Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Assignment Feedback #1

Parallel Programming Concepts
Winter Term 2013 / 2014

Dr. Peter Tröger, M.Sc. Frank Feinbube

Assignments

General Concepts (Foster)

- Prefix Scan with Threads
- Philosophers with Threads

Shared Memory Parallelism

- HeatMap with Threads
- Parallel Grep with Java Monitors

- Decrypt with OpenMP (25.11. - 8.12.)
- HeatMap with OpenMP
- Matrixmultiplication with OpenMP

Accelerators

Shared Nothing Parallelism

Assignment Characteristics

Assignments have to be solved alone or as a team of two persons. The oral exam admittance is achieved if 50% of the assignments are solved correctly.

50% must be solved correctly in order to pass the assignment. Documentation should be done inside the source code.

The student achieving the lowest average runtime will be announced in the lecture.

Assignment 1: Problems & Solutions

- Learning Goals: Foster and Concurrency @Threads
- Prefix Scan
  - Parallel reduction on a large set of values …
  - … is one of the fundamental computational challenges
- Dining Philosophers
  - Basic concurrency problem
- How would you ensure good performance assessments?

Example: ./parsum 30 1 10000000000

Compilation test

rm -f parsum

rm -f output.txt

cc -pthread parsum.c -o parsum

Validation test

Test description: assigment example

Running: ./parsum 30 1 1000000000

Max thread count: 31 [Ok]

Runtime: 5.02

Example: ./parsum 30 1 10000000000

Full test

Testing was terminated since it took too long (350 seconds).
Output so far:

Test description: assigment example
Running: ./parsum 30 1 1000000000
Max thread count: 31 [Ok]
Runtime: 3.81

Test description: overflow
Running: ./parsum 30 1 10000000000
Max thread count: 31 [Ok]
Runtime: 48.37
[ERROR]----------
Result file does not match regular expression:
^500000000050000000000$
output.txt: -5340232216128654848

Test description: thread count > work item count
Running: ./parsum 100 1 10
thread count: 1 [Too Few!]
Runtime: 0.17
[ERROR]----------
Result file does not match regular expression: ^55$
output.txt: -5340232216128654848

Test description: work item count % thread count > 0
Running: ./parsum 30 1 911
Max thread count: 0 [Too Few!]
Runtime: 0.13

Test description: starting index > 1
Running: ./parsum 100 912 1000000000
Max thread count: 101 [Ok]
Runtime: 0.89

Test description: race
Running: ./parsum 30 1 100000000000

64-bit overflow

Sum of 1 .. 10.000.000.000 is 50.000.000.005.000.000.000 > 2^{64}

→128—bit numbers must be used

```
typedef struct {
    unsigned long hi;
    unsigned long lo;
} int_128;

inline int_128 add_int_128(const int_128 a, const unsigned long b)
{
    unsigned long oldLo = a.lo;
    a.lo += b;
    if (oldLo > a.lo)
        a.hi++;
    return a;
}
```

Third party libraries

Could use GMP's `mpz_t` for 128-bit arithmetics.

- Slowdown due to generic data structures
- Harder to optimize
- Uglier code
- Use of third-party libraries is discouraged unless explicitely noted in the assignment description
- Implementing 128-bit add() is not that hard

“Illegal” Optimizations


```
for (; i + SIZE_BULK_ADD < end; i += SIZE_BULK_ADD)
{
    result += SIZE_BULK_ADD * i;
    result += 1;
    result += 2;
    result += 3;
    result += 4;
    result += 5;
    result += 6;
    result += 7;
    result += 8;
    result += 9;
    result += 10;
    result += 11;
    result += 12;
    result += 13;
    result += 14;
    result += 15;
}
```

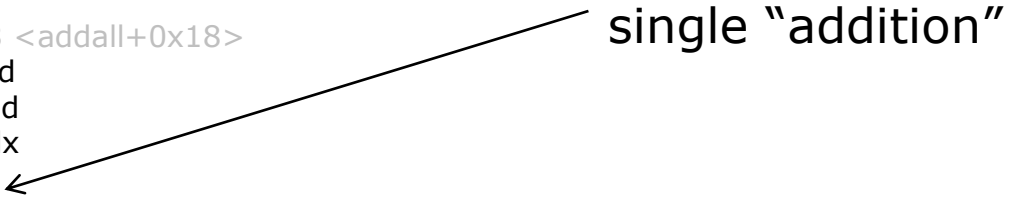
... what will happen?

“Illegal” Optimizations


```
for (; i + SIZE_BULK_ADD < end; i += SIZE_BULK_ADD)
{
    result += SIZE_BULK_ADD * i;
    result += 120;
}
```

Compiler will fold constants, no physical add operations.

```
cmp    $0x10,%ecx
jle    1004010fc <addall+0x2c>
mov    $0x10,%r8d
xor    %eax,%eax
xor    %r9d,%r9d
jmp    1004010e8 <addall+0x18>
mov    %r8d,%r9d
mov    %edx,%r8d
mov    %r9d,%edx
shl    $0x4,%edx
lea    0x78(%rax,%rdx,1),%eax
lea    0x10(%r8),%edx
cmp    %ecx,%edx
jl     1004010e2 <addall+0x12>
repz retq
xor    %eax,%eax
retq
```



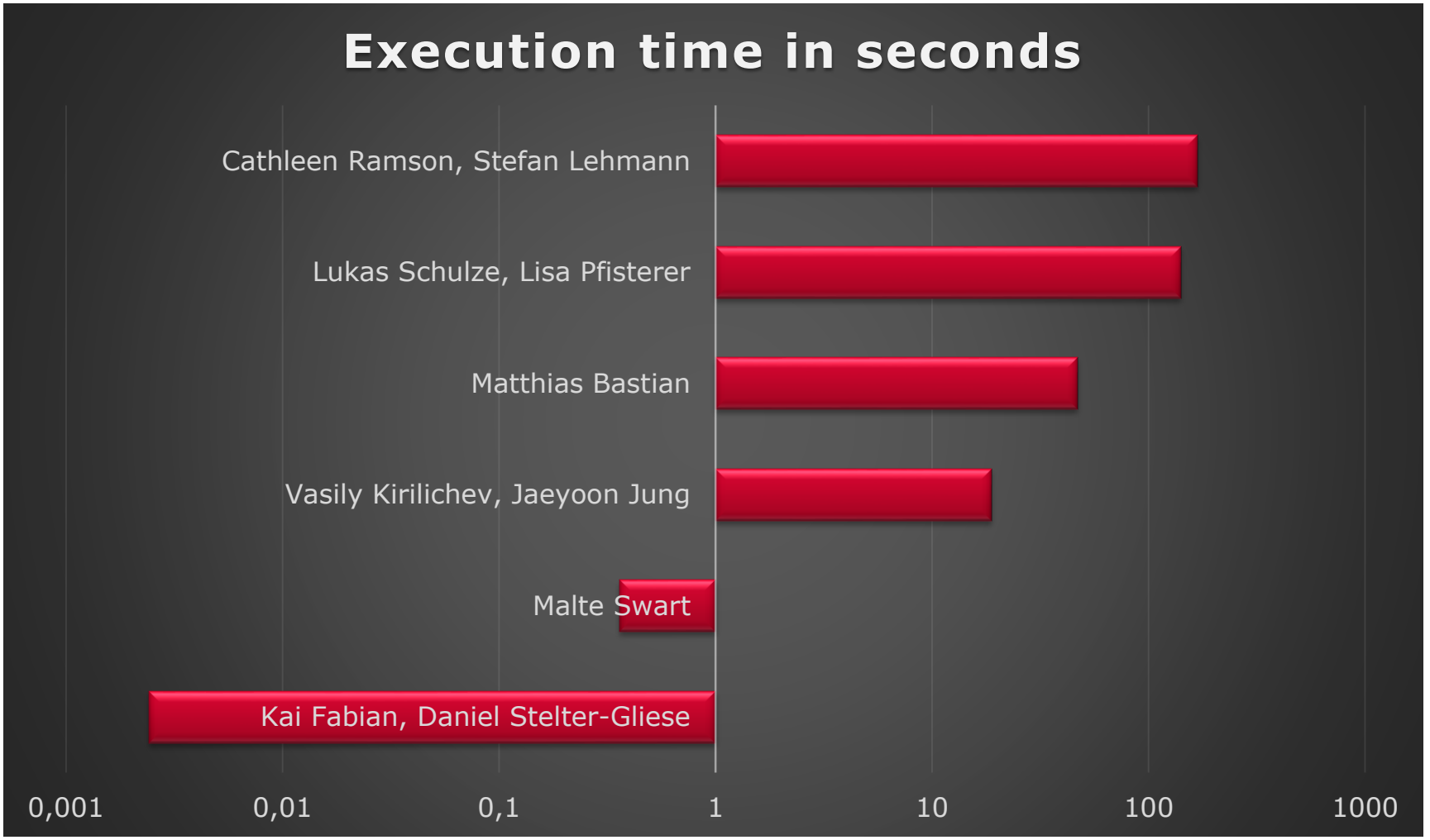
Making code fast

1. Use Optimization flags! `-O3` or `-Ofast`
    1. Can increase performance tenfold
2. Avoid debug- and performance-output
3. Manual Loop Unrolling

```
for (; start + 64 < myBlock.end; start += 64) {
    uint64_t sum = 0;
    sum += start + 0;
    sum += start + 1;
    sum += start + 2;
    // ...
    sum += start + 61;
    sum += start + 62;
    sum += start + 63;
    myBlock.sum += sum;
}

for (; start < myBlock.end; ++start) {
    myBlock.sum += start;
}
```

And the WINNER is…



And the WINNER is…

0,002429	Kai Fabian, Daniel Stelter-Gliese	correct results. super fast. > 600 LOC. highly optimized + assembler code. use of multiplications/shortcuts.
0,361774	Malte Swart	correct results. super fast. highly optimized. use of multiplications/shortcuts.

18,80469	Vasily Kirilichev, Jaeyoon Jung	way better with better compiler flags.
47,02925	Matthias Bastian	way better with better compiler flags.
140,3484	Lukas Schulze, Lisa Pfisterer	pretty slow due to gmp
168,2067	Cathleen Ramson, Stefan Lehmann	way better with better compiler flags.

Example: ./dinner 3 10
Good code or bad code?


```
for (i = 0; i < seats; i++)
        if (pthread_create(&threads[i], NULL, philosopher, (void*) i))
            exit(-1);
    sleep(time);

    for (i = 0; i < seats; i++)
        pthread_cancel(threads[i]);

if ((fp = fopen("output.txt", "w")))
        for (i = 0; i < seats; i++)
            fprintf(fp, i ? ";%lu" : "%lu", feedings[i]);
```

Example: ./dinner 3 10
Good code or bad code?


```
pthread_mutex_t* fork_mutex;
pthread_mutex_t eat_mutex;
[…]
void *diner(void* i) {
        int eating = 0;
        […]
        while (run) {
                pthread_mutex_lock(&eat_mutex);
                pthread_mutex_lock(&fork_mutex[v]);
                pthread_mutex_lock(&fork_mutex[(v + 1) % n_waiters]);
                eating++;
                pthread_mutex_unlock(&eat_mutex);
                […]
        }
        […]
}
```

Example: ./dinner 3 10
Good code or bad code?


```
while(true) {
        if (pthread_mutex_trylock(rightFork) != EBUSY &&
            pthread_mutex_trylock(leftFork) != EBUSY) {
                ++((struct ThreadParameterStructure*)input)->consumeCount;
        }
        pthread_mutex_unlock(rightFork);
        pthread_mutex_unlock(leftFork);
        pthread_testcancel();
}
```

Example: ./dinner 3 10
Good code or bad code?


```
while(!stop) {
        if((param->id % 2) == 0) {
                pthread_mutex_lock(&fork_mutexs[(param->id + 0) % num_forks]);
                pthread_mutex_lock(&fork_mutexs[(param->id + 1) % num_forks]);
        } else {
                pthread_mutex_lock(&fork_mutexs[(param->id + 1) % num_forks]);
                pthread_mutex_lock(&fork_mutexs[(param->id + 0) % num_forks]);
        }
        //eat
        result->feedings++;
        […]
  }
```

Assignments to come…

Shared Memory Parallelism

- HeatMap with Threads
- Parallel Grep with Java Monitors

Assignment 2: Questions?

- Decrypt with OpenMP (25.11. - 8.12.)
- HeatMap with OpenMP
- Matrixmultiplication with OpenMP

Accelerators

Shared Nothing Parallelism