上海科技大学
ShanghaiTech University

Discusstion 12: SIMD and OpenMP

YiHeng Wu

1. SIMD

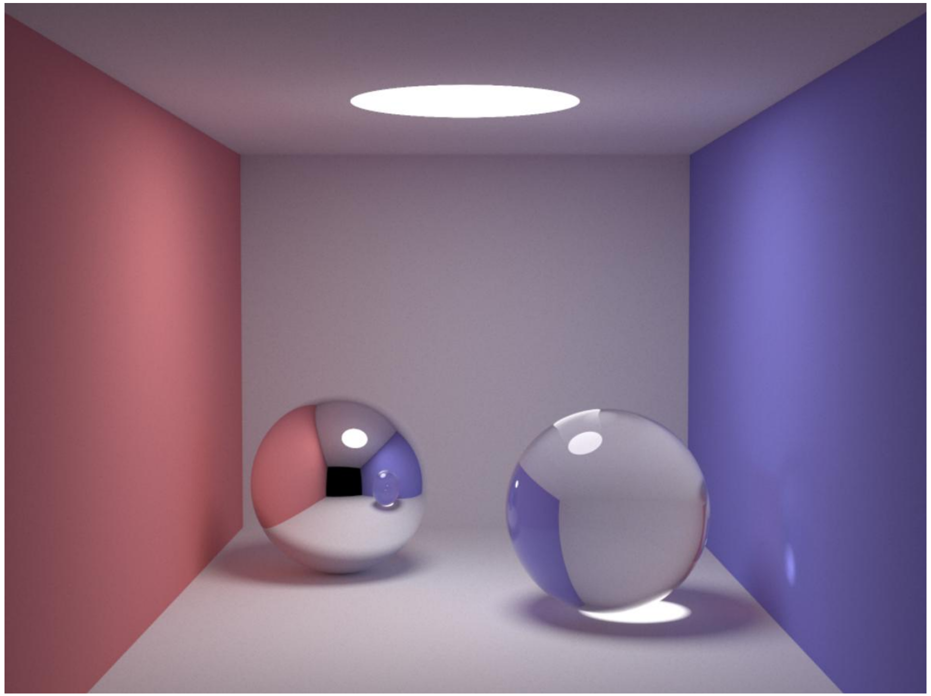
SIMD

- What is SIMD?
  - Single instruction multiple data
  - Vectorization
- How to code?
  - https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html

SIMD

- What is SIMD?
  - Single instruction multiple data
  - Vectorization
- How to code?
  - https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html
- **Let’s optimize HW5 with SIMD!**

HW5



SIMD
Method 1

```
typedef struct {
    float x;
    float y;
    float z;
} vec3;
```

SIMD
Method 1: Vectorized Vec3

```c
typedef struct {
    float x;
    float y;
    float z;
} vec3;
```

```c
#include <nmmintrin.h>

typedef struct {
    __m128 xyzw;
} vec3;
```

(simply ignore w)

上海科技大学 ShanghaiTech University

SIMD
Method 1: Vectorized Vec3

```
typedef struct {
        float x;
        float y;
        float z;
} vec3;
```

```
#include <nmmintrin.h>

typedef struct {
  union {
    __m128 m128;
    struct { float x, y, z; };
  };
} vec3
__attribute__((aligned(16)));
```

(simply ignore w)

https://github.com/embree/embree/blob/master/common/math/vec3fa.h

上海科技大学 ShanghaiTech University

SIMD
Method 1: Vectorized Vec3

```
vec3 vec3_init(float x, float y, float z) {
    vec3 res;
    res.x = x;
    res.y = y;
    res.z = z;
    return res;
}

vec3 vec3_add(const vec3 v1, const vec3 v2) {
    vec3 res;
    res.x = v1.x + v2.x;
    res.y = v1.y + v2.y;
    res.z = v1.z + v2.z;
    return res;
}
```

SIMD
Method 1: Vectorized Vec3

```
vec3 vec3_init(float x, float y, float z) {
    vec3 res = { _mm_set_ps(0, z, y, x) };
    return res;
}

vec3 vec3_add(const vec3 v1, const vec3 v2) {
    vec3 res = { _mm_add_ps(v1.m128, v2.m128) };
    return res;
}
```

SIMD
Method 1: Vectorized Vec3

```
int main()
{
    vec3 v1 = vec3_init(1.f, 2.f, 3.f);
    vec3 v2 = vec3_init(.1f, .2f, .3f);

    vec3 v3 = vec3_add(v1, v2);

    printf("(%f, %f, %f)\n", v3.x, v3.y, v3.z);
    return 0;
}
```

No Problem

SIMD
Method 1: Vectorized Vec3

```
int main()
{
    vec3 v1 = { 1.f, 2.f, 3.f };
    vec3 v2 = { .1f, .2f, .3f };

    vec3 v3 = vec3_add(v1, v2);

    printf("(%f, %f, %f)\n", v3.x, v3.y, v3.z);
    return 0;
}
```

No Problem

SIMD
Method 1: Vectorized Vec3

```
int main()
{
    vec3 v1, v2;
    v1.x = 1.f, v1.y = 2.f, v1.z = 3.f;
    v2.x = .1f, v2.y = .2f, v2.z = .3f;

    vec3 v3 = vec3_add(v1, v2);

    printf("(%f, %f, %f)\n", v3.x, v3.y, v3.z);
    return 0;
}
```

Not Recommended

上海科技大学 ShanghaiTech University

SIMD
Method 1: Vectorized Vec3

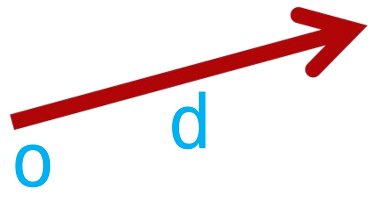
- Analyze
  - Of course good for some other applications.
  - But not good for HW5...
- Why?
  - For ray tracing, actually,
    this function is the most time consuming:

```
/* Whether or not ray hits objects in the scene */
bool intersect(const ray *r, float *t, int *id);
```

  - **Has little to do with our new vec3!**

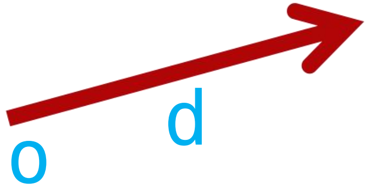
SIMD
Method 2

```
typedef struct {
    vec3 o;
    vec3 d;
} ray;
```

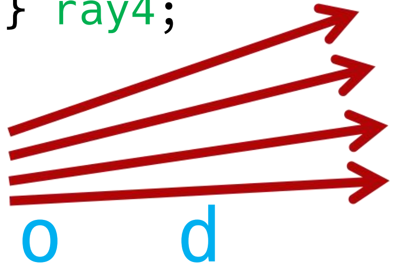


SIMD
Method 2: Packeted Rays

```
typedef struct {
    vec3 o;
    vec3 d;
} ray;
```

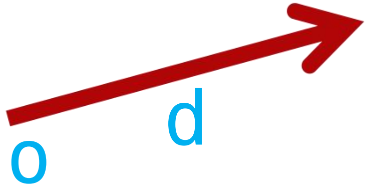


```
typedef struct {
    float ox[4];
    float oy[4];
    float oz[4];

    float dx[4];
    float dy[4];
    float dz[4];
} ray4;
```

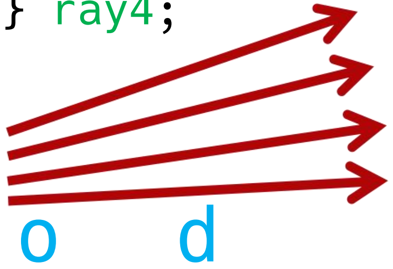


SIMD
Method 2: Packeted Rays

```
typedef struct {
    vec3 o;
    vec3 d;
} ray;
```



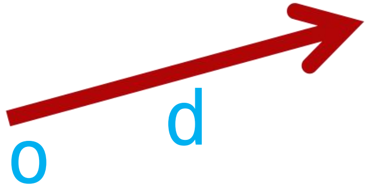
```
#include <nmmintrin.h>

typedef struct {
    __m128 ox;
    __m128 oy;
    __m128 oz;

    __m128 dx;
    __m128 dy;
    __m128 dz;
} ray4;
```



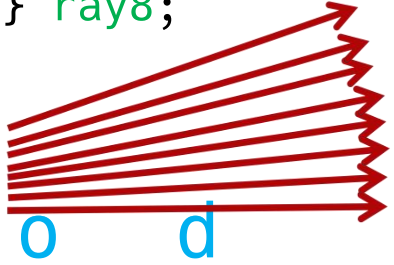
SIMD

Method 2: Packeted Rays

```
typedef struct {
    vec3 o;
    vec3 d;
} ray;
```



```
#include <immintrin.h>

typedef struct {
    __m256 ox;
    __m256 oy;
    __m256 oz;

    __m256 dx;
    __m256 dy;
    __m256 dz;
} ray8;
```



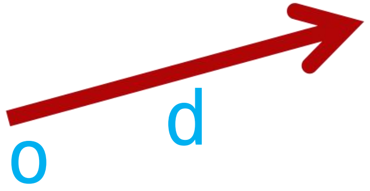
SIMD

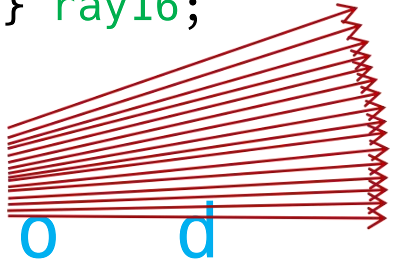
Method 2: Packeted Rays

```c
#include <immintrin.h>
```

```c
typedef struct {
    vec3 o;
    vec3 d;
} ray;
```

```c
typedef struct {
    __m512 ox;
    __m512 oy;
    __m512 oz;

    __m512 dx;
    __m512 dy;
    __m512 dz;
} ray16;
```





SIMD
Method 2: Packeted Rays

```
/* Whether or not 1 ray hits objects in the scene */
bool intersect(const ray *r, float *t, int *id);
```

SIMD
Method 2: Packeted Rays

```
/* Whether or not 1 ray hits objects in the scene */
bool intersect(const ray *r, float *t, int *id);
```

⇩

```
/* Whether or not 4 rays hit objects in the scene */
bool4 intersect4(const ray4 *r, float4 *t, int4 *id);
```

SIMD
Method 2: Packeted Rays

- Analyze
  - Much faster than method 1.
  - But difficult to make it scalable...

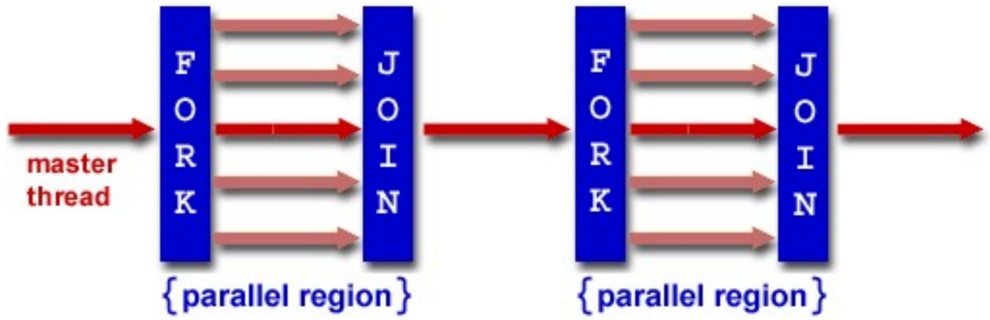
- Scalable?

  -
    ```
              Difficult to make ray4  faster than ray1.
         More difficult to make ray8  faster than ray4.
    More more difficult to make ray16 faster than ray8.
    ```
  - Why?

2. OpenMP

OpenMP

- What is OpenMP?
  - Fork + join



  - **#pragma omp parallel for**

OpenMP

- Pragmas
  - #pragma omp parallel for private(x)
  - #pragma omp parallel for reduction(+:sum)

- Functions
  - omp_set_num_threads(...);
  - ... = omp_get_num_threads();
  - ... = omp_get_thread_num();

OpenMP

- Pragmas
  - `#pragma omp parallel for private(x)`
  - `#pragma omp parallel for reduction(+:sum)`

- Functions
  - `omp_set_num_threads(...);`
  - `... = omp_get_num_threads();`
  - `... = omp_get_thread_num();`

- **Let’s optimize HW5 with OpenMP!**

OpenMP

```
for (y = 0; y < image_height; ++y) {
    ...
}
```

OpenMP

```
for (y = 0; y < image_height; ++y) {
    ...
}
```

$$\Downarrow$$

```
#pragma omp parallel for
for (y = 0; y < image_height; ++y) {
    ...
}
```

OpenMP

```
for (y = 0; y < image_height; ++y) {
    ...
}
```

⇩

```
#pragma omp parallel for schedule(dynamic, 2)
for (y = 0; y < image_height; ++y) {
    ...
}
```

上海科技大学
ShanghaiTech University

Thank you