OpenMP

Daniel Malik

Technical University of Munich
Garching

July 5, 2022

Structure

General Information

Implementation

Functionalities

Performance

Portability

Comparison

Summary

General Information

- Fortran, C and C++ supported
- Offloading added with OpenMP 4.0
- Exclusive to shared memory systems
- Highly popular

Implementation

- Enabled with compilerflag
- Add header / module
- High-level directives as pragmas / comments

```
gcc -fopenmp hello.c
```

```
#include <stdio.h>
#include <omp.h>

int main(void)
{
   #pragma omp parallel
    {
    printf("Hello!\n");
    }
  return 0;
}
```

Synchronization

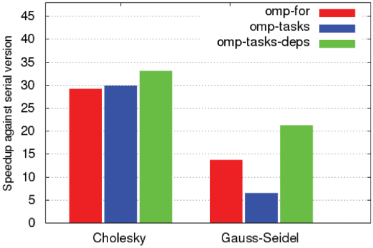
- Private and shared variables
- Mutex on shared variables with atomic statement
- Event synchronization with barrier construct

Types of Parallelism

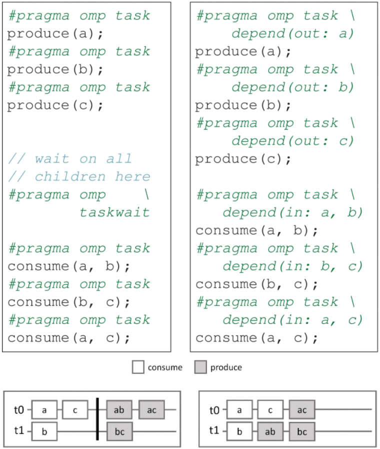
- Data parallelism
  - Loop-level parallelism
  - Loop collapse
- Task parallelism

Performance

- Dependences for better task scheduling
- Two Intel Xeon Platinum with 24 cores each



Bronis R. de Supinski et al. (2018)



```
#pragma omp task
produce(a);
#pragma omp task
produce(b);
#pragma omp task
produce(c);


// wait on all
// children here
#pragma omp     \
        taskwait

#pragma omp task
consume(a, b);
#pragma omp task
consume(b, c);
#pragma omp task
consume(a, c);
```

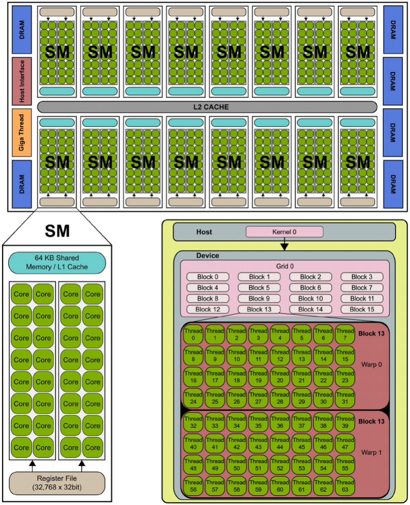
```
#pragma omp task \
     depend(out: a)
produce(a);
#pragma omp task \
     depend(out: b)
produce(b);
#pragma omp task \
     depend(out: c)
produce(c);

#pragma omp task \
    depend(in: a, b)
consume(a, b);
#pragma omp task \
    depend(in: b, c)
consume(b, c);
#pragma omp task \
    depend(in: a, c)
consume(a, c);
```

Bronis R. de Supinski et al. (2018)

GPU architecture and memory management

- Compilerflags for offloading and to specify target architecture
- Memory management with map clauses
- Bracket code to be offloaded with target construct

```
map (map-type : list)
```

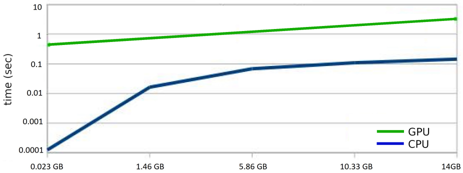


Moises Hernandez Fernandez et al. (2013)

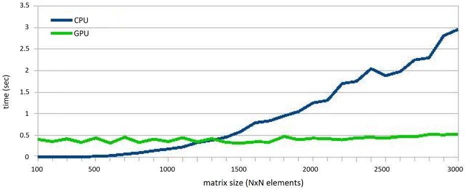
GPU offloading

- 4 NVIDIA Tesla V100 GPUs, 16GB memory each
- IBM POWER9 CPU with 24 Cores

Vector multiplication



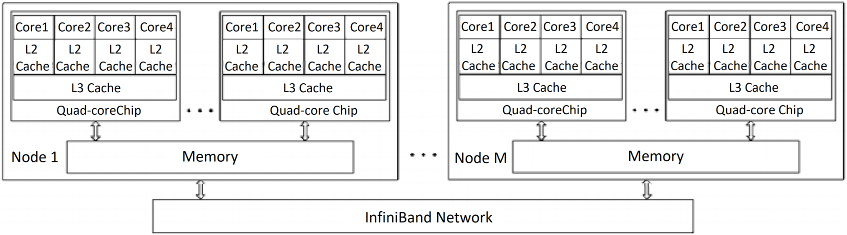
Matrix multiplication



Aditya Nitsure et al. (2019)

Supercomputers

- Hybrid model with MPI
- Cluster OpenMP
- Scalable up to high number of cores



Xiankun Miao et al. (2015)

OpenACC

- Share simplicity
- Similar performance
- More compiler flexibility than OpenMP

CUDA

- Low-level programming model
- Rewrite existing Code for porting
- Exclusive to NVIDIA hardware
- Better performance for more complex programs

Summary

- High-level language for specifying parallelism
- Easy to use
- Allows offloading code to GPUs
- Used on Supercomputers because of its scalability
- Good performance for simple programs