Extending the LLVM/Clang Framework for OpenMP Metadirective Support

Seminar Supercomputer: Forschung und Innovation

Marcel Robohm

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2021-12-14

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

informatik
die zukunft

Gliederung

1 Motivation

2 Grundlagen

3 Metadirektiven

4 Dynamische Direktiven

5 Fazit

6 Literatur

Motivation [MMC20]

- Code besser portierbar
  - Dynamisch Konfiguration nach Hardware bestimmen
- Erfüllung des OpenMP-5.0-Standards [ea18]

Was ist OpenMP?

- Parallelisierung mit geteiltem Speicher
- Direkt in C(++)-Code mittels `#pragma omp`
- Zusätzlich Prozeduren in `<omp.h>`

Beispiel

```
size_t length = 1000;
int* output = new int[length];


for (int i = 0; i < length; i++) {
    output[i] = heavyCalculation(i);
}
```

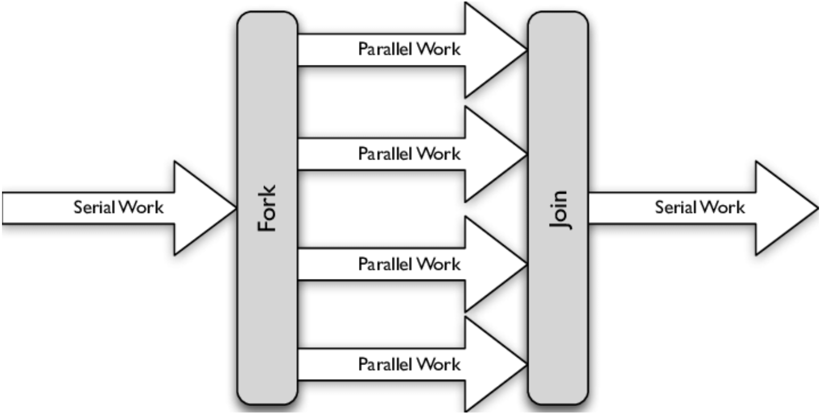
Einfache for-Schleife

Beispiel

```cpp
size_t length = 1000;
int* output = new int[length];

#pragma omp parallel for
for (int i = 0; i < length; i++) {
    output[i] = heavyCalculation(i);
}
```

Einfache for-Schleife mit Annotation

Beispiel - Fork-Join [Bec15]

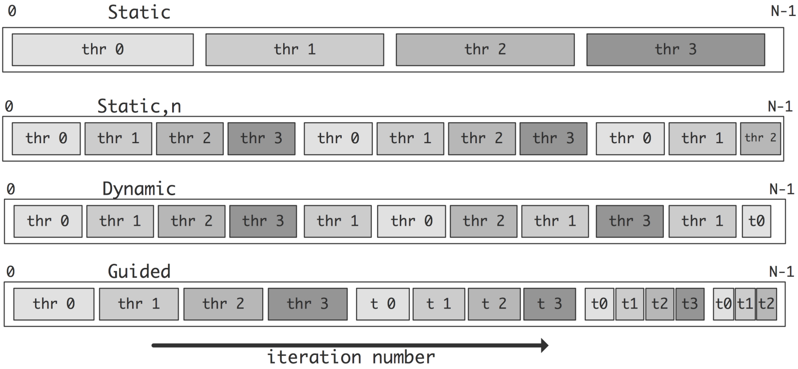


Beispiel

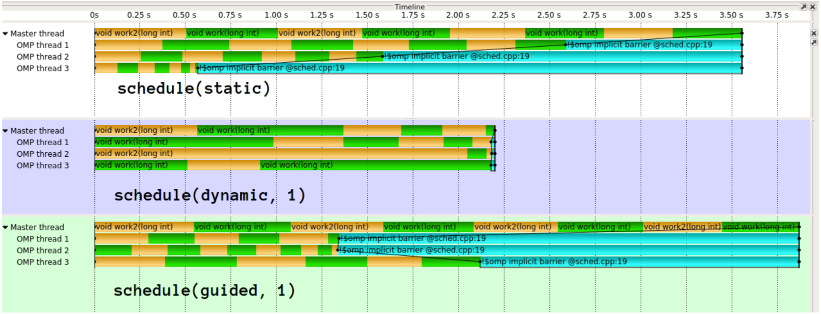
```
size_t length = 1000;
int* output = new int[length];

#pragma omp parallel for schedule(dynamic, 1)
for (int i = 0; i < length; i++) {
    output[i] = heavyCalculation(i);
}
```

Einfache for-Schleife mit Annotation und explizitem Scheduling

Scheduling - Schema [EvdGC16]



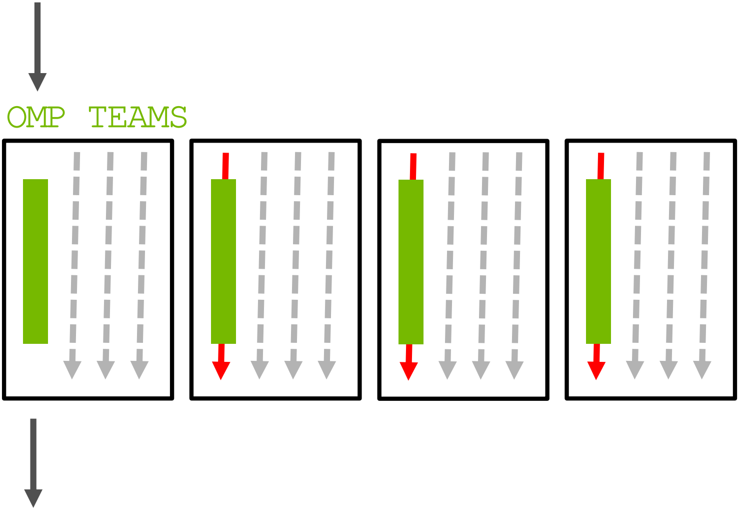
Scheduling - Beispiel [ZuIAD]



Auslagerung

- Auslagerung (Offloading) zur Steigerung der Leistung durch Spezialhardware
- Zuweisung von Teams

OpenMP - Teams [Lar18]



Auslagerung - Beispiel

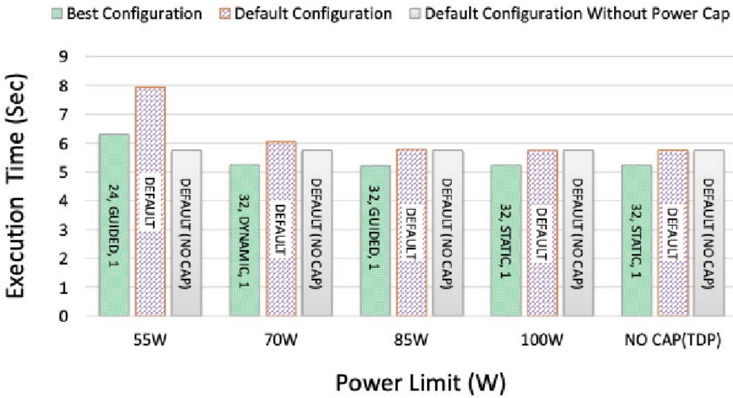
```
size_t length = 1000;
int* output = new int[length];
int* inputA = new int[length];
int* inputB = new int[length];

#pragma omp target teams distribute parallel for map(to:inputA,inputB), map(from:output)
for (int i = 0; i < length; i++) {
    output[i] = inputA[i] + inputB[i];
}
```

Auslagerung der Rechenlast mittels `target`

Herausforderung

- Heterogene Hardware
  - Speicher vs. Prozessorleistung
- Spezialhardware
  - GPU
  - FPGA
- Heterogene Einschränkungen (Stromverbrauch)

Adaption notwendig! [BCM+16]



Beispiel

```
int v1[N], v2[N], v3[N];
#if defined(nvptx)
    #pragma omp target teams distribute \
        parallel for map(to:v1,v2) map(from:v3)
    for (int i = 0; i < N; i++)
        v3[i] = v1[i] * v2[i];
#else
    #pragma omp target \
        parallel for map(to:v1,v2) map(from:v3)
    for (int i = 0; i < N; i++)
        v3[i] = v1[i] * v2[i];
#endif
```

Adaption ohne Metadirektiven

Beispiel

```
int v1[N], v2[N], v3[N];
#pragma omp target data map(to:v1,v2) map(from:v3)
#pragma omp metadirective \
    when(device=arch("nvptx"): target teams distribute parallel for) \
    default(target parallel for)
for(int i = 0; i < N; i++)
    v3[i] = v1[i] * v2[i];
```

Adaption mit Metadirektiven

Syntax

```
#pragma omp metadirective [metclause[[,] metclause]
    ↪ ... ] new-line \
```

Metadirektive

```
when(context-selector-specification:
    ↪ [directive-variant]) \
default (directive-variant) \
```

`metclause`

```
directive-name [clause[ [,] clause] ... ]
```

`directive-variant`

context-selector-specification

- Gerät
  - Art (host, nohost, cpu, gpu, fpga)
  - Befehlssatzarchitektur (x64, x86)
  - Architektur (nvptx, nvptx64, gcn)
- Implementation
  - Anbieter (nvidia, amd)
  - Erweiterungen

Syntax

```
for (idev=0; idev < omp_get_num_devices(); idev++)
    #pragma omp target device(idev)
    #pragma omp metadirective \
        when(implementation={vendor(nvidia)},
            ↪ device={arch("kepler")}: \
            teams num_teams(512) thread_limit(32) ) \
        when(implementation={vendor(amd)},
            ↪ device={arch("fiji" )}: \
            teams num_teams(512) thread_limit(64) ) \
        default(teams)
    #pragma omp distribute parallel for
    for(i = 0; i < N; i++)
        work_on_chunk(idev,i);
```

Umfangreiches Beispiel

Dynamische Adaption (bisher)

```
if(N > 1000) {
    #pragma omp target teams distribute parallel for
    for(int i = 0; i < N; i++)
        heavyCalculation(i);
} else if(N > 100 && N <= 1000) {
    #pragma omp parallel for
    for(int i = 0; i < N; i++)
        heavyCalculation(i);
} else {
    for(int i = 0; i < N; i++)
        heavyCalculation(i);
}
```

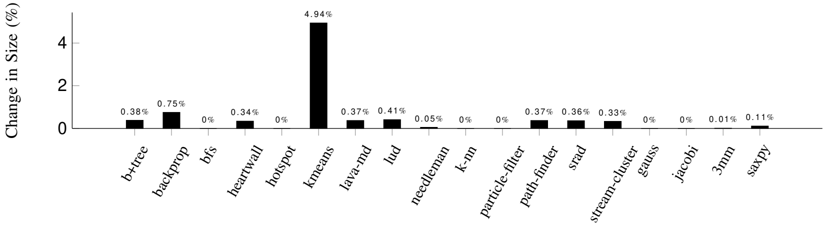
Dynamische Adaption ohne Metadirektiven

Dynamische Adaption (neu)

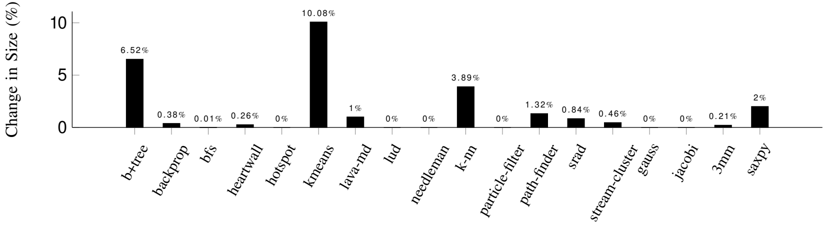
```
#pragma omp metadirective \
    when(user={condition(N > 1000)}: target teams
        ↪ distribute parallel for) \
    when(user={condition(N > 100 && N <= 1000)}:
        ↪ parallel for) \
    default()
for(int i = 0; i < N; i++)
    heavyCalculation(i);
```

Dynamische Adaption mit Metadirektiven

Speicherbedarf

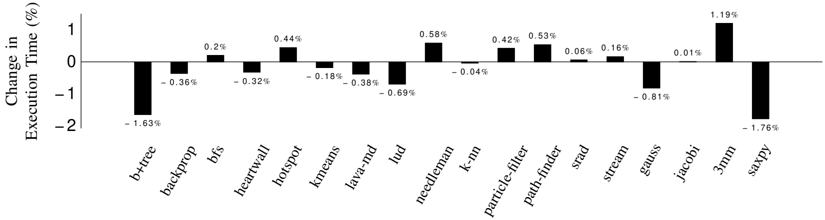


a: Summit using cuda version 10.1.105 and compute capability 7.0

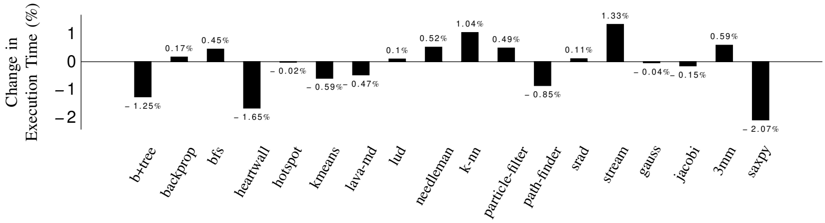


b: Seawulf using cuda version 9.1.85 and compute capability 3.5

Performanz



a: A Summit node with 2 IBM Power9 CPUs (128 threads) and NVIDIA Volta V100 GPU



b: A Seawulf node with 2 Intel Xeon E5-2683v3 CPUs (28 threads) and NVIDIA Tesla K80 GPU

Fazit

- Erfüllung des OpenMP 5.0 Standards
  - Statische Metadirektiven
- Einführung dynamischer Metadirektiven

Literatur

[BCM^{+}16] Md Abdullah Shahneous Bari, Nicholas Chaimov, Abid M Malik, Kevin A Huck, Barbara Chapman, Allen D Malony, and Osman Sarood. Arcs: Adaptive runtime configuration selection for power-constrained openmp applications. In *2016 IEEE international conference on cluster computing (CLUSTER)*. IEEE, 2016.

[Bec15] David Beckingsale. *Towards Scalable Adaptive Mesh Refinement on Future Parallel Architectures*. PhD thesis, 01 2015.

[ea18] O. Consortium et al. Openmp specification version 5.0. 2018.

[EvdGC16] Victor Eijkhout, Robert van de Geijn, and Edmond Chow. *Introduction to High Performance Scientific Computing*. Zenodo, Apr 2016.

[Lar18] Jeff Larkin. Openmp on gpus, first experiences and best practices. *NVIDIA GTC*, Mar 2018.

[MMC20] Alok Mishra, Abid M Malik, and Barbara Chapman. Extending the llvm/clang framework for openmp metadirective support. In *2020 IEEE/ACM 6th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC) and Workshop on Hierarchical Parallelism for Exascale Computing (HiPar)*. IEEE, 2020.

[ZulAD] Zulan. Openmp dynamic vs guided scheduling. *Stackoverflow*, Mar 27AD.