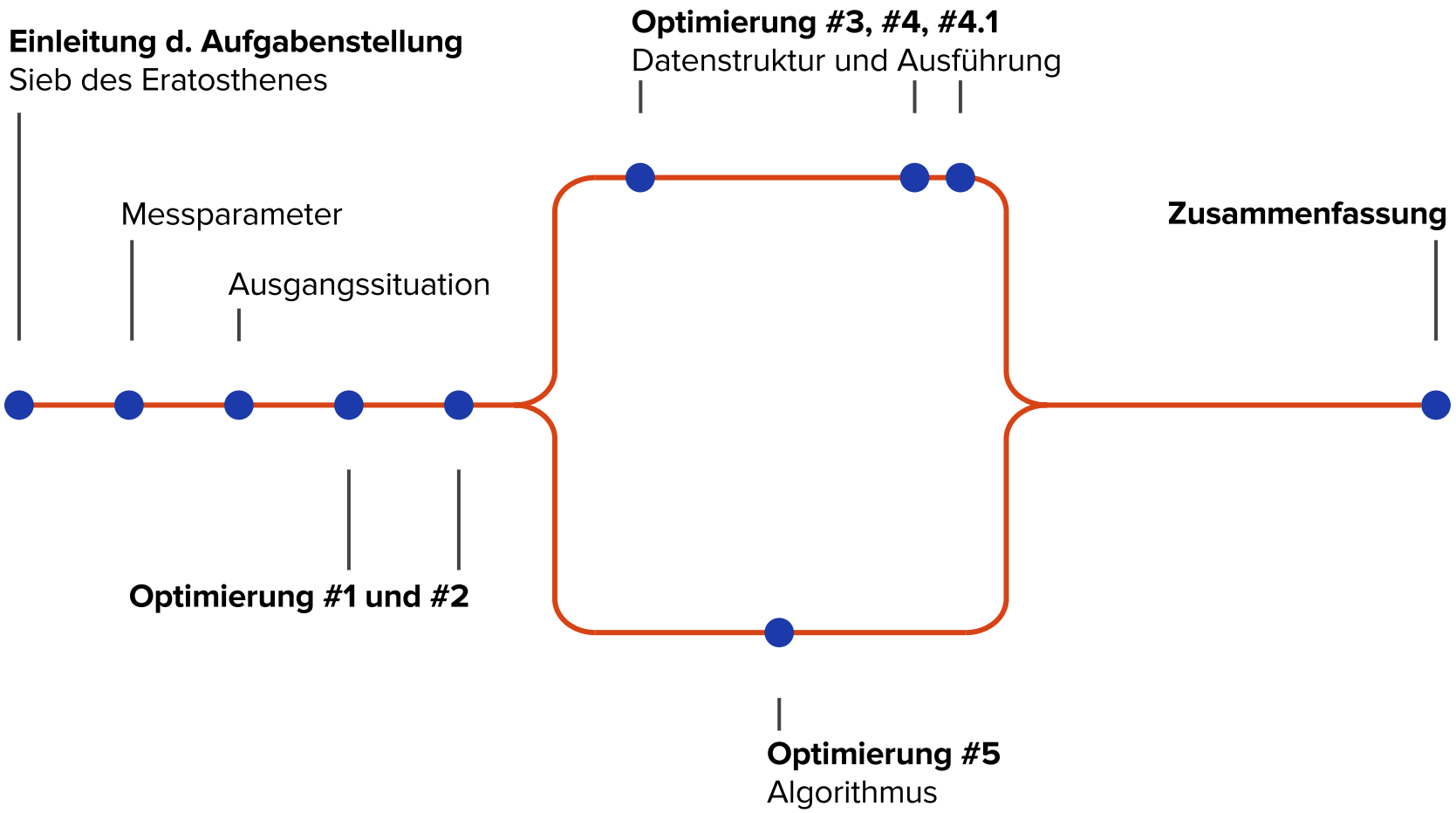
Effiziente Programme

Optimierung: *Sieb des Eratosthenes*

20.01.2017

EP Gruppe: #100
Fabian Theuretzbacher (1426066) | Sylvia Winkler (1428557)

Agenda


Einleitung d. Aufgabenstellung
Sieb des Eratosthenes
Messparameter
Ausgangssituation
Optimierung #1 und #2
Optimierung #3, #4, #4.1
Datenstruktur und Ausführung
Optimierung #5
Algorithmus
Zusammenfassung

Aufgabenstellung

- Anzahl der Primzahlzwillinge ≤ n
- Berechnung von *P* ➔ Sieb des Eratosthenes

Funktionsweise: n = 50

	3	5	7	9
11	13	15	17	19
21	23	25	27	29
31	33	35	37	39
41	43	45	47	49

	3	5	7	9
11	13	15	17	19
21	23	25	27	29
31	33	35	37	39
41	43	45	47	49

	3	5	7	9
11	13	15	17	19
21	23	25	27	29
31	33	35	37	39
41	43	45	47	49

… bis ≤ √(n) …

	3	5	7	9
11	13	15	17	19
21	23	25	27	29
31	33	35	37	39
41	43	45	47	49

Messung

Messbarkeit von Performanz:

- **perf**
  cycles (u+k), instructions, branch-misses, L1-*, LLC-*
- **/usr/bin/time -v**
  ➜ maximum resident set size
- **valgrind**
  ➜ heap memory usage at exit
- **gcov**
  Diskrete Laufzeitanalyse

Kompilierung und Ausführung

- `gcc -O3 -lm [...]`
- `./pi 1 000 000 000`
- Durchschnitt aus 3x

Ausgangssituation

Laufzeit		
cycles:u	37,109,028,373	-
cycles:k	410,309,417	-
instructions	16,409,716,981	-
branch-miss	106,744,343	-
L1-load-miss	332,517,306	-
L1-store-miss	6,324,332,321	-
LLC-load-miss	123,945,205	-
LLC-store-miss	752,550,644	-
Runtime (sec)	11.15	-
Speicherverbrauch		
Max RSS (KB)	488,780	-
Heap Mem. (B)	499,999,999	-

Optimierung #1

- Äußere Schleife läuft unnötigerweise bis **n/2**
- Prüfung nur innerhalb $2 \leq p \leq \sqrt{n}$ notwendig

```
for(i = 0 ; i <= size; i++) {
   if(*(flags+i)) {
      prime = i + i + 3;
      for(k = i + prime ; k<=size ; k+=prime) {
         *(flags+k)=0;
      }
   }
}
```

Optimierung #1 - Ergebnisse

Laufzeit		
cycles:u	24,215,116,938	-34.7%
cycles:k	374,534,833	-8.72%
instructions	10,366,209,518	-36.8%
branch-miss	52,515,597	-50.8%
L1-load-miss	168,739,489	-49.3%
L1-store-miss	5,147,824,713	-18.6%
LLC-load-miss	51,897,729	-58.1%
LLC-store-miss	523,579,386	-30.4%
Runtime (sec)	7.32	34.3%
Speicherverbrauch		
Max RSS (KB)	488,780	0%
Heap Mem. (B)	499,999,999	0%

```
./pi2 10000

0.gocv
 5000: for(i=0 ; i <= size ; i++) {
 4999:   if(*(flags+i)) {
 1228:     prime = i + i + 3;
10062:       for(...) {
 8834:         *(flags+k)=0;
     :       }
     :     }
     :   }
     : }

1.gocv
    1: long outerlimit = sqrt(m+1);
  101: for(i=0 ; i <= outerlimit; i++) {
  100:   if(*(flags+i)) {
   45:     prime = i + i + 3;
 7244:       for(...) {
 7199:         *(flags+k)=0;
     :       }
     :     }
     :   }
     : }
```

Optimierung #2

- Innere Schleife muss Vielfache erst ab p^2 prüfen

```
./pi2 900
Prime:  3++++++++++++++++++++++++++++++++++++++++ [...]
Prime:  5-++-++-++-++-++-++-++-++-++-++-++-++-++ [...]
Prime:  7--+-++-++-+--++--+-++-++-+--++--+-++-++-+- [...]
Prime: 11----++-++-+--++--+-++-+--+--++--+-++--+-
Prime: 13-----+-++-+--++--+-++-+--+--++--+-
Prime: 17-------++-+--++--+-++-+--
Prime: 19--------+-+--++--+-++-+
Prime: 23----------+--++--+-
Prime: 29------------++
Prime: 31--------------
Prime: 37-----------
Prime: 41----------
Prime: 43---------

35 Primzahl-Zwillinge <= 900
```

Prime 3	Prime 5	Prime 7
9	15	21
15	25	35
21	35	49
27	45	63
33	55	77
39	65	...
45	75	
51	...	

Optimierung #2 - Ergebnisse

Laufzeit		
cycles:u	21,310,456,295	-12.0%
cycles:k	371,849,277	-0.71%
instructions	9,962,948,783	-3.89%
branch-miss	52,928,586	+0.79%
L1-load-miss	117,939,808	-30.1%
L1-store-miss	5,078,957,998	-1.34%
LLC-load-miss	27,324,933	-47.4%
LLC-store-miss	466,032,838	-11.0%
Runtime (sec)	6.45	-11.9%
Speicherverbrauch		
Max RSS (KB)	488,784	0%
Heap Mem. (B)	499,999,999	0%

```
./pi2 10000

1.gocv
  101: for(i=0 ; i <= outerlimit; i++) {
  100:   if(*(flags+i)) {
   45:     prime = i + i + 3;
 7244:       for(...]) {
 7199:         *(flags+k)=0;
      :       }
      :     }
      :   }
      : }

2.gocv
  101: for(i=0 ; i <= outerlimit; i++) {
  100:   if(*(flags+i)) {
   45:     prime = i + i + 3;
 6040:       for(k = ...]) {
 5995:         *(flags+k)=0;
      :       }
      :     }
      :   }
      : }
```

Optimierung #3

- Mitzählen der Primzahlzwillinge in **`sieve(...)`**
- Eliminierung von **`n/2`** Schleifendurchläufen
- ➜ Anpassung von Optimierung #1

SPOILER-ALERT

- Laufzeit von 6.45s ➜ **6.65s verschlechtert**
- Weitere Optimierungen notwendig
  - Zusammenlegung der IF-Statements
  - Neue Anordnung der IF-Statements
  - Arithmetik mit Flags

Optimierung #3 - Ergebnisse

Laufzeit		
cycles:u	20,208,129,747	-5.12%
cycles:k	364,877,540	-1.87%
instructions	12,220,499,996	+22.7%
branch-miss	90,242	-99.8%
L1-load-miss	120,621,192	+2.27%
L1-store-miss	5,107,155,196	+0.56%
LLC-load-miss	27,338,308	+0.05%
LLC-store-miss	466,020,941	0.0%
Runtime (sec)	6.12	-5.12%
Speicherverbrauch		
Max RSS (KB)	488,784	0%
Heap Mem. (B)	499,999,999	0%

▲ Instruktionen → Äußere Schleife iteriert wieder bis **n/2**.

Optimierung #4

- **Aktuell:** 1 *Byte* repräsentiert eine (Prim)zahl

							3								5				
-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-

- **Besser:** 1 *Bit* repräsentiert eine (Prim)zahl

3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41
1	1	1	0	1	1	0	1	1	0	1	0	0	1	1	0	0	1	0	1

```
//...
#define CHECK_BIT(var, pos)           ((var) & (1 << (pos)))
#define CHECK_BIT_ARRAY(arr, pos)     CHECK_BIT( (arr)[((pos) >> 3)], ((pos) & 0b111) )
//...
```

Optimierung #4 - Ergebnisse

Laufzeit		
cycles:u	17,037,558,424	-15.7%
cycles:k	72,106,800	-80.2%
instructions	19,514,903,780	+59.7
branch-miss	55,834,327	+600x
L1-load-miss	509,114,365	+320%
L1-store-miss	53,405,603	-99.0%
LLC-load-miss	247,319,041	+805%
LLC-store-miss	206	-99.9%
Runtime (sec)	5.09	-16.8%
Speicherverbrauch		
Max RSS (KB)	61,540	-87.4%
Heap Mem. (B)	62,500,000	-87.5%

- ▲ Instruktionen → Bit-Operationen
- ▲ Branch-Misses → IF-Statements haben sich verschlechtert
- ▼ Store-Misses → Höhere Lokalität
- ▼ Speicherverbrauch: 8x effizienter

Optimierung #4.1

- Verschlechterungen:
  - ▲ Instruktionen
  - ▲ Branch-Misses
- Anpassungen:
  - Dedizierte **`twins()`**-Funktion
    - bessere Branch-Prediction!?
    - Weniger Schleifendurchläufe (nur bis **`sqrt(n)`**)
  - **`twins()`**: Schleifendurchlauf über all Bits
    - Byte-Counter wird 8x redundant berechnet
    - Aufteilen in *Byte*-Schleife und *Bit*-Schleife
    - Überarbeitetes memset

1	1	1	0	1	1	0	1	1	0	1	0	0	1	1	0	0	1	0	1
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**
0								**1**								**2**			

Optimierung #4.1 - Ergebnisse

		Opt. 4	Opt. 3
Laufzeit			**LZ**
cycles:u	14,576,932,431	-14.4%	-27.9%
cycles:k	69,781,916	-3.22%	-80.9%
instructions	15,521,265,442	-20.5%	+27.0%
branch-miss	7,081,433	-87.3%	+77x
L1-load-miss	515,901,258	+1.33%	+327%
L1-store-miss	54,373,253	+1.81%	-98.9%
LLC-load-miss	247,695,479	+0.15%	+806%
LLC-store-miss	208	+0.97%	-99.9%
Runtime (sec)	4.36	-14.3%	-28.8%
Speicherverbrauch			**SV**
Max RSS (KB)	61,540	0%	-87.4%
Heap Mem. (B)	62,500,000	0%	-87.5%

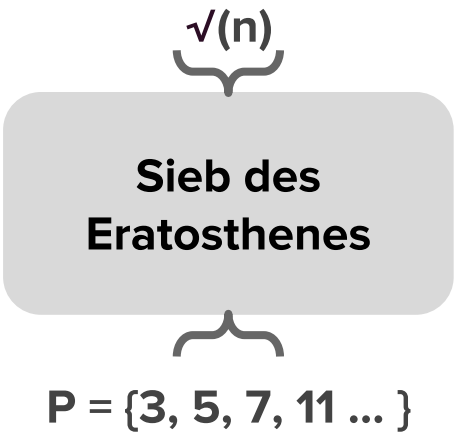
▲ Optimierung kommt mit komplexerer Codebasis

Optimierung #5

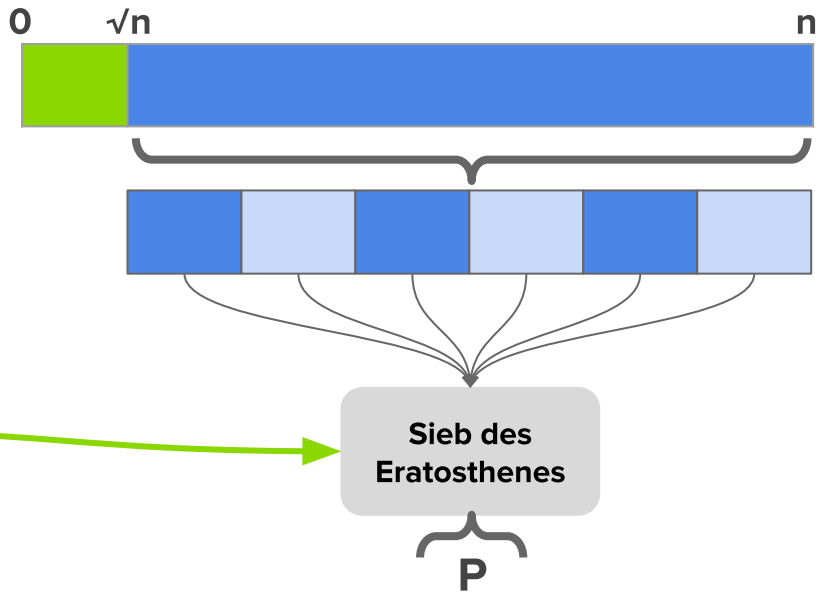
Segmented Sieve:

- Gegenmaßnahme zu O(n) Speicherverbrauch
- Lokalitätseigenschaft
- Ausnützen des L1-Cache

Schritt 1:



Schritt 2:



Optimierung #5 - Ergebnisse

	Opt. 4	
Laufzeit		
cycles:u	7,569,107,284	-55.6%
cycles:k	4,401,056	-93.9%
instructions	10,594,353,665	-45.7%
branch-miss	64,352,541	+15.3%
L1-load-miss	3,334,360	-99.3%
L1-store-miss	1,425,311,934	+25.7x
LLC-load-miss	477,636	-99.8%
LLC-store-miss	44,112,110	214k
Runtime (sec)	2.26	-55.6%
Speicherverbrauch		
Max RSS (KB)	788	-98.7%
Heap Mem. (B)	15,810*	-99.9%

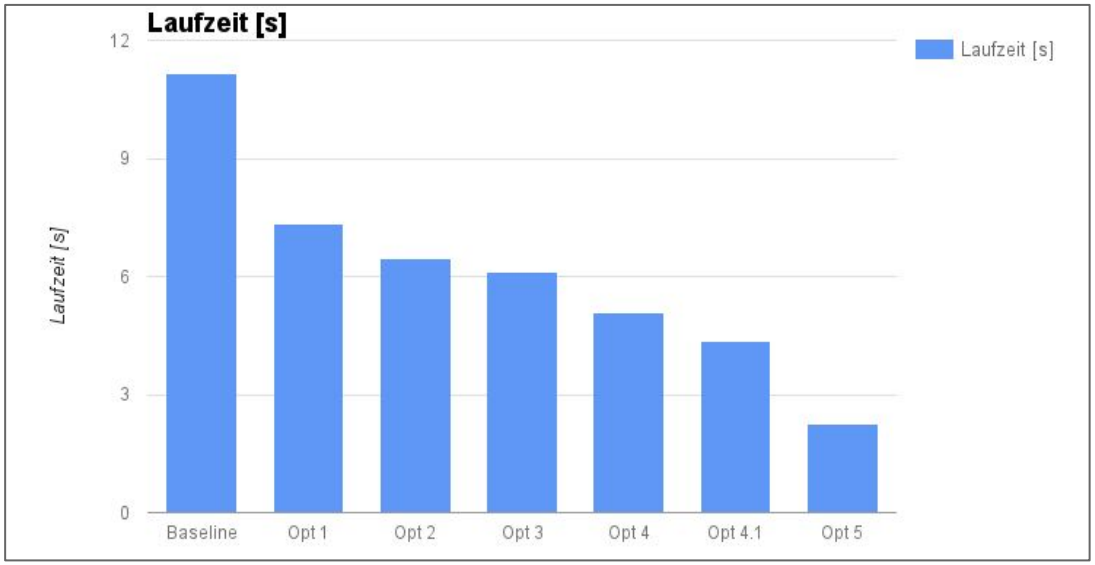
- “Max. Heap Memory Usage: 272 KBytes”
- “Total allocated Memory: 500 MBytes”

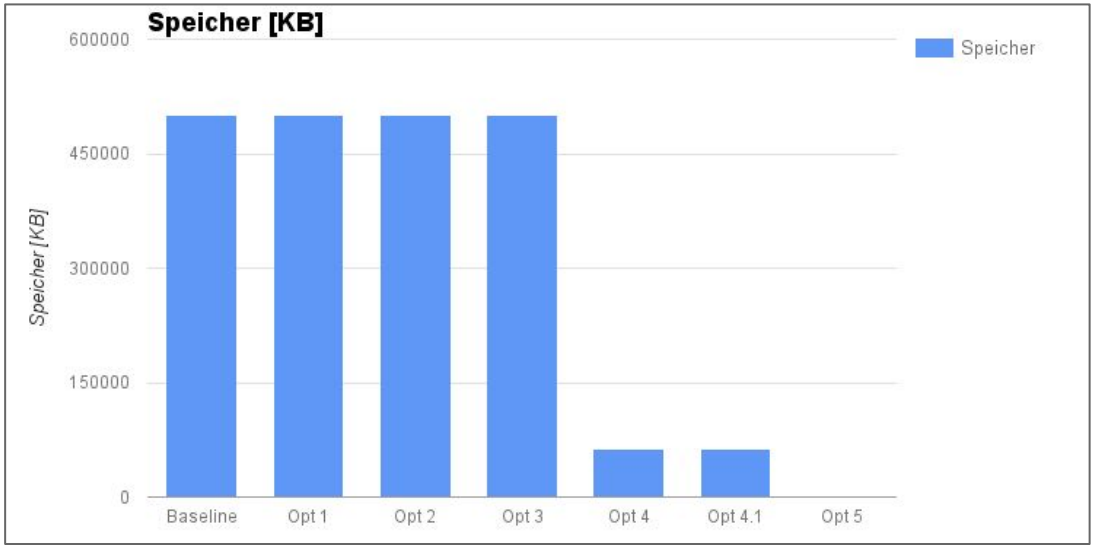
* Beim Beenden des Programmes

Optimierung #5 - “Drawbacks”

- Keine Verwendung von Bitarray
  - ⇒ Laufzeit verschlechtert sich
- Keine integrierte **`twins()`**-Funktion
- Keine p^2-Optimierung
  - ⇒ Laufzeit verbessert sich **nicht**

Zusammenfassung





{?!}

EP Gruppe: #100
Fabian Theuretzbacher (1426066) | Sylvia Winkler (1428557)