ETH Zürich

F. Vogel

HPCSE I HS16

12-929-266

General

```
-> setxkbmap ch -variant de_nodeadkeys
Makefile  make
 CXX = g++
 CFLAGS = -Wall -O3 -std=c++11

  test:  test.cpp
     $(CXX) $(CFLAGS) -o $@  $<

  clean:
     rm -f test
```

Threads

```
compile      g++ -std=c++11 -pthread
import       #include <thread>
launch func     std::thread t(foo, arg1);
launch λ        std::thread([&]() { //do something });
```

mutex:

```
import       #include <mutex>
declare      std::mutex mtx;
lock         mtx.lock();
               :
             mtx.unlock();
lock_guard     std::lock_guard<std::mutex> l(mtx);
               ↦ locked when initialized
               ↦ unlocked when destroyed
unique_lock    std::unique_lock<std::mutex> l(mtx1, mtx2);
               ↦ locks work with multiple locks (no deadlock)
               l.unlock();, l.lock(); for manual
```

Example

```
int nthreads = 4;
int nstep = N/nthreads;
std::vector<std::thread> threads(nthreads);
for (int t=0, t<nthreads; t++) {
    threads[t] = thread([&,t] {
        for (int i = t*nstep; i<(t+1)*nstep; i++){
            z[i] = x[i] + y[i];
    });
}
for (std::thread& t : threads) {
    t.join();
}
```

MPI

```
compile    mpicc, mpic++
import     #include <mpi.h>
run        mpiexec -np 4 ./a.out
info       MPI_Comm_rank(MPI_COMM_WORLD, &rank);
           MPI_Comm_size(MPI_COMM_WORLD, &size);
```

Example

```
int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);
    int num;
    MPI_Comm_rank (MPI_COMM_WORLD, &num);
    if (num == 0) {  //master
        MPI_status status;
        char txt[100];
        MPI_Recv(txt, 100, MPI_CHAR, 1, 42,
                            MPI_COMM_WORLD, &status);
        std::cout << txt << '\n';
    } else {  //worker
        std::string text = "Bla";
        MPI_Send(const_cast<char*>(text.cstr()),
                   text.size()+1, MPI_CHAR, 0, 42,
                   MPI_COMM_WORLD);
    }
    MPI_Finalize();
}
```

Open MP

```
compile    g++ -fopenmp -std=c++11
import     #include <omp.h>
info       omp_get_thread_num()   -> id
           omp_get_num_threads()  -> #threads
```

ETH Zürich

HPCSE I HS16

F. Vogel

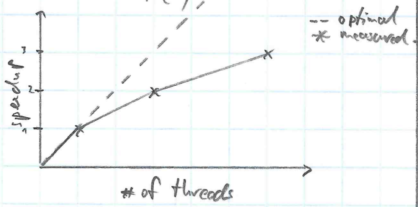
12-929-246

Parallel Scaling

Amdahls Law: $S(n) = \frac{1}{(1-p) + \frac{p}{n}}$

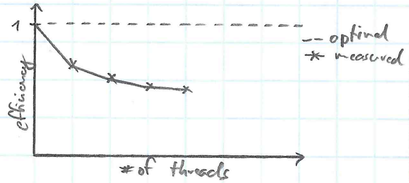
Gustafsons Law: $S(n) = 1 - p + n \cdot p$ (weak scaling)

Strong scaling: $S(n) = \frac{T(1)}{T(n)}$, $E(n) = \frac{S(n)}{n}$



Weak scaling: $E(n) = \frac{T(1)}{T(n)}$ fixed problem size per thread

Note: Problem size for N^2 solver is N^2 not N



Roofline Model

Operational Itensity [Flop/Byte] r

→ $r = \frac{\text{\# of Flops}}{\text{\# of Mem accesses} \cdot \text{\# of Bytes}}$

→ Example: $RHS_{i,j} = C_1 \cdot (q^n_{i+1,j} + q^n_{i-1,j} + q^n_{i,j+1} + q^n_{i,j-1} - 4q^n_{i,j})$

→ 6 Flops (4 ADD + 2 MUL)

→ Mem Access:

↦ No cache: 5 read + 1 write = 6

↦ Infinite cache: 1 read + 1 write = 6

→ Op. Int.

↦ No cache: $\frac{6\text{ Flops}}{6 \cdot 4\text{B}} = 0.25$ Flop/B

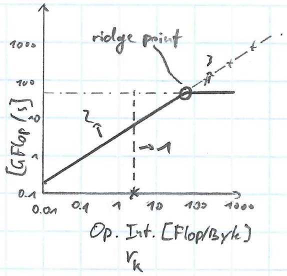
↦ Infinite cache: $\frac{6\text{ Flops}}{2 \cdot 4\text{B}} = 0.75$ Flop/B

Nominal performance

→ f_{peak} = processor clock/sec · vector size · instructions/clock · # cores

→ b_{peak} = memory clock/sec · channel size · #channels / 8[bits/Byte]

Plot



1: Locality

2: Communication

3: Computation

limit by comp. -·-·- $f_{peak} = f(r_k)$

limit by DRAM -x-x- $r_k \cdot b_{peak} = f(r_k)$

Bugs

→ Threads : Pass thread id by value, not by reference

→ SIMD : Race condition by interleaving of loads

→ SIMD : Undefined behavior by loading not allocated memory

→ OMP : implicit barrier @ end of for, all threads have to call it. Fix: schedule(dynamic) nowait

→ MPI : Don't update asynchronous send buffer

→ Threads : Not passing all needed variables

→ Threads : Race condition by not locking result double

→ Threads : Result reduction before join is called on each thread

→ Threads : Master thread: not releasing lock before calling join can result in deadlock

→ MPI : Ordering needed to prevent deadlock (replacing with asynchronous works too)