Pruning the clown sheep of the family

Philipp Czerner

Remark on the previous assignments

```
pthread_attr_t attrs;
pthread_attr_init(&attrs);
int main_cpu = sched_getcpu();
int num_cores = sysconf(_SC_NPROCESSORS_ONLN);
for (int i = 0; i < num_threads; ++i) {
    int thread_cpu = (main_cpu + i) % num_cores;
    cpu_set_t cpu;
    CPU_ZERO(&cpu);
    CPU_SET(thread_cpu, &cpu);
    pthread_attr_setaffinity_np(&attrs, sizeof(cpu_set_t), &cpu);
    pthread_create(&threads[i], &attrs, thread_function, &args[i]);
}
pthread_attr_destroy(&attrs);
```

- Last time, OpenMP was still faster than even my crazy initialisation
- A large part of that is setting CPU affinity

Overview

optimisation	time	change	difficulty	generality
baseline	1			
precomputed lookup table	0.00000354	281908	🤡	😁
OpenMP	0.00028949	0.0123	😒	😸
total	0.00000354	281908		

- Baseline already has optimisations enabled, although only for my functions

Baseline

```
void traverse_rec(tree* node) {
    if (node != NULL) {
        node->IQ = compute_IQ(node->data);
        genius[node->id] = node->IQ;
        traverse_rec(node->left );
        traverse_rec(node->right);
    }
}
int is_prime_or_zero_or_one(int n) {
    for (int i = 2; i <= n/2; ++i)
        if (n % i == 0) return false;
    return true;
}
int compute_IQ(int data) {
    int sum = 0;
    for (int i = 0; i < data; i++)
        sum += is_prime_or_zero_or_one(i);
    return 70 + (sum % 100)*(sum % 100)*(sum % 100)*(sum % 100)/1000000;
}
```

- The code of `compute_IQ` will be irrelevant

Precomputed lookup table ($\times 281908$)

```
void traverse_rec(tree* node) {
    if (node != NULL) {
        node->IQ = lookup_iq[node->data];
        genius[node->id] = node->IQ;
        traverse_rec(node->left );
        traverse_rec(node->right);
    }
}
```

- **Context:** There are only 9113 possible values for `node->data`
- **Context:** The function `compute_IQ` easily takes more than 1000 cycles to compute, even if optimised

Working inside arbitrary restrictions

- Computing the lookup at runtime is unnecessary, takes too long
- Instead, store it in the binary
- Problem?

Working inside arbitrary restrictions

- Computing the lookup at runtime is unnecessary, takes too long
- Instead, store it in the binary
- Problem? We have to fit it into the 10 KiB limit
- Fitting 9113 values into 10240 bytes $\Rightarrow$ Compress data
- How? Just look at it.

Working inside arbitrary restrictions

```
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,71,71,71,71,71,71,71,71,71,71,71,71,71,71,
71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,71,72,72,72,72,72,72,72,72,
72,72,72,72,72,72,72,72,72,72,72,72,72,72,73,73,73,73,73,73,73,73,73,73,73,73,73,73,73,73,73,73,
74,74,74,74,74,74,74,74,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,
76,76,76,76,76,76,77,77,77,77,77,77,77,77,77,77,78,78,79,79,79,79,79,79,79,79,79,79,79,79,79,79,
79,79,80,80,80,80,80,80,81,81,81,81,81,81,82,82,82,82,82,82,82,82,83,83,83,83,84,84,85,85,85,85,
85,85,85,85,85,85,86,86,86,86,86,86,86,86,86,86,86,86,86,86,87,87,87,87,88,88,90,90,90,90,91,91,
91,91,91,91,91,91,91,91,91,91,91,91,92,92,92,92,92,92,94,94,94,94,94,94,94,94,94,94,95,95,96,96,
96,96,98,98,98,98,98,98,99,99,99,99,99,99,99,99,101,101,101,101,101,101,103,103,103,103,103,103,
105,105,105,105,107,107,107,107,107,107,108,108,108,108,108,108,108,108,110,110,110,110,113,113,
113,113,113,113,113,113,115,115,115,115,115,115,115,115,115,115,115,117,117,119,119,119,119,119,
119,119,119,119,122,122,124,124,124,124,124,124,127,127,127,127,129,129,129,129,129,129,132,132,
132,132,132,132,132,132,135,135,135,135,138,138,141,141,141,141,144,144,144,144,144,144,144,144,
144,144,144,144,148,148,148,148,148,148,148,148,151,151,151,151,154,154,154,154,154,154,154,154,
158,158,158,158,162,162,162,162,162,162,166,166,166,166,166,166,166,166,166,166,166,166,70,70,
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,
70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70, ...
```

The glorious C preprocessor

- Lots of numbers repeat!
- Length of repetitions only increases later on
- Compact encoding using the C preprocessor
- `R4(7)` expands to `7,7,7,7`

```
#define R2(x) x,x
#define R3(x) x,x,x
#define R4(x) R3(x),x
#define R6(x) R4(x),x,x
#define R8(x) R6(x),x,x
...
#define R44(x) R42(x),x,x
#define R52(x) R44(x),R8(x)
#define R104(x) R52(x),R52(x)
#define R208(x) R104(x),R104(x)
```

The glorious C preprocessor

```
uint8_t iq_lookup[9113] = {R104(70),R10(70),R38(71),R22(72),R18(73),R8(74),
  R24(75),R6(76),R10(77),R2(78),R16(79),R6(80),R6(81),R8(82),R4(83),R2(84),
  R10(85),R14(86),R4(87),R2(88),R4(90),R14(91),R6(92),R10(94),R2(95),R4(96),
  R6(98),R8(99),R6(101),R6(103),R4(105),R6(107),R8(108),R4(110),R8(113),
  R10(115),R2(117),R10(119),R2(122),R6(124),R4(127),R6(129),R8(132), ... };
```

- ∼4800 bytes of source code
- Generated by a small Python script
  - Also optimises the `#define`s at the beginning

OpenMP ($\times 0.0123$)

```
void traverse_rec(tree* node) {
    if (node != NULL) {
        node->IQ = iq_lookup[node->data];
        genius[node->id] = node->IQ;

        #pragma omp task
        traverse_rec(node->left );

        traverse_rec(node->right);
    }
}
```

- Could be optimised further, but has no chance of beating the previous version
  - Creating a thread alone takes quadruple of its total time

Questions?

For later, `noclowns@nicze.de` or just talk to me.