# GPU Programming and Visualization

## Graduate Course
## Fall 2024
## Lecture 5

Fall 2024, Professor Hanno Rein

# WebGPU

## Timing

- The timestamp-query feature is only implemented in the beta version of Safari
- Chrome
- Goto about:flags search foe WebGPU development features

- Reason why this is not enabled by default: fingerprinting

# WebGPU

## Task 1:

- Use the trick from last lecture to convert the branched kick step to an un-branched one:

  select( *[false]*, *[true]*, *[condition]* );

- Measure speedup

# WebGPU

## Task 2:

- Expand Particle structure to include a place to store the energy (f32).
- What does this do to padding?

# WebGPU

## Task 3:

- Write a new compute shader which computes the energy for each particle and stores it in the particle structure.
- How much longer does it take to run the code?

# WebGPU

## Task 4:

- Change RNG

  ```
  let rng_state = 20;
  initialConditions[i] = Math.random()-0.5;
  rng_state = (1103515245 * rng_state + 12345) % 0x80000000;
  initialConditions[i] = rng_state / (0x80000000 - 1) - 0.5;
  ```

- Comment out Leap frog step so we can time energy function.
- Copy data to CPU

  (see code at bottom)

# WebGPU

## Task 5:

- Sum energies on GPU before transfer.
- First try:

```
@compute @workgroup_size(${workgroup_size}) fn sum_energy(
 @builtin(local_invocation_index) local_invocation_index: u32,
 @builtin(num_workgroups) num_workgroups: vec3<u32>,
 @builtin(workgroup_id) workgroup_id : vec3<u32>,
) {
 let pi = workgroup_id.x * ${workgroup_size} + local_invocation_index;
 if (pi>0){
  particles[0].energy += particles[pi].energy;
 }

}
```

- Duration: 0.00009475s. But non reproducible results

# WebGPU

## Task 6:

- Serial version:

```
@compute @workgroup_size(1) fn sum_energy() {
    for (var j = 1u; j<${Nparticles}; j++){
        particles[0].energy += particles[j].energy;
    }
}
```

- Duration: 0.026341875s. Reproducible results but much slower.

# WebGPU

## Task 7:

- Using 1 workgroup:

```
@compute @workgroup_size(64) fn sum_energy(
  @builtin(local_invocation_index) local_invocation_index: u32,
) {
  let chunk = u32(${Nparticles}/64);
  let jstart = local_invocation_index*chunk;
  for (var j = jstart+1; j<jstart+chunk; j++){
    particles[jstart].energy += particles[j].energy;
  }
  workgroupBarrier();
  if (local_invocation_index==0){
    for (var j = 1u; j<64; j++){
      particles[0].energy += particles[j*chunk].energy;
    }
  }
}
```

- Duration: 0.001193875s. Reproducible results, much faster.

# WebGPU

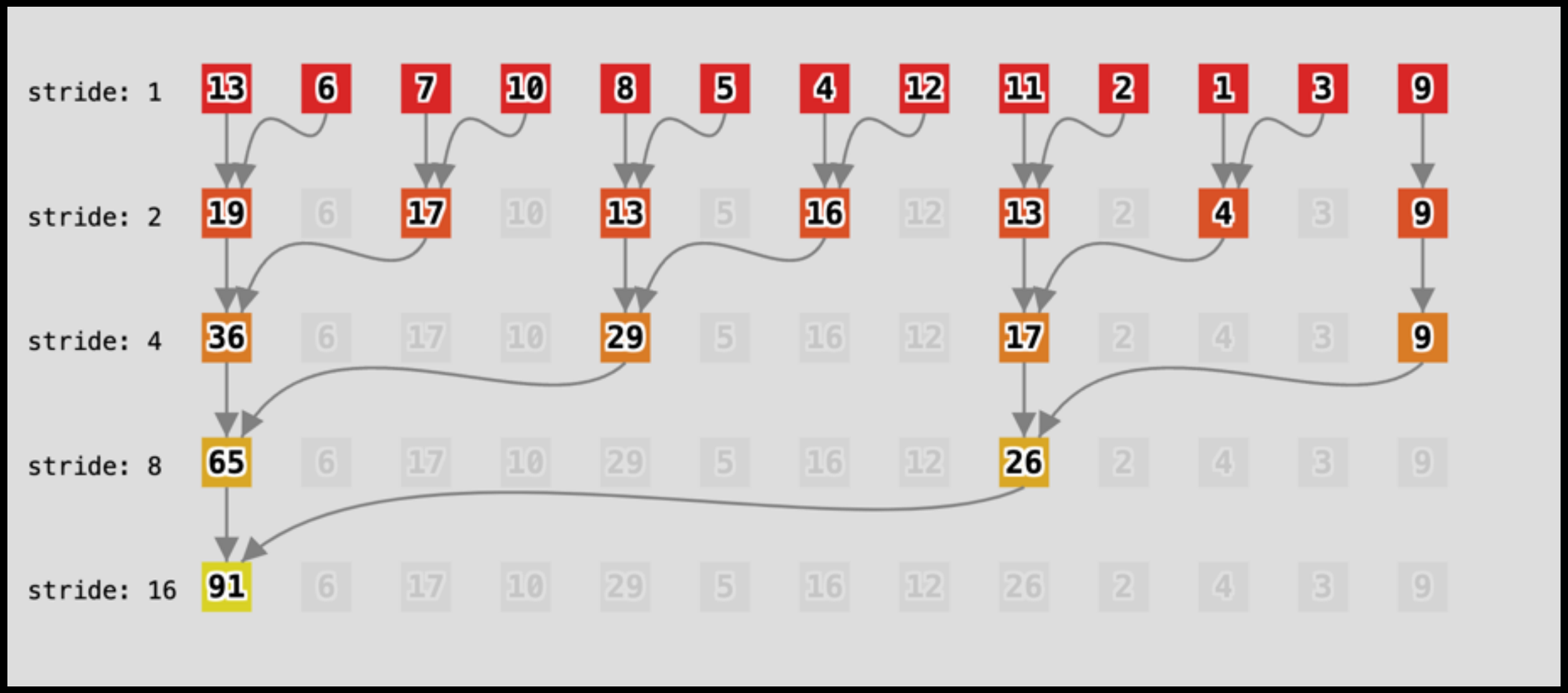
## Task 8:

- Using multiple workgroups

- Duration: 0.000192583s. Reproducible results, much faster.

# WebGPU

## Reduce



# WebGPU

## Task 9:

- Using reduce with a workgroup size of 8

- Duration: 0.000149875s. Reproducible results, a bit faster.