Fair Share Schedule
Another metric: fair share

- How to make sure P1 gets exactly K-times the execution time of P2

- How to enforce the fair share in a dynamic and modular way
  - Adjustable at run time
  - Alice’s decision shouldn’t affect Bob’s processes’ behavior

\[
\frac{1\text{ms}}{} \rightarrow 10\text{ms}
\]
Naïve solutions

- Adjust the time slice length
  - Problem: can lead to unreasonable slice length

- Adjust the number of process instances in the ready queue
  - Problem: ...
Lottery schedule

- Every process gets $x$ lottery tickets
  - $X$ is determined by the “fair share”
- Scheduler runs lottery; the process holding the winning number runs
- Example
Lottery schedule – ticket exchange rate

• Base tickets
• Regional tickets
• ➔ ensure dynamic and modular management
• Example
Is lottery schedule perfect?

• No!
• Non-deterministic
• Complexity
Before we move on ...

\[ S_{\text{side}} = A^2 \quad B^1 \]

\[ A^{1/2} \quad \frac{1}{2} + \frac{1}{2} = \frac{1}{2} \quad A^{1} = \frac{1}{2} \]

B \quad \underline{B} \quad \underline{B}

A \quad \underline{A}

B
Stride schedule

• Motivation
  • How to make lottery schedule deterministic?

• Algorithm
  • Every process has a stride and a start time
  • Every time a process is scheduled, it progresses its stride
  • The scheduler picks the least-progressed process to run

• Example
Linux Complete Fair Scheduler

• Which process to run?
  • Similar with stride scheduling
  • “Nice” level decides the stride length

• What is the time slice?
  • Vary based on the number of ready processes, etc.

• An efficient scheduler is VERY important these days