A. Traditional Scheduling Approach

- multi-level feedback queue algorithm – hasn’t changed in 30 years.
- Separate CPU and IO intensive jobs
- Priority based.
- Breaks down for mixed CPU and IO intensive jobs, like video applications, security enabled web servers, databases etc.
- Using real time priority leads to starvation and live locks.
- Behavior can be hard to predict
- deadlocks, live locks or priority inversion may occur.
- Poor adaptation for adaptive time-sensitive workloads.

B. O(1) scheduler

C. Pure Fairshare Scheduling

- Time based approach opposed to priority.
- No starvation. Overall fairness in the system.
- Better balance between desktop and server performance needs.
- Benefits from recent infrastructural components
  - Fine grained time accounting.
  - High resolution timers.
  - Effective data structures (heaps, red-black trees etc.)

Q: Can we do better? A: Yes, by combining fair sharing with cooperation.

D. Overview of Our Approach: Cooperative Polling

- Have overall fairness.
- Allow cooperation between time sensitive tasks via the kernel:
  - Give preferential treatment to TS tasks within the boundaries of fairness.
  - Facilitates uniform fidelity across tasks.

E. Overview of our implementation

- Virtual time based.
- One new system call: coop_poll() 
- Uses efficient heaps for priority queues.
- Benefits from high resolution one-shot timers & precise time accounting in the kernel.
- We use playback of multiple videos to represent a rich workload of multiple time-sensitive applications.

F. Pure Fairshare vs Cooperative Approach

- Fairshare at finest granularity has 5x latency of coop, yet context switch rate is 2x worse.
- Cooperative approach leverages application information to context switch in a much more strategic fashion.

G. Coordinated Adaptation

- Frame rate of all 12 videos at overload.
- The videos are able to maintain a uniform quality even at overload.

H. Conclusion

- Coop + fairness:
  - Gives better timeliness (smaller latency) even under overload.
  - Facilitates coordinated adaptation for multiple adaptive tasks.
  - Informed context switching is cache efficient – leading to a better timeliness-throughput balance.