Transcript

Introduction to Operating Systems VO

Version v1.0

Daniel Kocher

Daniel.Kocher@stud.sbg.ac.at

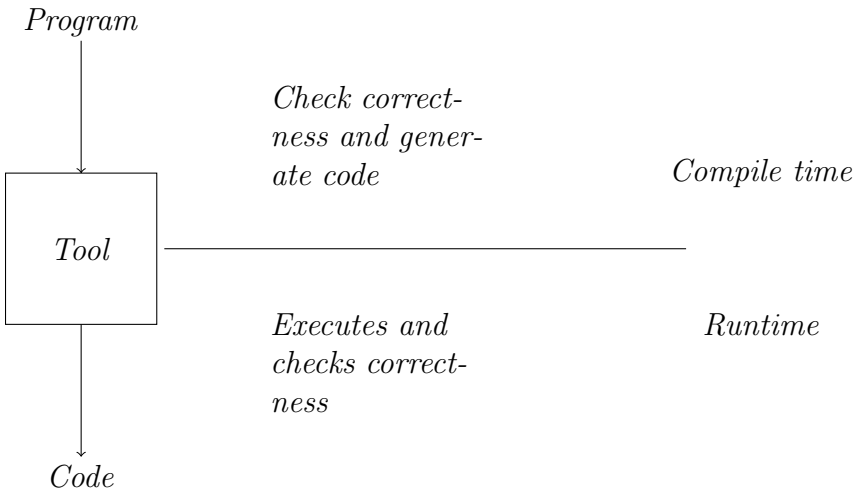
Salzburg, October 5, 2014

Myth Busting

Myth: I prefer programming language X because it makes it more convinient to implement my application.

Answer: Wrong!

The fundamental problem of programming is to establish functional correctness and adequate performance. Different languages provide different tools of automating the process of establishing correctness and performance. The language should be chosen based on that insight.



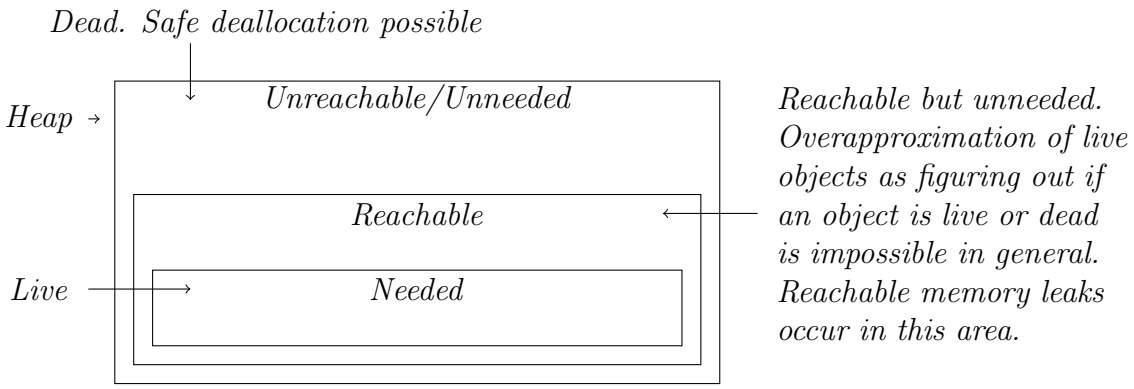
Ultimate goal:
A Compiler checking correctness in such a sense that no exception is thrown while executing in any case. But this is infeasible (mathematical proof possible).

Hardware Exception: Interrupts, a mechanism to stop memory accesses in hardware.

Myth: I like garbage-collected languages like Java because they free me from memory management.

Answer: Wrong!

A garbage collector (GC) provides safe deallocation of unneeded memory but the programmer still needs to say what is unneeded, otherwise the system will run out of memory (memory leak).

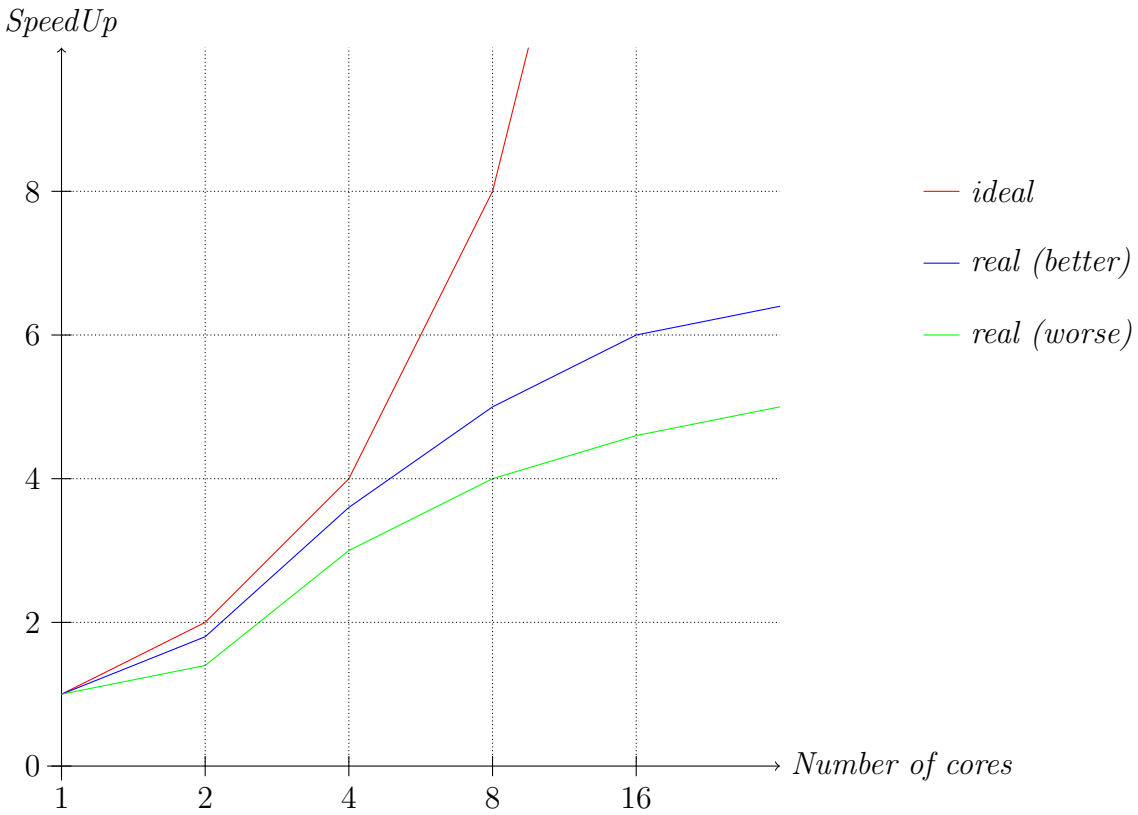


General runtime complexity of a garbage collector: the size of the heap.

Multicore

Amdahl's Law: P represents program parallelism on N cores

- $S(N) = N$ if $P = 100\%$. This is ideal multicore scalability.
- In general: $S(N) = \frac{1}{(1-P) \cdot \frac{P}{N}}$



Sequential vs. Parallelized Code
The bottleneck of parallelized is the memory bus. It is limiting execution even if a problem can be perfectly parallelized without any side effects. Thus a parallelization factor of 100% is infeasible. Any shared resource at any level of an architecture creates a limitation.