The Three Questions

- What is the problem?
- What is new or different?
- What are the contributions and limitations?
Motivation

Scheduling of scarce computer resources (e.g., CPU)

- Has major impact on throughput and response time
- Should be fair (scientific applications)
- But also needs to adjust rapidly (interactive applications)

Priority-based schemes

- Rely on ad-hoc assignment of priorities
- Are poorly understood
- Do not provide encapsulation, modularity
A Probabilistic Solution: Lottery Scheduling
Lottery Scheduling

- Provides a randomized mechanism
  - Not suitable for (hard) real-time systems
- Provides control over relative execution rates
- Can be implemented efficiently
- Supports modular resource management
The Basic Ingredients

- Tickets
  - Abstract, relative, and uniform resource rights
- Lotteries
  - Probabilistically fair selection of next resource holder
  - Throughput proportional to client's ticket allocation
    - Binominal distribution, accuracy improves with \( \sqrt{n} \)
    - Average response time inversely proportional to client's ticket allocation
Fun with Lottery Tickets

- **Ticket transfers**
  - Useful for RPC-based systems
  - Avoid priority inversion problem
- **Ticket inflation**
  - Provides alternative to transfers (no communication!)
  - Needs to be avoided/contained in general
- **Ticket currencies**
  - Support flexible naming, sharing, and protecting of resource rights
- **Compensation tickets**
  - Make up for underutilization
Lottery scheduling is probabilistic

- Mean error slows down, but still grows
- Fairly high distribution of response times
A Deterministic Solution: Stride Scheduling
The Basic Ingredients

- Tickets
  - Abstract, relative, and uniform resource rights
- Strides (stride = stride1 / tickets)
  - Intervals between selections
- Passes (pass += stride)
  - Virtual time index for next selection
    - Clients with smallest pass gets selected
Making Strides

3:2:1 Allocation

Figure 4

Making Strides

3:2:1 Allocation

Figure 4
Dynamic Client Participation

- Need to maintain aggregate information
  - Global tickets: sum of all active clients’ tickets
  - Global stride: stride1 / global tickets
  - Global pass: incremented by global stride per quantum
- Need to maintain additional per-client information
  - Remain: client’s pass - global pass
    - Added back in when rejoining the system
Figure 4: Allocation Change. Modifying a client’s allocation from $tickets$ to $tickets'$ requires only a constant-time recomputation of its $stride$ and $pass$. The new $stride'$ is inversely proportional to $tickets'$. The new $pass'$ is determined by scaling $remain$, the remaining portion of the the current $stride$, by $stride' / stride$. 
One More Thing

- Throughput error
  - Bounded to single quantum between any two clients
  - But absolute error may still be $O(\# \text{ clients})$
- Approach: combine clients into groups
  - Larger ticket allocations, smaller strides
- Algorithm: balanced binary tree of groups
  - Aggregate ticket, stride, and pass values
  - Updates propagated to each of a client’s ancestors
Simulations
Lotteries v Strides

Lotteries

Strides

Cumulative Quanta over Time
Throughput Accuracy

Figure 9: Throughput Accuracy. Simulation results for two clients with 7:3 (top) and 19:1 (bottom) tick ratios over 1000 allocations. Only the first 100 quanta are shown for the stride scheduler, since its quantization error is deterministic.

(a) Mean lottery scheduler error, averaged over 1000 separate 7:3 runs.
(b) Stride scheduler error for a single 7:3 run.
(c) Mean lottery scheduler error, averaged over 1000 separate 19:1 runs.
(d) Stride scheduler error for a single 19:1 run.
More Throughput Accuracy

Figure 10: Throughput Accuracy – Dynamic Allocations.

Simulation results for two clients with [2,12]:3 (top) and 190:[5,15] (bottom) tick rates over 1000 allocations. The notation [2,12] indicates a random tick allocation that is uniformly distributed from 2 to 12. Random tick allocation was dynamically updated every other quantum.

(a) Mean lottery scheduler error, averaged over 1000 separate [2,12]:3 runs.
(b) Stride scheduler error for a single [2,12]:3 run.
(c) Mean lottery scheduler error, averaged over 1000 separate 190:[5,15] runs.
(d) Stride scheduler error for a single 190:[5,15] run.
Response Time Distribution

(a) Lottery - 19
(b) Stride - 19
(c) Lottery - 1
(d) Stride - 1

Figure 12: Simulation results for two clients with a 19:1 ticket ratio over one million allocations.

15
Hierarchical Strides

The figure shows a comparison between hierarchical and flat scheduling approaches. The graph on the left represents hierarchical scheduling, while the graph on the right represents flat scheduling. The x-axis represents the number of clients, and the y-axis represents the frequency (in thousands) of certain events.

- **Hierarchical Scheduling**
  - Shows a higher frequency of events as the number of clients increases.
  - This is expected due to the hierarchical nature of the scheduling algorithm, where higher priorities are assigned to clients in a structured manner.

- **Flat Scheduling**
  - Shows a lower frequency of events as the number of clients increases.
  - This indicates that the flat scheduling approach does not handle increased load as efficiently as the hierarchical approach.

The hierarchical scheduling approach enables tighter response times and handles a higher number of clients more effectively compared to the flat scheduling approach.
In the Real World
Figure 14: **CPU Rate Accuracy.** For each allocation ratio, the observed iteration ratio is plotted for each of three 30 second runs. The gray line indicates the ideal where the two ratios are identical. The observed ratios are within 1% of the ideal for all data points.

Figure 15: **CPU Fairness Over Time.** Two processes executing the compute-bound arith benchmark with a 3:1 ticket allocation. Averaged over the entire run, the two processes executed 2409.18 and 802.89 iterations/sec., for an actual ratio of 3.001:1.
Network Scheduling

![Graph showing network scheduling]

Figure 16: **Ethernet UDP Rate Accuracy.** For each allocation ratio, the observed data transmission ratio is plotted for each of three runs. The gray line indicates the ideal where the two ratios are identical. The observed ratios are within 5% of the ideal for all data points.
What Do You Think?