- Wie parallelisiere ich?
- Was kann ich parallelisieren?
- Wann macht es Sinn?

Parallelisierung voll toll, wie geht man das an

Technische Hochschule Ingolstadt

Zukunft in Bewegung

EMB^2

Vergleich zu OpenMP

Simon Varga **7. Juni 2015**

1. Embedded Multicore Building Blocks
2. kein Paper, sondern API

Inhalt

Inhalt
Parallelisierung
OpenMP
EMB^2
Beispiele
Ergebnis

1. ganz kurz nochmal zur Parallelisierung
2. kurze Vorstellung von OpenMP
3. danach auch EMB2 und unterschiede zu OpenMP
4. Beispiele in OpenMP und EMB2

Parallelisierung
Wieso macht man das?

- Multicore-Systeme
- Schnelle Ausführung

Parallelisierung
Wieso macht man das?
- Multicore-Systeme
- Schnelle Ausführung

1. auch in Autos, kleine Gebrauchsgegenstände
2. Prozessorgeschwingikeit steigt nicht mehr, Anzahl dagegen schon

Parallelisierung
Was kann ich parallelisieren?

- Sequentieller $\leftrightarrow$ Paralleler Anteil
- Logische Unabhängigkeit

Parallelisierung
Was kann ich parallelisieren?
- Sequentieller ↔ Paralleler Anteil
- Logische Unabhängigkeit

1. identifizierung
2. verschieden unabhängige Bereiche im Programm

Parallelisierung
Beispiele

src/hello.cpp

```
1 int main() {
2     std::cout << "Hello World!" << std::endl;
3 }
```


Parallelisierung
Beispiele
src/hello.cpp
int main() { std::cout << "Hello World!" << std::endl; }

1. kurzes Programm, stellvertretend für viele Sequenzielle Codezeilen
2.
3. nicht oder nur sehr schlecht zu Parallelisieren

Parallelisierung
Beispiele

src/hello.cpp

```
1 int main() {
2     std::cout << "Hello World!" << std::endl;
3 }
```


Parallelisierung
Beispiele
src/hello.cpp

1. kurzes Programm, stellvertretend für viele Sequenzielle Codezeilen
2.
3. nicht oder nur sehr schlecht zu Parallelisieren

Parallelisierung
Beispiele

src/section.cpp

```
1  int main() {
2      std::ifstream file1("file1.txt");
3      std::string line1;
4      std::getline(file1, line1);
5      std::cout << "File1: " << line1 << std::endl;
6
7      std::ifstream file2("file2.txt");
8      std::string line2;
9      std::getline(file2, line2);
10     std::cout << "File2: " << line2 << std::endl;
11
12     return 0;
13 }
```


Parallelisierung
Beispiele
src/section.cpp

1. 1. Zeile von Datei ausgeben
2.
3. unabhängige Bereiche, trennen

Parallelisierung
Beispiele

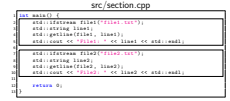
src/section.cpp

```cpp
int main() {
    std::ifstream file1("file1.txt");
    std::string line1;
    std::getline(file1, line1);
    std::cout << "File1: " << line1 << std::endl;

    std::ifstream file2("file2.txt");
    std::string line2;
    std::getline(file2, line2);
    std::cout << "File2: " << line2 << std::endl;

    return 0;
}
```


Parallelisierung
Beispiele



1. 1. Zeile von Datei ausgeben
2.
3. unabhängige Bereiche, trennen

Parallelisierung
Beispiele

src/loop.cpp

```
1 int main() {
2     for (unsigned int i = 0; i < 10; ++i) {
3         std::cout << "i = " << i << std::endl;
4     }
5 }
```


Parallelisierung
Beispiele
src/loop.cpp

1. Schleife (in der was ausgegeben wird)
2.
3. Aufteilung auf beliebig viele unabhängige Einzelteile

Parallelisierung
Beispiele

src/loop.cpp

```
1 int main() {
2     for (unsigned int i = 0; i < 10; ++i) {
3         std::cout << "i = " << i << std::endl;
4     }
5 }
```


Parallelisierung
Beispiele

1. Schleife (in der was ausgegeben wird)
2.
3. Aufteilung auf beliebig viele unabhängige Einzelteile

Parallelisierung
Beispiele

src/prefix_computation.cpp

```
1 int main() {
2     std::vector<unsigned int> range(11);
3     range[0] = 0;
4
5     for (unsigned int i = 1; i < range.size(); ++i) {
6         range[i] = range[i - 1] + 1;
7     }
8 }
```


Parallelisierung
Beispiele
src/prefix_computation.cpp

1. Schleife, aber abhängig von vorherigem Ergebnis
2. Schwierig, nicht von allen Unterstützt

Parallelisierung
Was nun?

- Programmstruktur identifizieren
- Geeignete Mittel einsetzen
- Anpassung an Zielsystem

Parallelisierung
Was nun?
- Programmstruktur identifizieren
- Geeignete Mittel einsetzen
- Anpassung an Zielsystem

1. Wie ist das Programm aufgeteilt
2. Hängt ab vom Ziel, Multiplattform, Embedded UND vom Programm selber, ist es schon fertig, wird ein neues geschrieben
3. Auf Eigenheiten des Systems eingehen, evtl. sogar auf 2,3,4 Prozessoren fest einstellen

Parallelisierung
Was setze ich ein?

- fork
- Compilerflag (-ftree-parallelize-loops)
- PThread, std::thread, ...
- OpenMP
- EMB^2

Parallelisierung
Was setze ich ein?
- fork
- Compilerflag (-ftree-parallelize-loops)
- PThread, std::thread, ...
- OpenMP
- EMB^2

1. sollte jeder kennen, (komplett) getrennte Prozesse
2. für C/C++ nur Schleifen, auch nicht alle, aber man muss nur flag setzen und kann es ausprobieren
3. Standardwerkzeuge, C++-11
4. gleich mehr
5. was kann das, kleiner Vergleich

OpenMP
Wie funktioniert das?

- Für C/C++, Fortran
- ähnlich mächtig zu PThreads, std::thread
- Präprozessormakros (#pragma)
- integriert in Compiler

OpenMP
Wie funktioniert das?
- Für C/C++, Fortran
- ähnlich mächtig zu PThreads, std::thread
- Präprozessormakros (#pragma)
- integriert in Compiler

1. Aktuell bei gcc-5.* OpenMP 4.0
2.
3. Steuerung, Angabe wie parallelisiert werden soll, Definition von Compileroptionen, wenn nicht ausgewertet werden kann, einfach ignoriert
4. Compiler muss unterstützen, aber alle gängigen

OpenMP
Wie sieht das aus?

src/openmp.cpp

```cpp
#include <iostream>
#include <omp.h>

int main() {
    int max = 10;

    #pragma omp parallel for //num_threads(2)
    for (int i = 0; i < max; i++) {
        std::cout << "Done in Thread "
                  << omp_get_thread_num()
                  << ", num = " << i << std::endl;
    }

    return 1;
}
```


OpenMP
Wie sieht das aus?
src/openmp.cpp

kleine Beispielanwendung

1. {2} OpenMP Header
2. {7} Magic, man kann die Anzahl der Threads angeben
3. {10} auf Threadnummer zugreifen

OpenMP
Wie sieht das aus?

src/openmp.out

```
1  Done in Thread Done in Thread Done in Thread 10, num = 33,
       num = , num = 80Done in Thread
2
3
4  2, num = 6Done in Thread 1, num = 4
5  Done in Thread 1, num = 5
6
7  Done in Thread 2, num = 7
8  Done in Thread 3, num = 9
9  Done in Thread 0, num = 1
10 Done in Thread 0, num = 2
```


OpenMP
Wie sieht das aus?
src/openmp.out

Output bei 4 Threads, mehrere Sachen sehen:

1. mehrere schreiben gleichzeitig, durcheinander, leerzeilen
2. nicht in fester Reihenfolge
3.
4. Wie siehts mit EMB2 aus

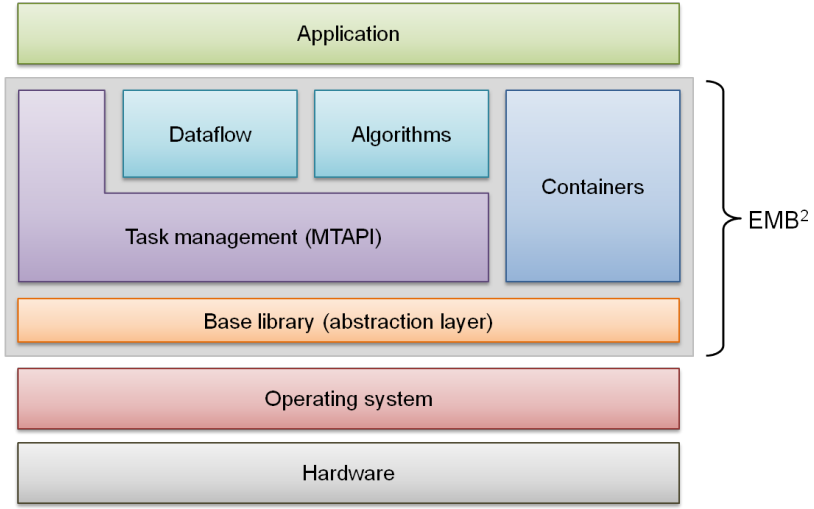
EMB^2
Embedded Multicore Building Blocks

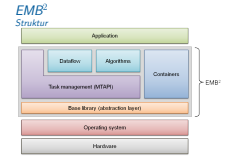
- von Siemens entwickelt
- vor allem für Embedded Systeme
- C/C++
- Ziel: Abstraktion von (low-Level) Thread-Management
- Unterstützt Task Prioritäten
- Aufgebaut auf MTAPI
  - standardisiertes Programminterface
  - Unterstützt symmetrische und asymmetrische Prozessoren

EMB^2
Embedded Multicore Building Blocks
- von Siemens entwickelt
- vor allem für Embedded Systeme
- C/C++
- Ziel: Abstraktion von (low-Level) Thread-Management
- Unterstützt Task Prioritäten
- Aufgebaut auf MTAPI
  - standardisiertes Programminterface
  - Unterstützt symmetrische und asymmetrische Prozessoren

1. ab 1.10.2014 Open Source, aktuell 0.3.0 vom 27. Mai
2. aber auch für normale Software geeignet
3.
4. man muss sich nicht mehr selber drum kümmern
5. hat geheißen, das machen nicht viele
6. um Parallelisierung auf Embedded Systeme zu bringen
7. asymmetrische: nicht alle haben gleiche Priorität/Aufgabenverteilung, z.B. nur einer kann Betriebssystem-Code ausführen

EMB²
Struktur





1. oben Applikation, die die Bibliothek benutzt
2. eigentliche Bib, setzt auf MTAPI für die Taskverwaltung auf
3. bietet zusätzliche Klassen und Funktionen, für viel genutzte Algorithmen
4. bietet auch eigene Containerklassen, deren Funktionen parallel bearbeitet werden
5. läuft auf den verschiedensten Prozessorarchitekturen (x86, ARM)

Code

kommt gleich danach → Codeblocks

EMB^2
Unterschied zu OpenMP

+ neue Software
- vorhandene Software
- Algorithmen, Datenstrukturen

EMB^2
Unterschied zu OpenMP
+ neue Software
- vorhandene Software
- Algorithmen, Datenstrukturen

1. wenn Software neu geschrieben wird
2. relativ viel umschreiben
3. bietet Standardalgorithmen und Datenstrukturen

Code

in Codeblocks

Ergebnis
Aufwand

- OpenMP relativ einfach
- EMBB für neue Programme

Ergebnis
Aufwand
- OpenMP relativ einfach
- EMBB für neue Programme

1. bei bestehenden Programmen
2. etwas schwierig die genau gleichen Beispiele, wenn man eh neu schreibt, kann man die beste Logik benutzen

Ergebnis
Zeiten

	Std	OpenMP	SpeedUp	EMBB	SpeedUp
Schleife	1.000.957 μs	315.302 μs	3,17	6.655 μs	150,41
Abh. Schleife	900.662 μs	303.323 μs	2.97	10.291 μs	87,52
Reduce	1.000.736 μs	301.900 μs	3,13	1.066 μs	938,78
Reduce Custom	1.000.742 μs	301.928 μs	3,15	1.089 μs	918,96
Section	200.231 μs	100.102 μs	2,00	196 μs	1021,59
Sort	1.901.300 μs	1.609.059 μs	1,18	5.227 μs	363,75

Ergebnis
Zeiten

	Std	OpenMP	SpeedUp	EMBB	SpeedUp
Schleife	1.000.957 μs	315.302 μs	3,17	6.655 μs	150,41
Abh. Schleife	900.662 μs	303.323 μs	2.97	10.291 μs	87,52
Reduce	1.000.736 μs	301.900 μs	3,13	1.066 μs	938,78
Reduce Custom	1.000.742 μs	301.928 μs	3,15	1.089 μs	918,96
Section	200.231 μs	100.102 μs	2,00	196 μs	1021,59
Sort	1.901.300 μs	1.609.059 μs	1,18	5.227 μs	363,75