Parallel Computing with GPUs

Advanced OpenMP
Part 1 - Parallel Reduction

The University Of Sheffield.

Dr Paul Richmond
http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

- ❑Reduction
  - ❑Perform a parallel reduction using the reduction clause
  - ❑Recognise the limitations of the reduction functionality

- What do we need to look out for when considering applying OpenMP to this example?

```
void main(){
    int i;
    float vector[N];
    float sum;

    init_vector_values(vector);
    sum = 0;

    for (i = 0; i < N; i++){
        float v = some_func(vector[i]);
        sum += v;
    }
    printf("Sum of values is %f\n", sum);
}
```

Parallel Reduction

- A Reduction is the combination of local copies of a variable into a single copy
  - Consider a case where we want to sum the values of a function operating on a vector of values;

```
void main(){
    int i;
    float vector[N];
    float sum;

    init_vector_values(vector);
    sum = 0;

    for (i = 0; i < N; i++){
        float v = some_func(vector[i]);
        sum += v;
    }
    printf("Sum of values is %f\n", sum);
}
```

Candidate for parallel reduction…

Reduction clause

```
void main(){
    int i;
    float vector[N];
    float sum;

    init_vector_values(vector);
    sum = 0;

#pragma omp parallel for reduction(+: sum);
    for (i = 0; i < N; i++){
        float v = some_func(vector[i]);
        sum += v;
    }
    printf("Sum of values is %f\n", sum);
}
```

Without reduction we would need a critical section to update the shared variable!

OpenMP Reduction

- Reduction is supported with the reduction clause which requires a reduction variable
  - E.g. `#pragma omp parallel reduction(+: sum_variable) {…}`
  - Reduction variable is implicitly private to other threads
- OpenMP implements this **in parallel** by;
  - Creating a local (private) copy of the (shared) reduction variable
  - Combining (merging) local copies of the variable at the end of the structured block
  - Saving the reduced value to the shared variable in the master thread.
- Reduction operators are `+`, `-`, `*`, `&` , `|`, `&&` and `||`
  - `&`: bitwise and
  - `|`: bitwise or
  - `&&`: logical and
  - `||`: logical or

Summary

- ❑Reduction
  - ❑Perform a parallel reduction using the reduction clause
  - ❑Recognise the limitations of the reduction functionality

- ❑Next Lecture: Scheduling