Optimization

Monday, November 9, 2020 8:05 AM

Optimization

Goal Make program faster + more efficient in time, space

Targeted optimizations to alleviate bottleneck can improve perf

Checklist

1. Seldom-used process → write simplest code
2. Often-used/big inputs → Make big-O cost reasonable
3. Allow compiler to optimize further
4. Optimize explicitly as last resort

GCC Optimization

```
gcc -O0 // literal C translation
gcc -O2 // nearly all reasonable optimization
```

Constant folding Precalculate constants at compile time

Common sub-expression elimination Save value and reuse it multiple times
(also done @ -O0)

Dead code elimination Removes code that doesn't serve a purpose

Strength reduction Modifies costly instructions
- divide → multiply
- multiply → add/shift
- mod → and

Code motion Pull repeated calculations out of a loop

Tail recursion — Identifies recursive patterns and converts them to iteration

Loop unrolling — Do n iterations of loop work per iteration, reduces test/jump overhead

Caching

Mem. access bottleneck

=> For frequently used data, want to store it in faster memory

Temporal locality - data accessed at same time

Spatial locality - related data used together