Compsys Recap 06/01/25

concurrency and parallel programs

by: TA Jóhann Utne

Agenda

- Processes and threads
- parallel and concurrency
- Mutexes and semaphores
- Scaling
- Exam exercises

Processes, Threads and races

- Processes
  - duplicate of parent, but with own address space (changes are not reflected)
  - Child processes must be reaped, adopted buy init process if termination of parent. Processes that never terminates are “zombie children”
- Threads
  - run in same address as the calling process (changes are reflected)
  - have own thread context: ID, SP, PC, general purpose registers, condition codes
  - can access “critical memory” must be handled with semaphores (mutexes) and / or condition variables
- Races / Deadlocks
  - code depend on order of execution
  - Starvation: some threads do nothing / one thread does everything
  - Deadlock: each thread blocks

Parallel & concurrent

Parallel : >1 or more logical events /flows happening at the same time

Concurrent: 1 > more logical events/flows may happen in any order

Mutexes & semaphores

- Mutexes:
  - lock
  - unlock
- Semaphores:
  - more general “mutex”
  - P - “lock”
  - V - “unlock”

Amdahls and gustafsons Law

Definition (Amdahl's Law)

If p is the proportion of execution time that benefits from parallelisation, then $S(N)$ is maximum theoretical speedup achievable by execution on N threads, and is given by

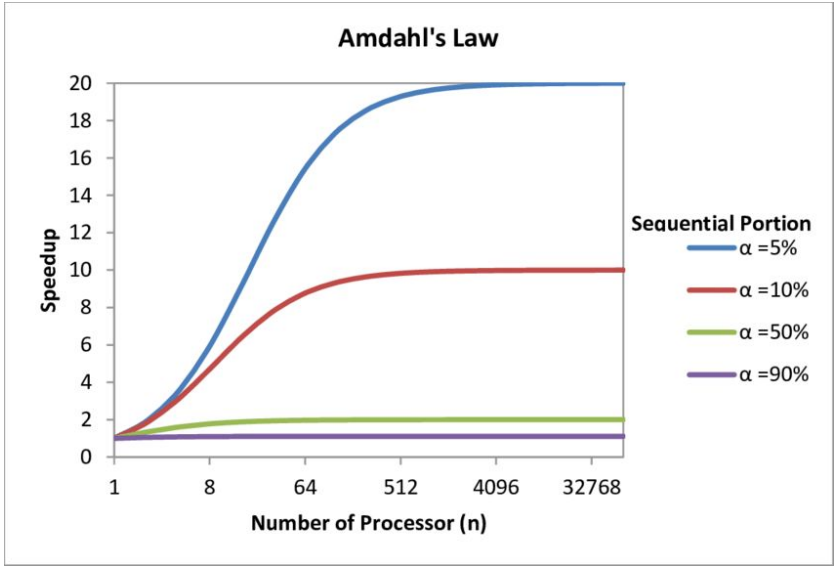
$$S(N) = \frac{1}{(1-p) + \frac{p}{N}}$$

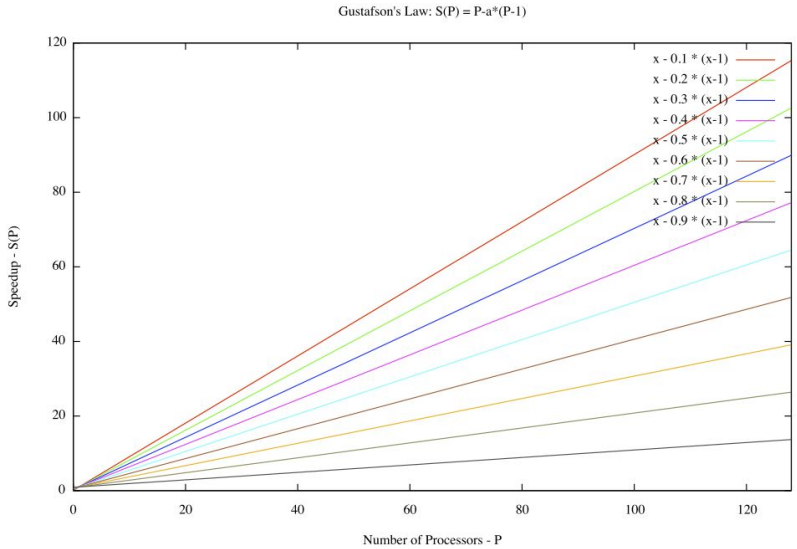
Definition (Gustafson's Law)

If $s = 1 - p$ is the proportion of execution time that must be sequential, then $S(N)$ is maximum theoretical speedup achievable by execution on N threads, and is given by

$$S(N) = N + (1 - N) \times s$$

Graphs of Amdahls (strong scaling) and Gustafsons (weak scaling)





“the rest”

- threadpools
- condition variables
- atomicity

Fork example from exam:

```
#inc
#inc
#inc

voi
    wr
}

int
    wr
    pi
    pi
    fo
    wr
}
```

```
int main () {
  if (fork() == 0) {
    printf("1");
    if (fork() == 0) {
      printf("2");
    }
  } else {
    pid_t pid = fork();
    if (waitpid(pid, NULL, 0) > 0) {
      if (fork() == 0) {
        printf("3");
      } else {
        printf("4");
      }
    }
    printf("5");
  }
}
```

Exam set:

Exam-set 2023-24

if time

Re-Exam-set 2023-24