



2015-06-07 EMB²

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

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

Parallelisierung voll toll, wie geht man das an

*Zukunft in
Bewegung*

EMB²

Vergleich zu OpenMP

Simon Varga

7. Juni 2015

2015-06-07 EMB²

Embedded Multicore Building Blocks





- 1 Parallelisierung
- 2 OpenMP
- 3 EMB²
- 4 Beispiele

2015-06-07 EMB²

└ Inhalt

Inhalt

- Parallelisierung
- OpenMP
- EMB²
- Beispiele

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

2015-06-07

EMB²

- └ Parallelisierung
 - └ Wieso macht man das?
 - └ Parallelisierung

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

Parallelisierung

Was kann ich parallelisieren?



2015-06-07

EMB²

- └ Parallelisierung
 - └ Was kann ich parallelisieren?
 - └ Parallelisierung

Parallelisierung
Was kann ich parallelisieren?

- Sequentieller ↔ Paralleler Anteil
- Logische Unabhängigkeit

- Sequentieller ↔ Paralleler Anteil
- Logische Unabhängigkeit

1. identifizierung
2. verschieden unabhängige Bereiche im Programm



src/hello.cpp

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

2015-06-07

EMB²

└ Parallelisierung

└ Beispiele

└ Parallelisierung

src/hello.cpp

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

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



src/hello.cpp

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

2015-06-07

EMB²

- └ Parallelisierung
- └ Beispiele
- └ Parallelisierung



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



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 }
```

2015-06-07

EMB²

└ Parallelisierung

└ Beispiele

└ Parallelisierung

```
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 }
```

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



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 }
```

2015-06-07

EMB²

- └ Parallelisierung
- └ Beispiele
- └ Parallelisierung

```
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 }
```

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



src/loop.cpp

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

2015-06-07

EMB²

└ Parallelisierung

└ Beispiele

└ Parallelisierung

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

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

src/loop.cpp

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

2015-06-07

EMB²

└ Parallelisierung

└ Beispiele

└ Parallelisierung

src/loop.cpp

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

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

Parallelisierung

Was nun?



2015-06-07

EMB²

- └ Parallelisierung
 - └ Was nun?
 - └ Parallelisierung

Parallelisierung
Was nun?

- Programmstruktur identifizieren
- Geeignete Mittel einsetzen
- Anpassung an Zielsystem

- 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
- EMB²

2015-06-07

EMB²

- └ Parallelisierung
 - └ Was setze ich ein?
 - └ Parallelisierung

Parallelisierung
Was setze ich ein?

- fork
- Compilerflag (-ftree-parallelize-loops)
- PThread, std::thread, ...
- OpenMP
- EMB²

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



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

2015-06-07

EMB²

└ OpenMP

└ Wie funktioniert das?

└ OpenMP

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

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

OpenMP

Wie sieht das aus?



src/openmp.cpp

```
1 #include <iostream>
2 #include <omp.h>
3
4 int main() {
5     int max = 10;
6
7     #pragma omp parallel for //num_threads(2)
8     for (int i = 0; i < max; i++) {
9         std::cout << "Done in Thread "
10                 << omp_get_thread_num()
11                 << ", num = " << i << std::endl;
12     }
13
14     return 1;
15 }
```

2015-06-07

EMB²

└ OpenMP

└ Wie sieht das aus?

└ OpenMP

OpenMP
Wie sieht das aus?

```
src/openmp.cpp
1 #include <iostream>
2 #include <omp.h>
3
4 int main() {
5     int max = 10;
6
7     #pragma omp parallel for //num_threads(2)
8     for (int i = 0; i < max; i++) {
9         std::cout << "Done in Thread "
10                 << omp_get_thread_num()
11                 << ", num = " << i << std::endl;
12     }
13
14     return 1;
15 }
```

kleine Beispielanwendung

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



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
```

2015-06-07

EMB²

└ OpenMP

└ Wie sieht das aus?

└ OpenMP

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

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



- 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

2015-06-07

EMB²

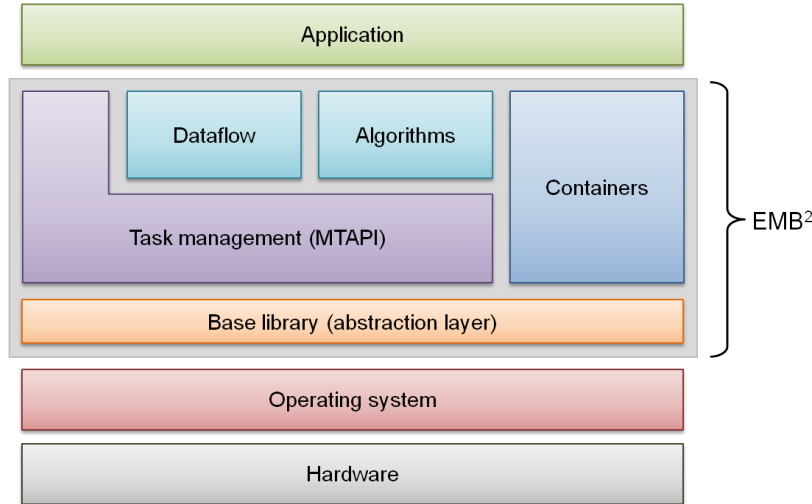
└ EMB²

└ Embedded Multicore Building Blocks

└ EMB²

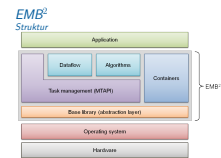
- von Siemens entwickelt
- vor allem für Embedded Systeme
- C/C++
- Ziel: Abstraktion von (low-Level) Thread-Management
- Unterstützt Task Prioritäten
- Aufgebaut auf MTAPI
 - standardisiertes Programminterface
 - Unterstützt symmetrische und asymmetrische Prozessoren

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



2015-06-07

EMB²
└ EMB²
└ Struktur
└ EMB²



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

zu lang → Codeblocks



Code

2015-06-07

EMB²

└ Beispiele

└ Beispiele

in Codeblocks