

# Effiziente Programme WS10/11

tuning stuff for fun and profit

David Berger, Martin Wieser, Serap Kadam, Alexander Duml

Januar 14, 2011

# Warnings

- oprofile statt papiex
- Davids PC statt g0

- low-overhead
- Performance Counter bei unseren Tests
- Profilbasierend (Systemweit)
- akkumulativ  $\Rightarrow$  1000 Durchläufe/Test

# oprofile - Beispielsession

```
oprofile --start  
./test shortest-path  
oprofile -cl shortest-path  
opannotate --source --assembly shortest-path  
opcontrol --reset
```

# oprofile - Beispielsession

---

1121706	92.6244	optimize_rewrite
1121706	100.000	optimize_rewrite [self]

---

78016	6.4421	cost_codesize
78016	100.000	cost_codesize [self]

---

11296	0.9328	main
11296	100.000	main [self]

---

7	5.8e-04	__libc_csu_init
7	100.000	__libc_csu_init [self]

---

1	8.3e-05	_init
1	100.000	_init [self]

---

# oprofile - Beispielsession

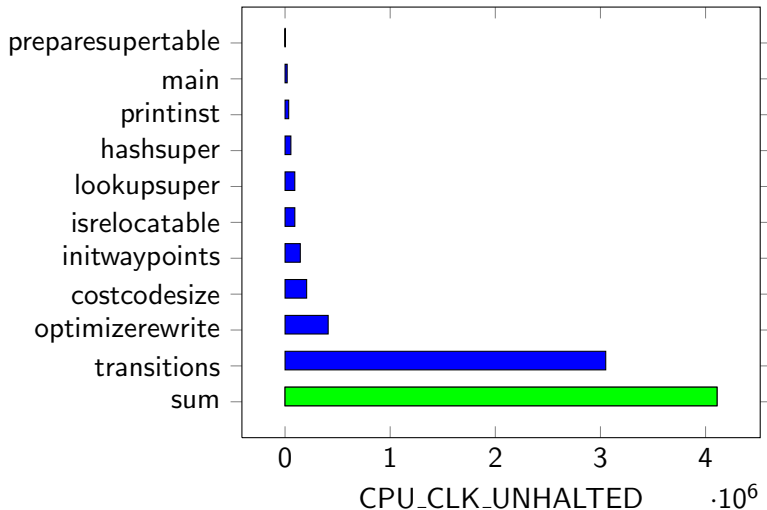
---

```
1121706  92.6244  optimize_rewrite
1121706  100.000  optimize_rewrite [self]
```

---

CPU\_CLK\_UNHALTED - unhalted cycles welche CPU in Funktion verbringt

# Ursprungsprogramm



# Ask not what you can do for your compiler

... ask your compiler what he can do for you.

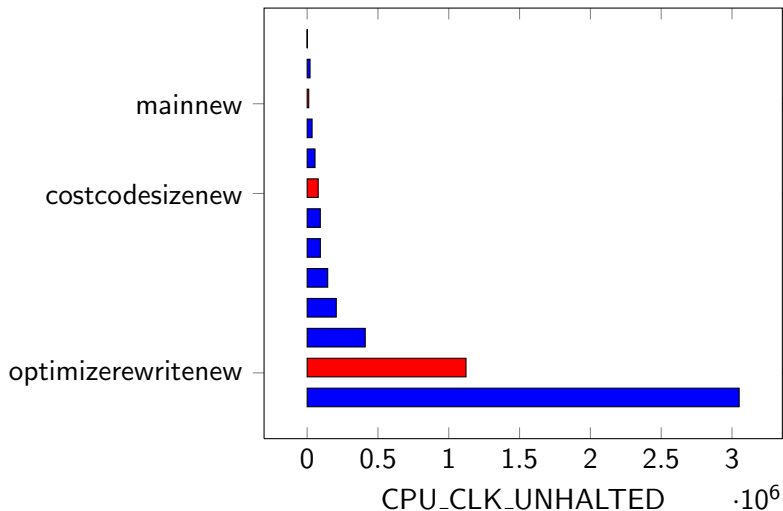




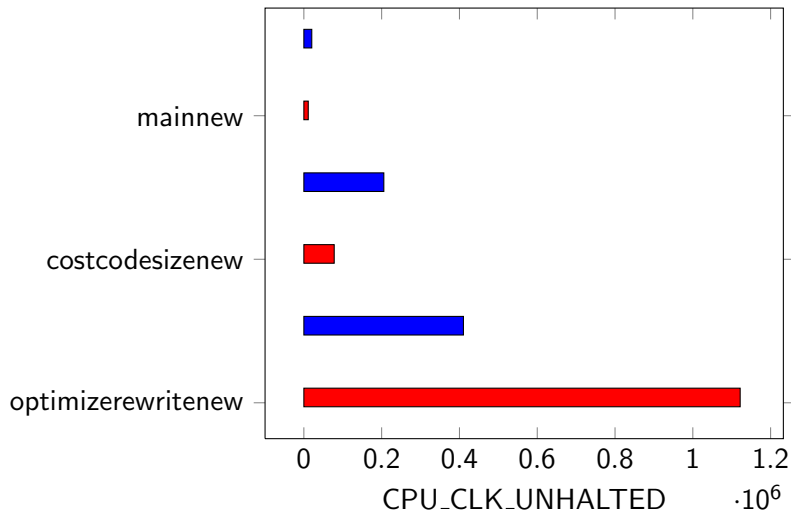
# Ask not what you can do for your compiler

- -O3 statt -O0
- $\Rightarrow$  mass inlining
- $\Rightarrow$  unrolling
- $\Rightarrow$  a lot of other optimizations

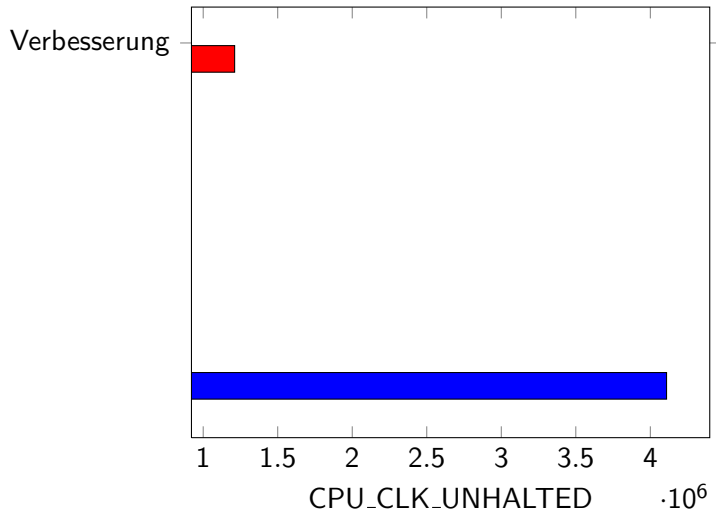
# Ask not what you can do for your compiler



# Ask not what you can do for your compiler



# Ask not what you can do for your compiler



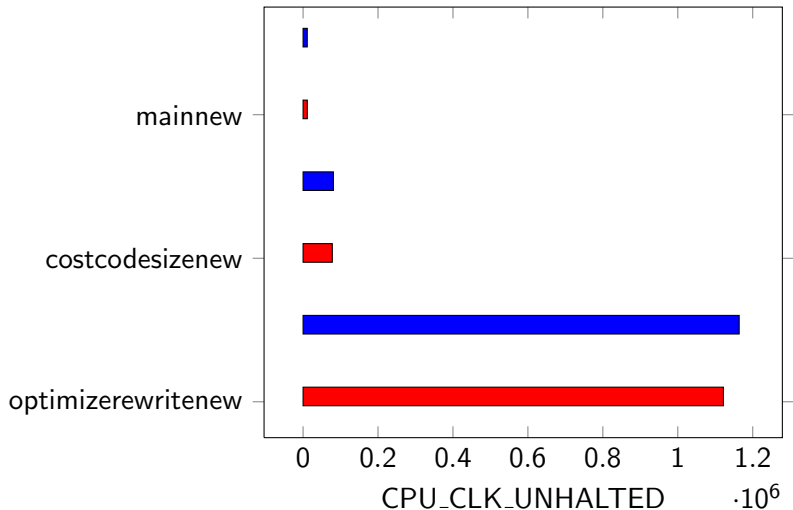
# Ask not what you can do for your compiler

... but don't ask too much of him.

# Ask not what you can do for your compiler

- -funroll-loops
- $\Rightarrow$  GCC unrolled Schleifen aggressiv
- $\Rightarrow$  Codesize größer
- $\Rightarrow$  Performance schlechter

# Ask not what you can do for your compiler

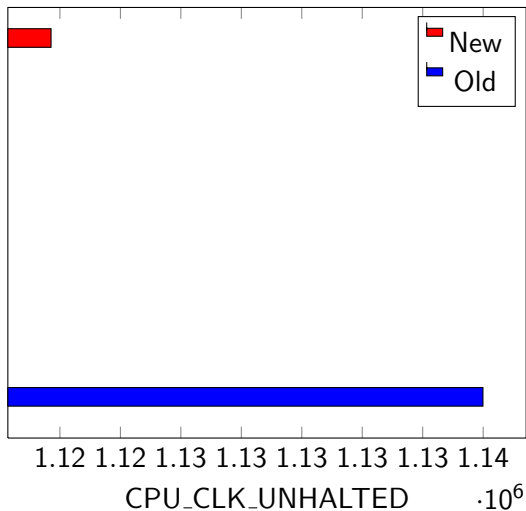


# Lets actually do something

- ersetze `ss_cost` durch `cost_codesize`
- `cost_codesize` verschwindet komplett
- main und `optimize_rewrite` marginal besser



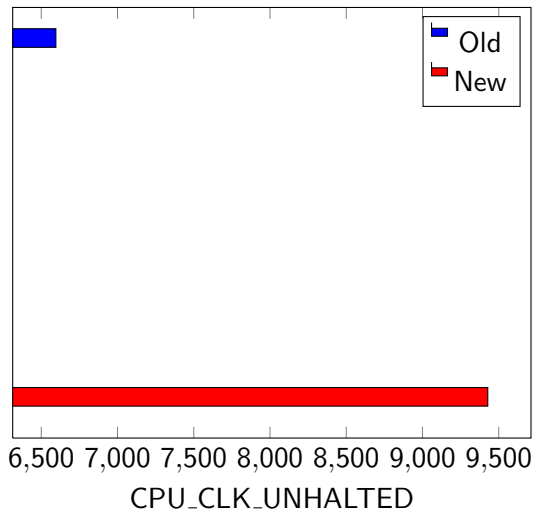
## Lets actually do something



# Lets actually do something

- Verbesserung main Schleife

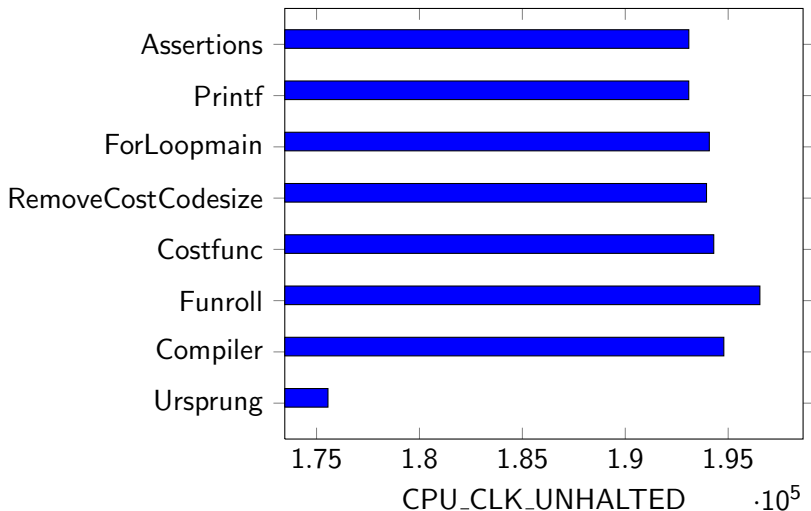
# Lets actually do something



# That's it?

- GCC leistet ganze Arbeit
- viele unserer anderen "Optimierungen" schlecht
- Codesize kann man noch verbessern

# Codesize



## papiex Endergebnisse

```
PAPI_TOT_CYC .....4.27796 e+08  
PAPI_TOT_INS .....5.30222 e+08  
PAPI_BR_MSP .....1.28136 e+06  
PAPI_FP_OPS .....487
```

Listing 1: "Ursprung"

```
PAPI_TOT_CYC .....1.44674 e+08  
PAPI_TOT_INS .....2.2851 e+08  
PAPI_BR_MSP .....1.04732 e+06  
PAPI_FP_OPS .....507
```

Listing 2: "Ursprung -O3"

```
PAPI_TOT_CYC .....1.42435 e+08  
PAPI_TOT_INS .....2.00493 e+08  
PAPI_BR_MSP .....1.42997 e+06  
PAPI_FP_OPS .....486
```

Listing 3: "Endergebnis"

# Food for Thought

<http://leto.net/docs/C-optimization.php#Compute-bound>

<http://people.redhat.com/drepper/cpumemory.pdf>

<http://www.fefe.de/dietlibc/diet.pdf>

<http://www.fefe.de/know-your-compiler.pdf>