# CS 202: Advanced Operating Systems

## Stride Scheduling

# Stride scheduling

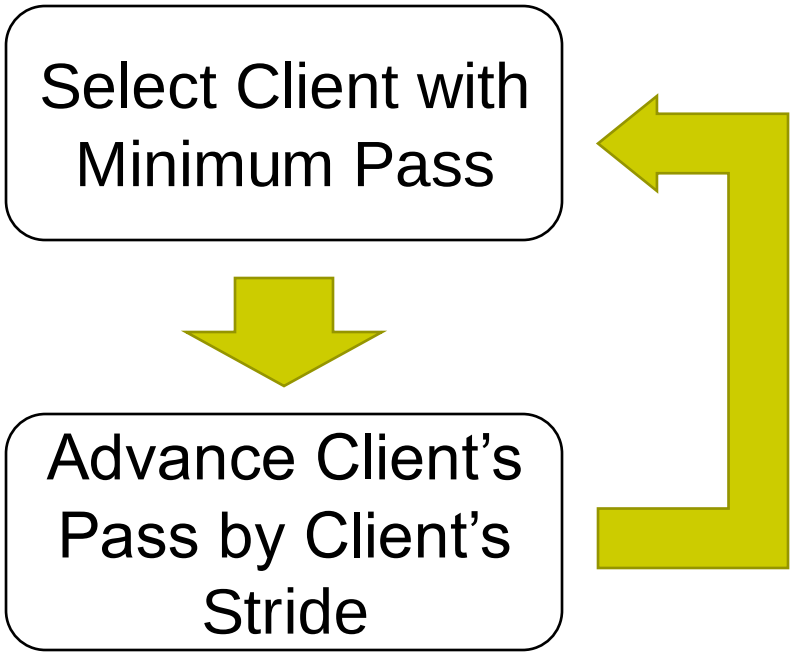
- Deterministic version of lottery scheduling
- Mark time virtually (counting passes)
  - Each process has a stride: number of passes between being scheduled
  - Stride inversely proportional to number of tickets
  - Regular, predictable schedule
- Can also use compensation tickets
- Similar to weighted fair queuing
  - Linux CFS is similar

# Stride Scheduling – Basic Algorithm

Client Variables:

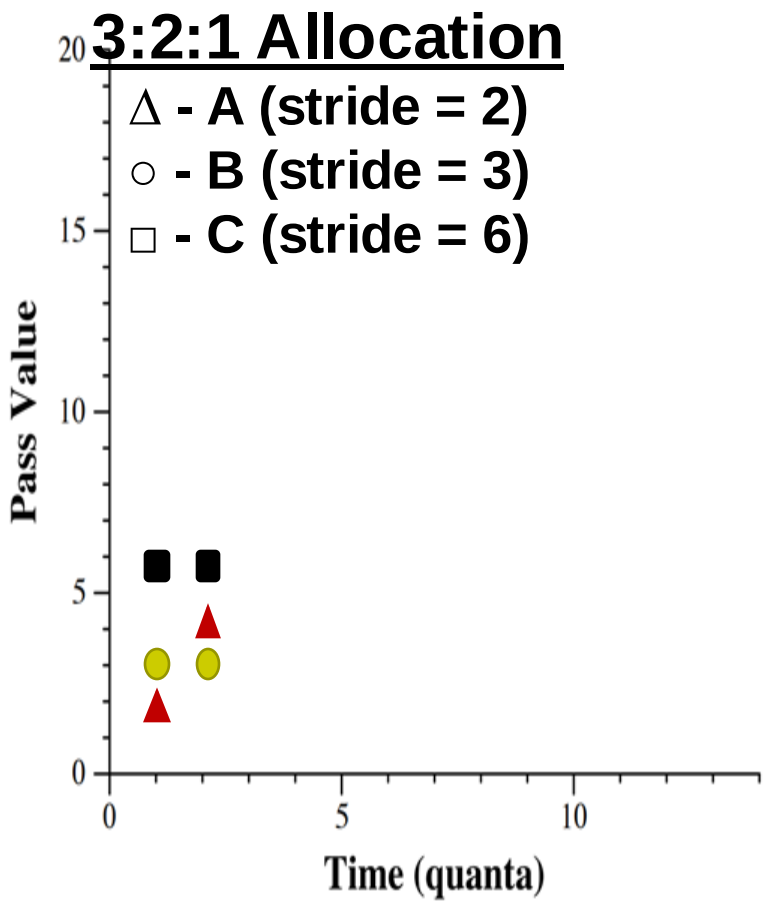
- Tickets
  - Relative resource allocation
- Strides (
  - Interval between selection
- Pass (
  - Virtual index of next selection

- minimum ticket allocation



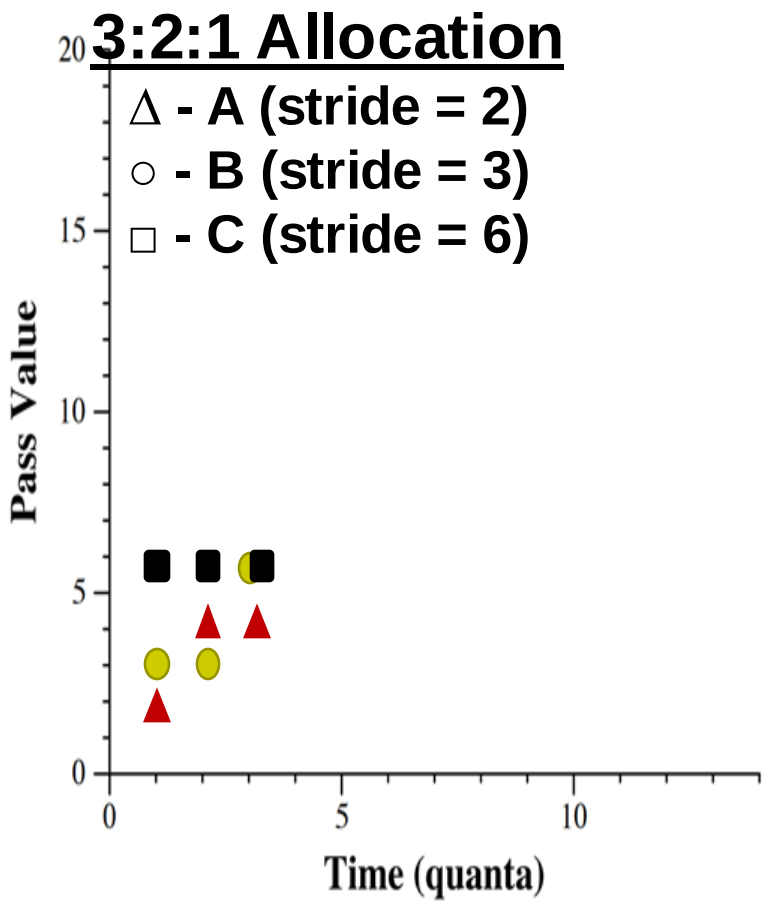

Slide and example from Dong-hyeon Park

# Stride Scheduling – Basic Algorithm



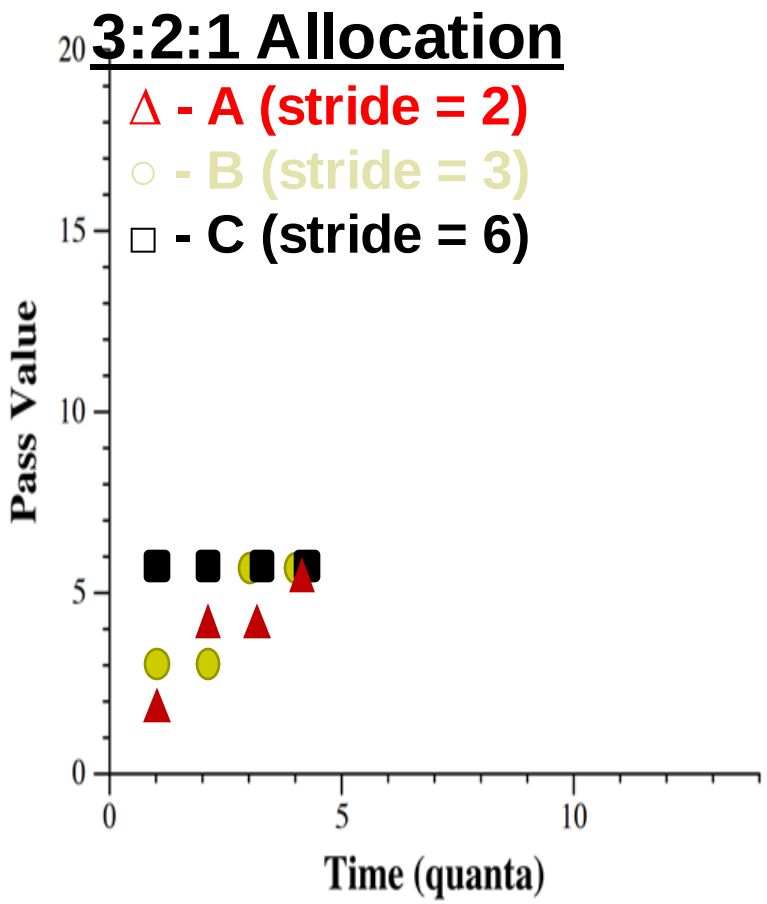

| | ▲ | ● | ■ |
|---|---|---|---|
| **Time 1:** | 2 | 3 | 6 |
| | **+2** | | |
| **Time 2:** | 4 | 3 | 6 |

# Stride Scheduling – Basic Algorithm



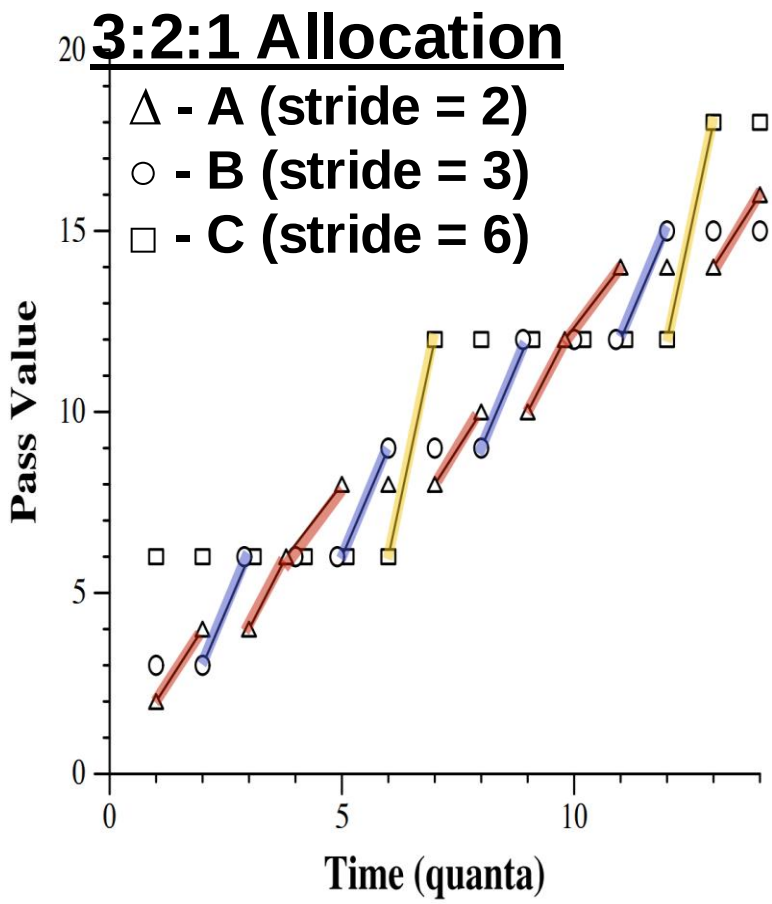

| | ▲ | ● | ■ |
|---|---|---|---|
| **Time 1:** | 2 | 3 | 6 |
| | **+2** | | |
| **Time 2:** | 4 | 3 | 6 |
| | | **+3** | |
| **Time 3:** | 4 | 6 | 6 |

# Stride Scheduling – Basic Algorithm




| | ▲ | ● | ■ |
|---|---|---|---|
| **Time 1:** | 2 | 3 | 6 |
| | **+2** | | |
| **Time 2:** | 4 | 3 | 6 |
| | | **+3** | |
| **Time 3:** | 4 | 6 | 6 |
| | **+2** | | |
| **Time 4:** | 6 | 6 | 6 |

# Stride Scheduling – Basic Algorithm




| | ▲ | ● | ■ |
|---|---|---|---|
| **Time 1:** | 2 | 3 | 6 |
| | **+2** | | |
| **Time 2:** | 4 | 3 | 6 |
| | | **+3** | |
| **Time 3:** | 4 | 6 | 6 |
| | **+2** | | |
| **Time 4:** | 6 | 6 | 6 |

⋮

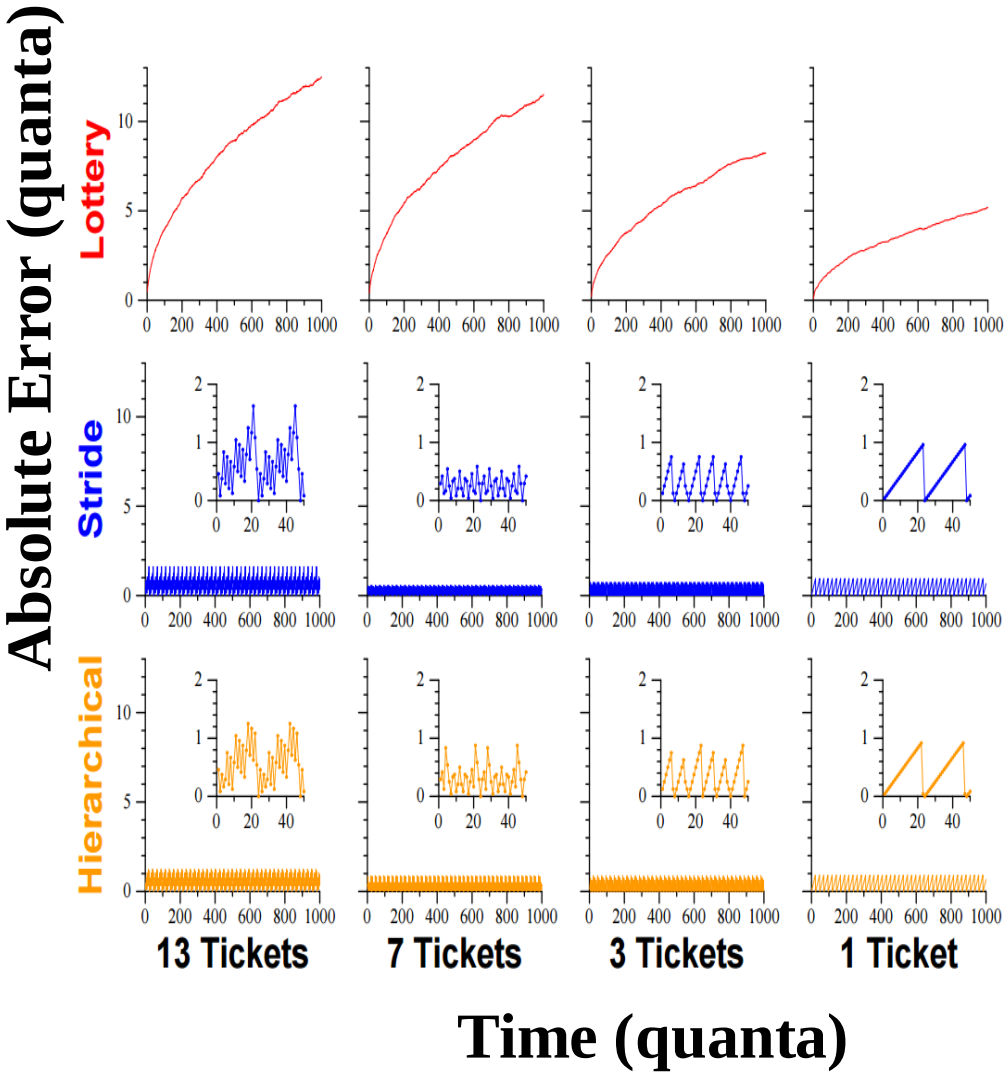
# Dynamic Client Participation

- A key extension
  - global_tickets: total ticket sum for all active clients
  - global_pass: advances at the rate of global_stride per quantum, where global_stride = stride1/global_tickets

# Hierarchical Stride Scheduling

- Consider a set of 101 clients with 100:1:…:1 ticket allocation
  - The 100-ticket client receives 100 quanta before any other clients receiving a single quantum
  - It results a large absolute error of 50
- Solution
  - Treat a group of clients as one client, and then apply the same algorithm within the group
  - Two brilliant ideas:
    - Virtualization
    - Recursion

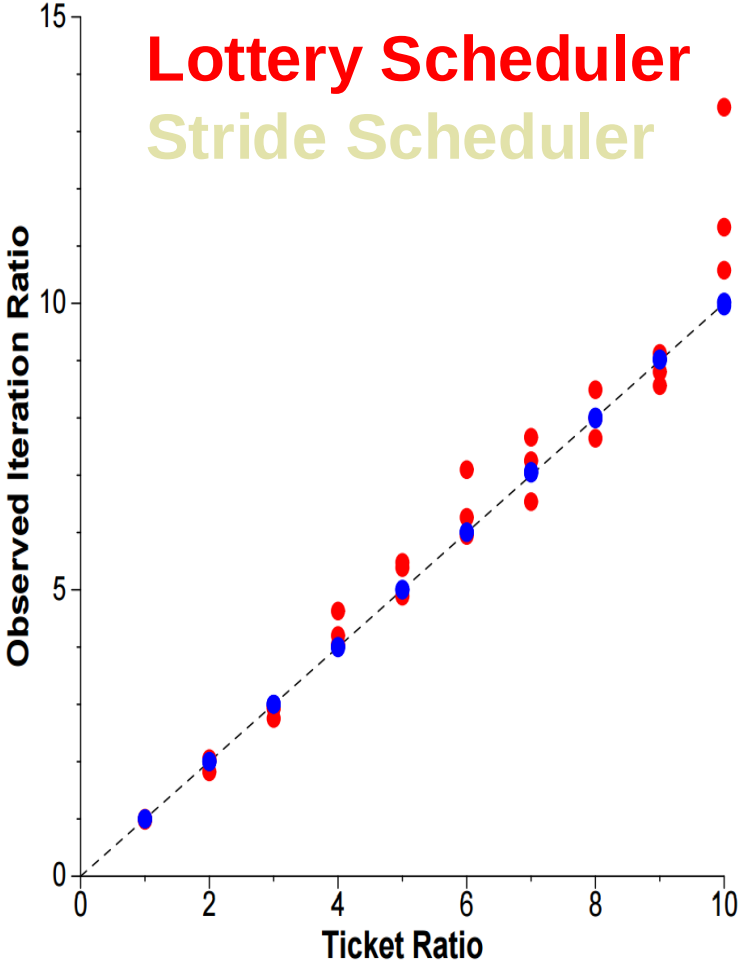
# Throughput Error Comparison




Error is independent of the allocation time in stride scheduling

Hierarchical stride scheduling has more balance distribution of error between clients.

# Accuracy of Prototype Implementation




- Lottery and Stride Scheduler implemented on real-system.
- Stride scheduler stayed within 1% of ideal ratio.
- Low system overhead relative to standard Linux scheduler.

# Linux scheduler

- Went through several iterations
- Currently CFS
  - Fair scheduler, like stride scheduling
  - Supersedes O(1) scheduler: emphasis on constant time scheduling –why?
  - CFS is O(log(N)) because of red-black tree
  - Is it really fair?
- What to do with multi-core scheduling?