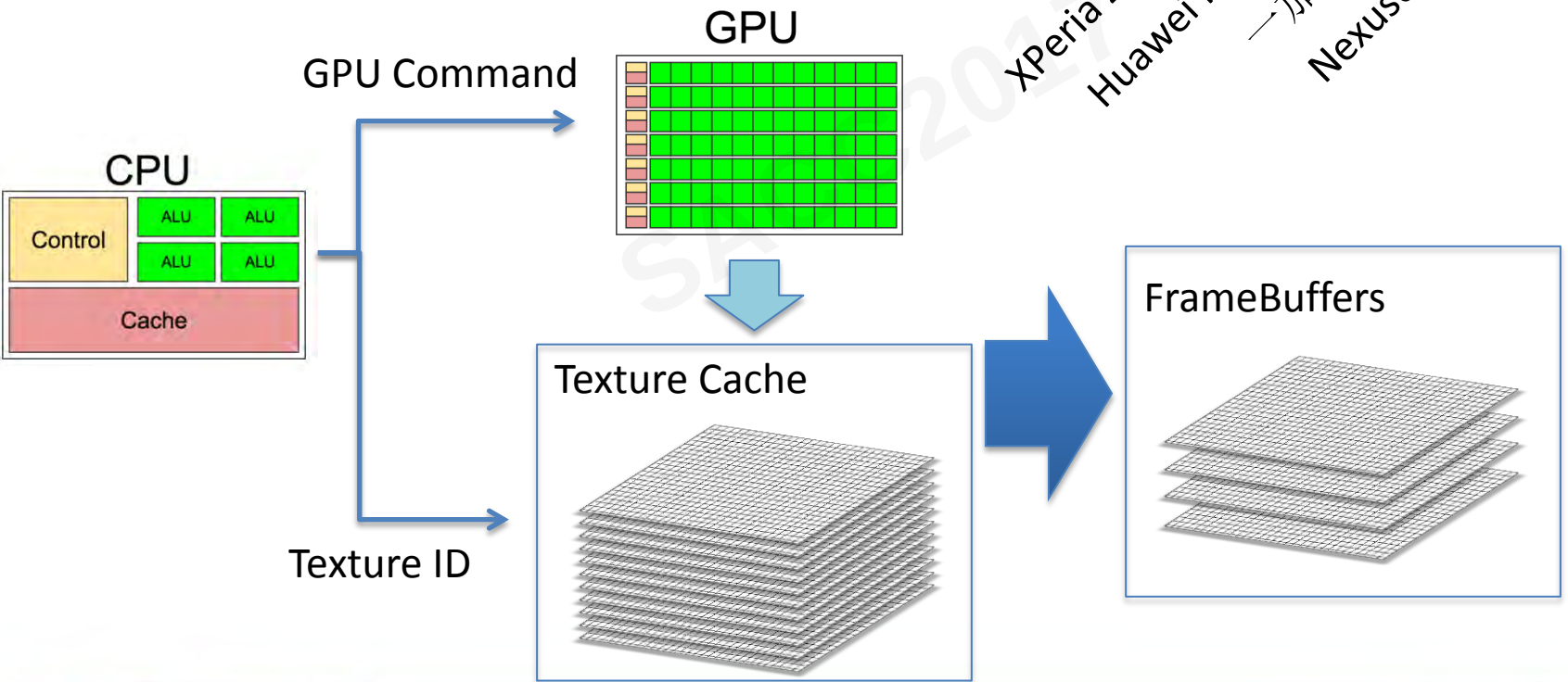
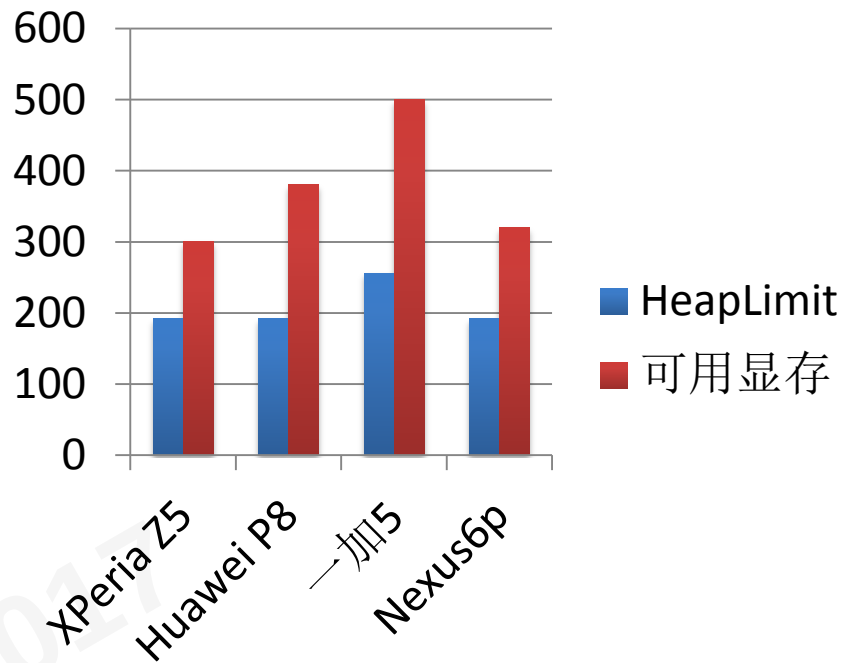
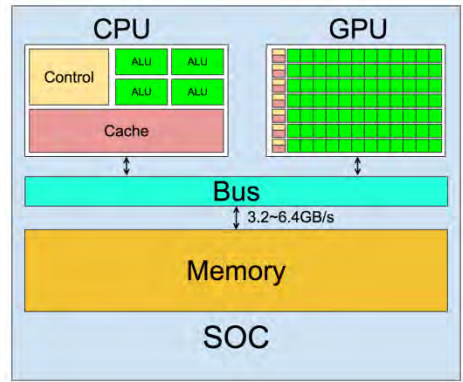
曲线调节效果



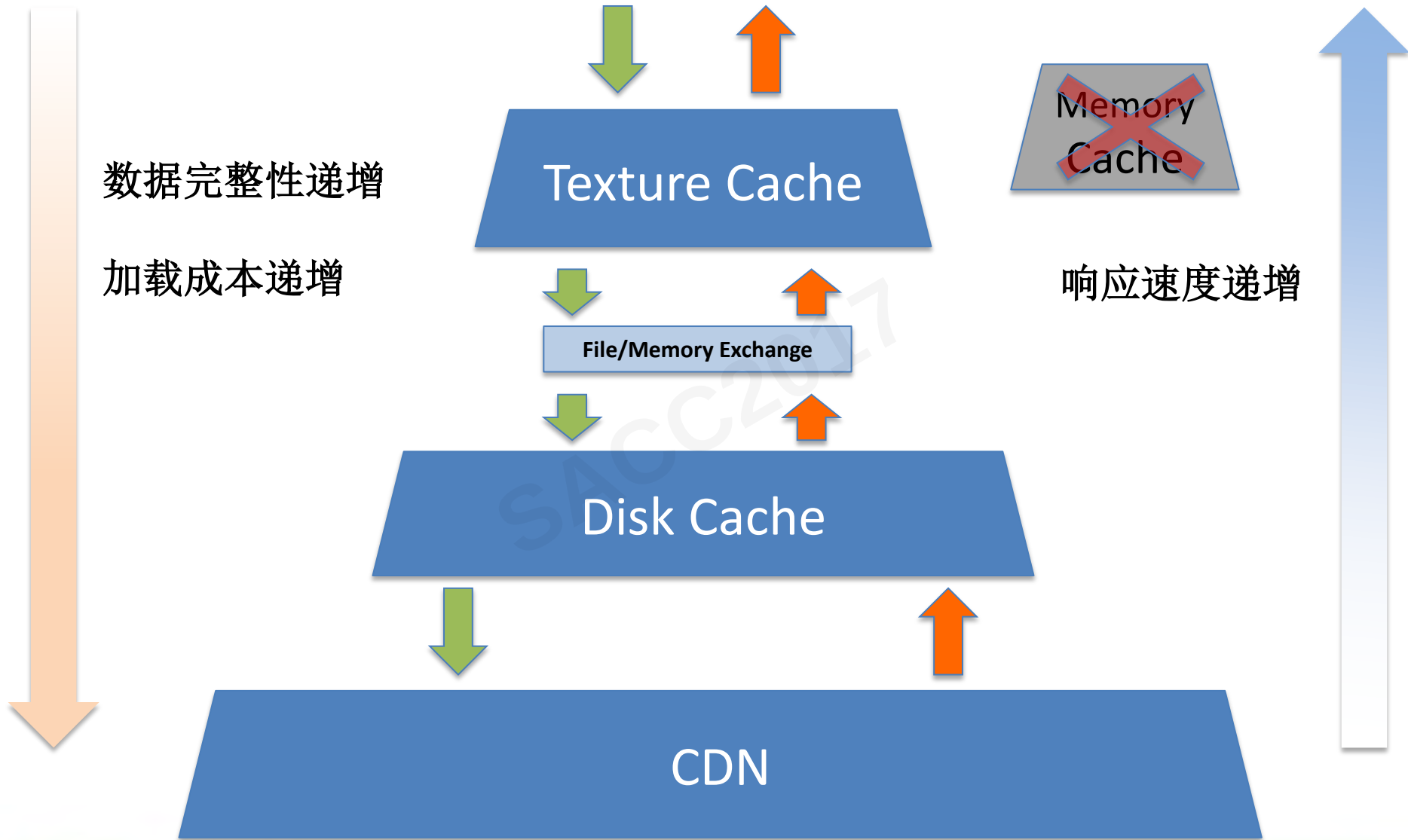
图片的动态显示效果



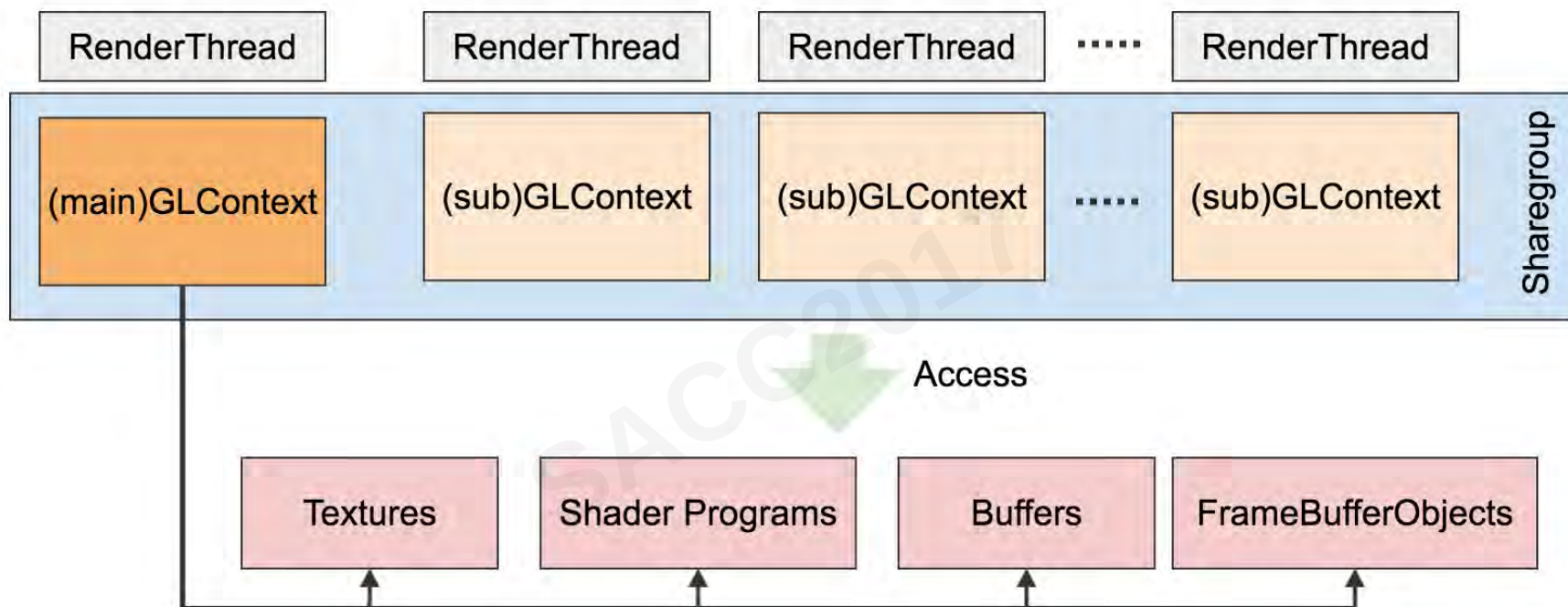
纹理缓存



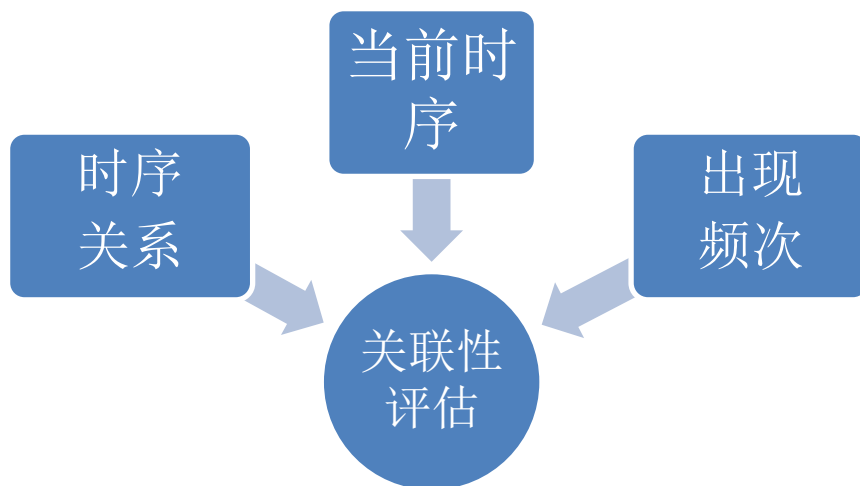
Request Return



渲染线程和共享上下文



基于时序的缓存策略



$$P(B|A) \propto P(B) * P(A|B)$$

$P(A)$:当前内容被使用概率, $P(B)$:待评估内容被使用概率

$P(B|A)$:当前内容被使用时待评估内容也被使用的概率



缓存预测

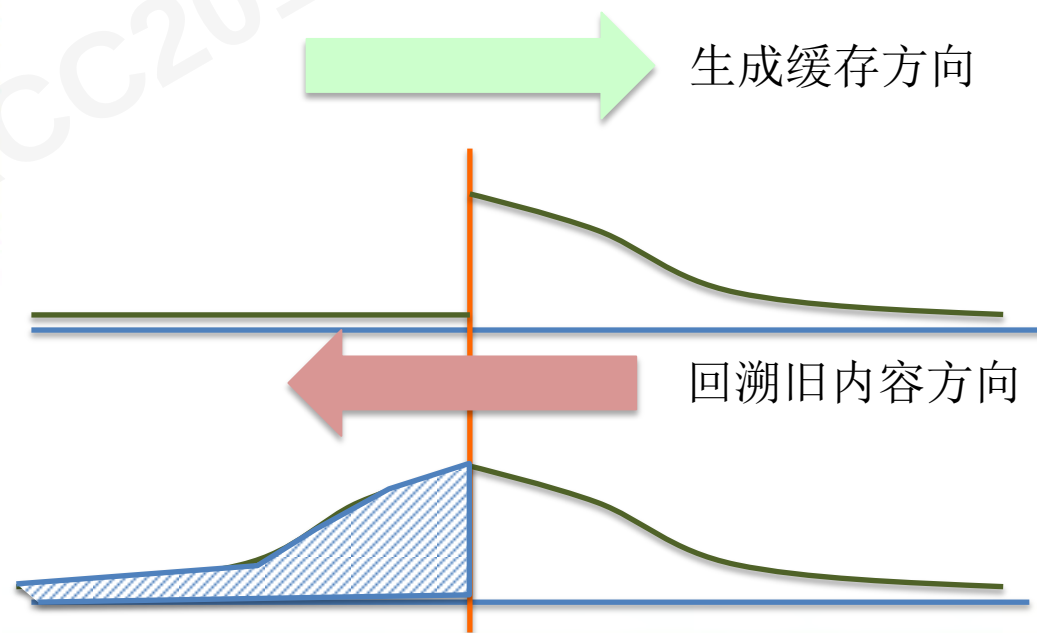
缓存淘汰

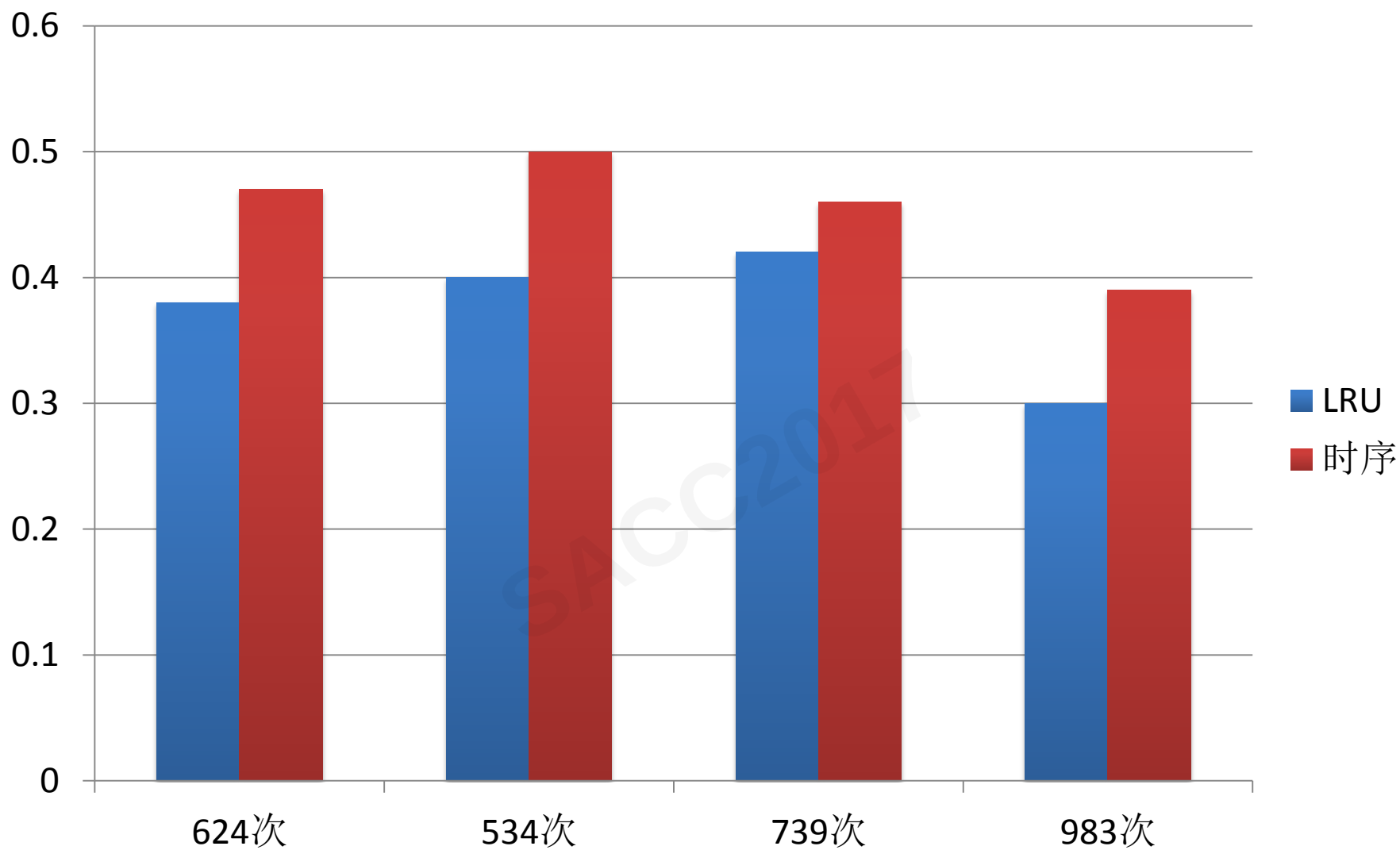


$$\text{LRU: } P(B|t) \propto P(B) * \frac{1}{(t - t_b)}$$

VS

$$\text{内容时序相关性: } P(B|A) \propto P(B) * P(A|B)$$

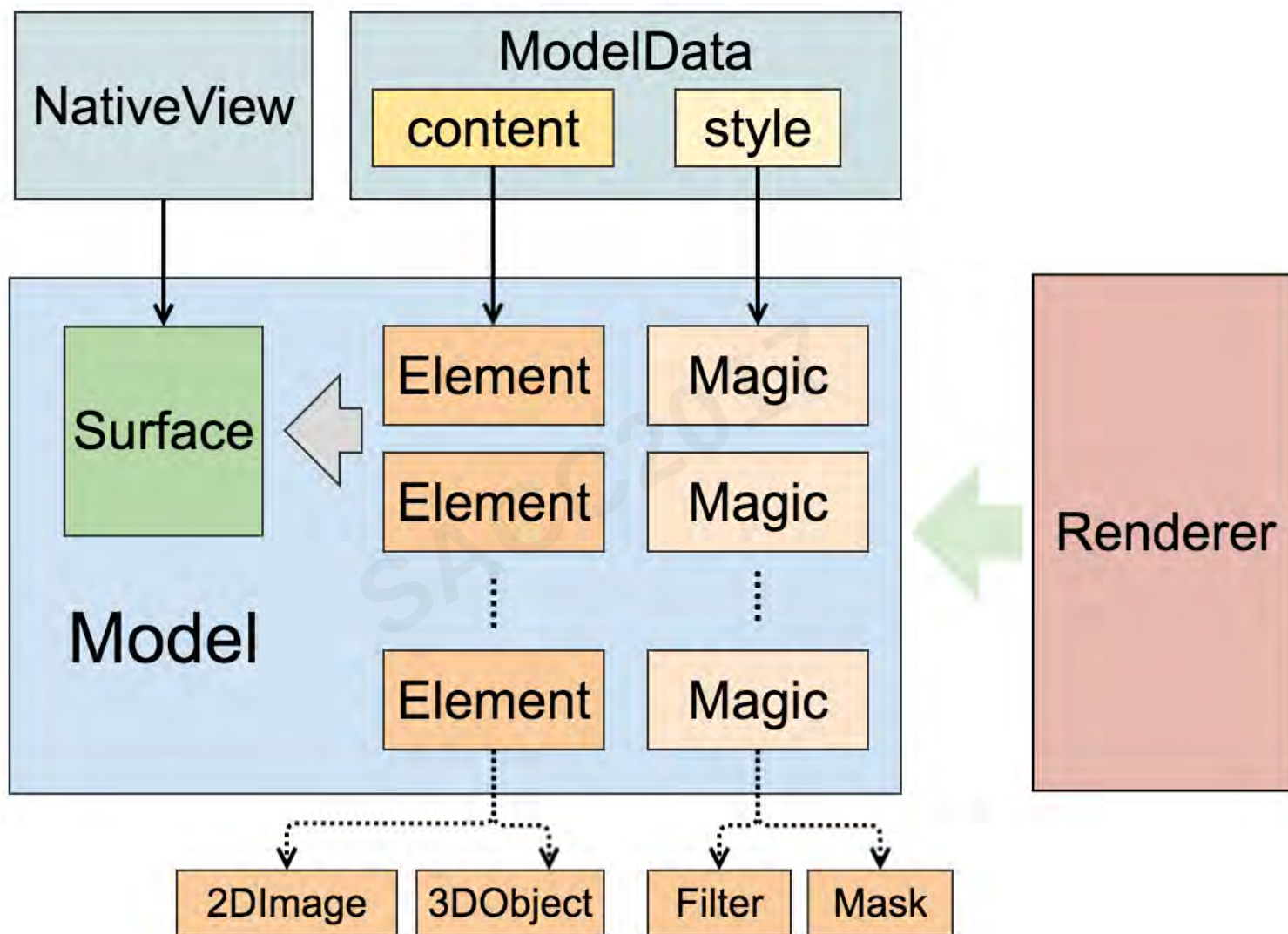




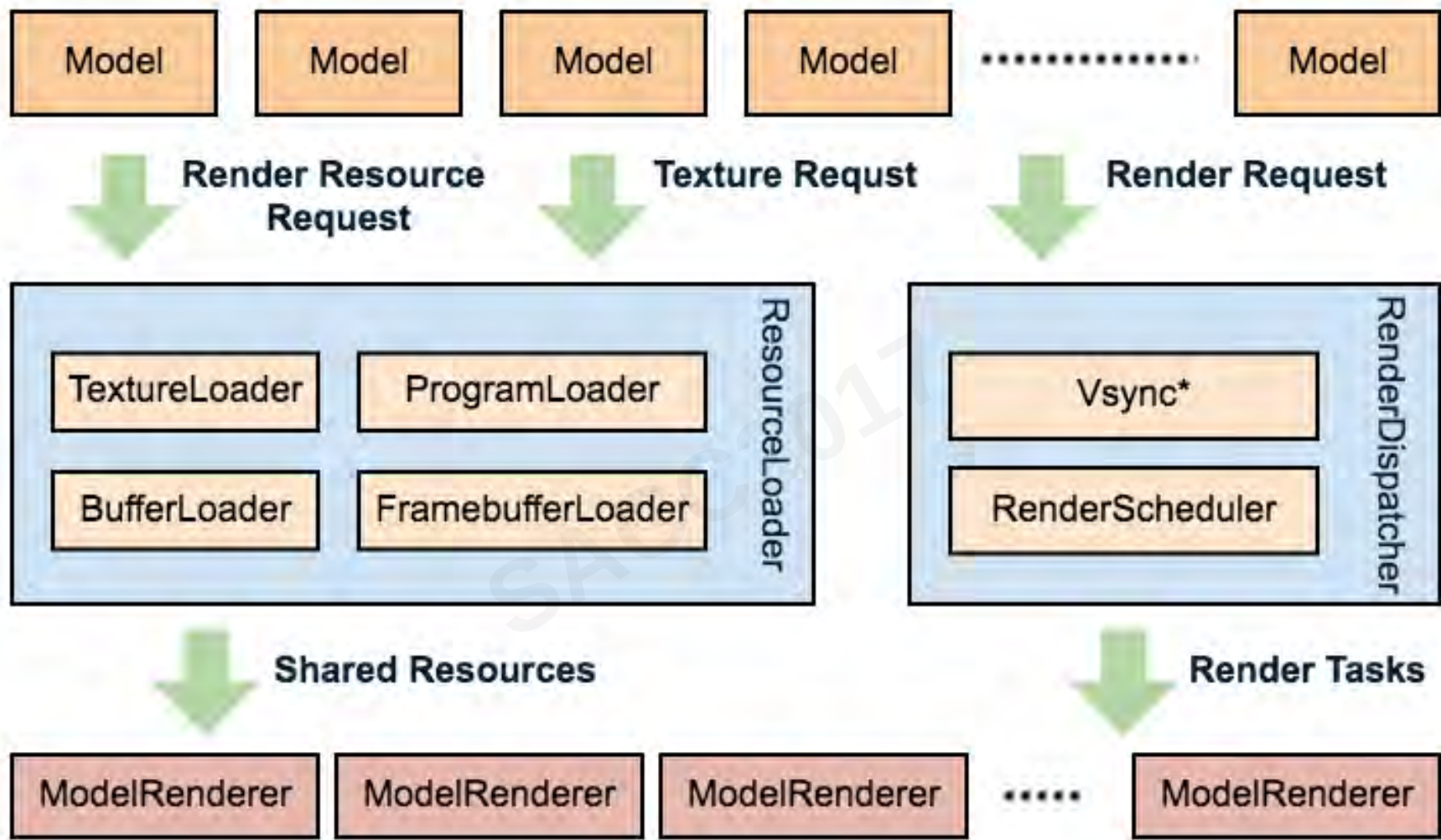
AGENDA

- 图片内容业务和行业现状
- 技术背景和关键特性
- 框架设计与整合
 - 模型渲染框架
 - 资源加载和渲染任务分发

组件模型渲染框架



资源加载和渲染任务分发



THANKS

