Announcements

1. Problem Set 1 out today, due Thursday, Feb 12 (start of class)

2. Mid-term exam on Thursday, Feb 26 (in class)

3. Callback

Basic Idea

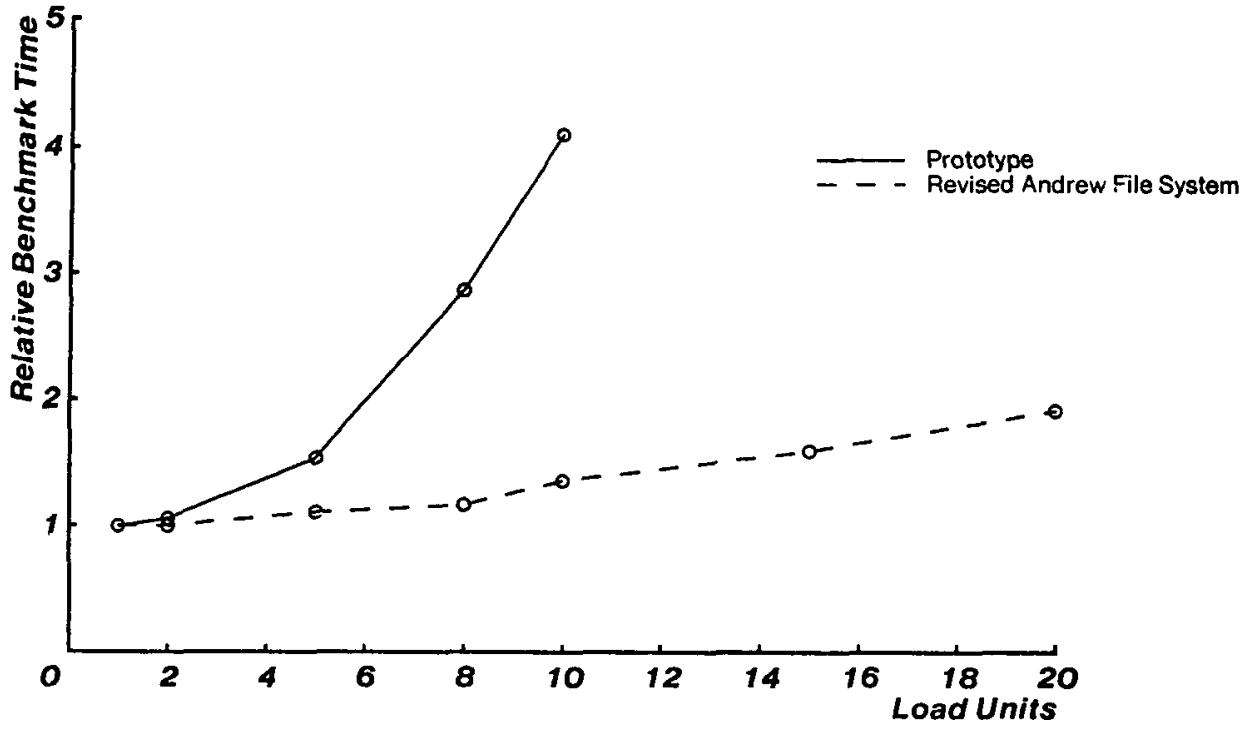
- *targeted notification of caching sites*
- master copy tracks sites with cached data
- typically done at coarse granularity (e.g. entire file)
  can be made to work with byte ranges
- on update, all sites with cached data notified *("callback")*

Original Use

- AFS-2 (circa 1985)

Advantages

- excellent scalability for Unix workloads
- *zero network traffic for read of cached-valid objects*
- precursor to caching for disconnected operation
- biases read performance in favor of write-performance



Disadvantages

- **sizable state on server**
- **complexity of tracking cached state on clients**
- ***silence ambiguous for client***
  - **network failure → lost callbacks**
  - **periodic "keepalive" probes**
  - **data could be stale between probes**
- **NAT networks with masquerading firewalls**

4. Leases

Basic Idea

Caching site obtains finite-duration control from master copy

duration is called "lease period", typically few seconds

multiple sites can obtain read lease; only one can get write lease

- leases have to be renewed, else control expires with lease

Limiting cases

- lease duration = 0 → Check on Use
- lease duration = ∞ → callback (Targeted Notification)

Original Use

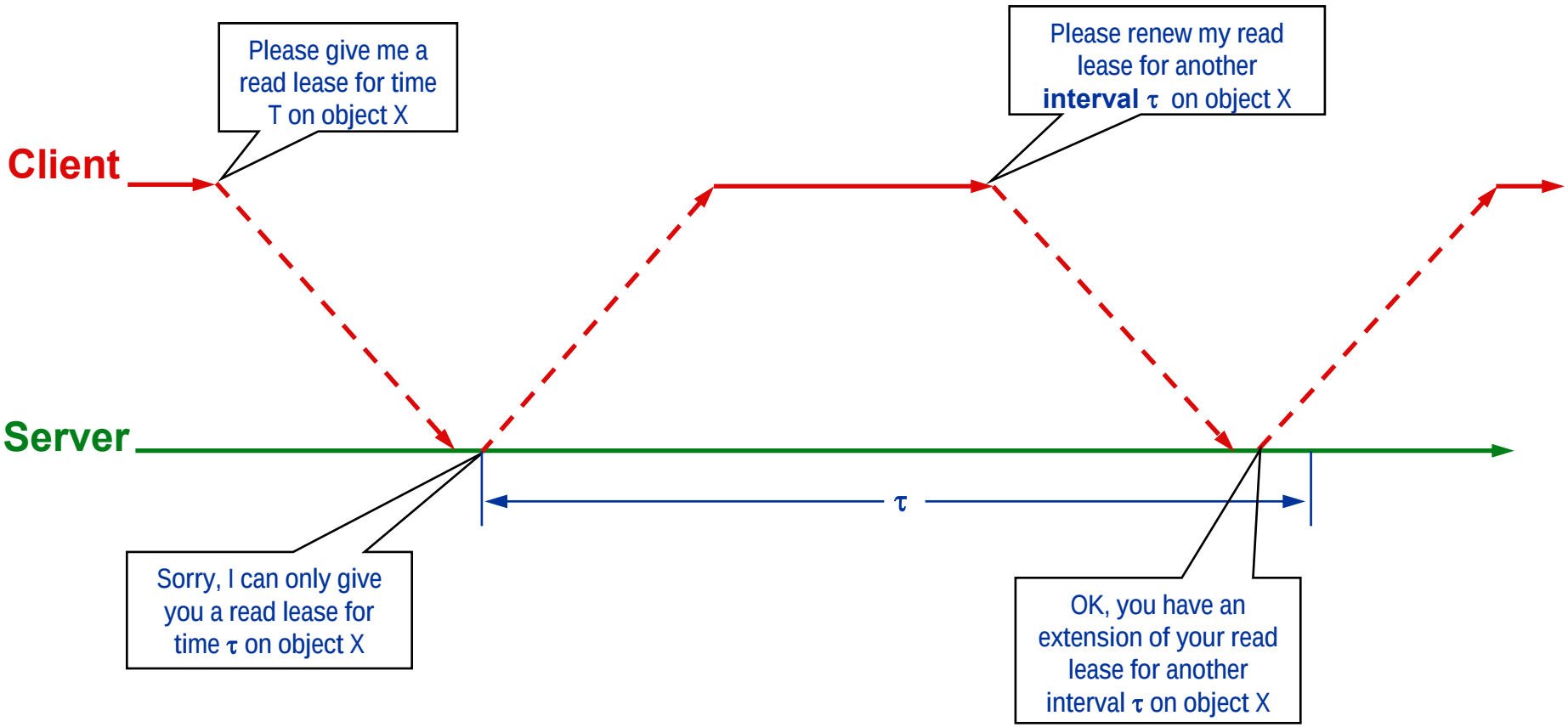
- V System (circa 1985) [*Gray89*]

Advantages

- generalizes the check on use and callback schemes
- lease duration can be tuned to adapt to mutation rate
  Lease duration is a clean tuning knob for design flexibility
- conceptually simple yet flexible
- *time becomes a hidden communication channel*

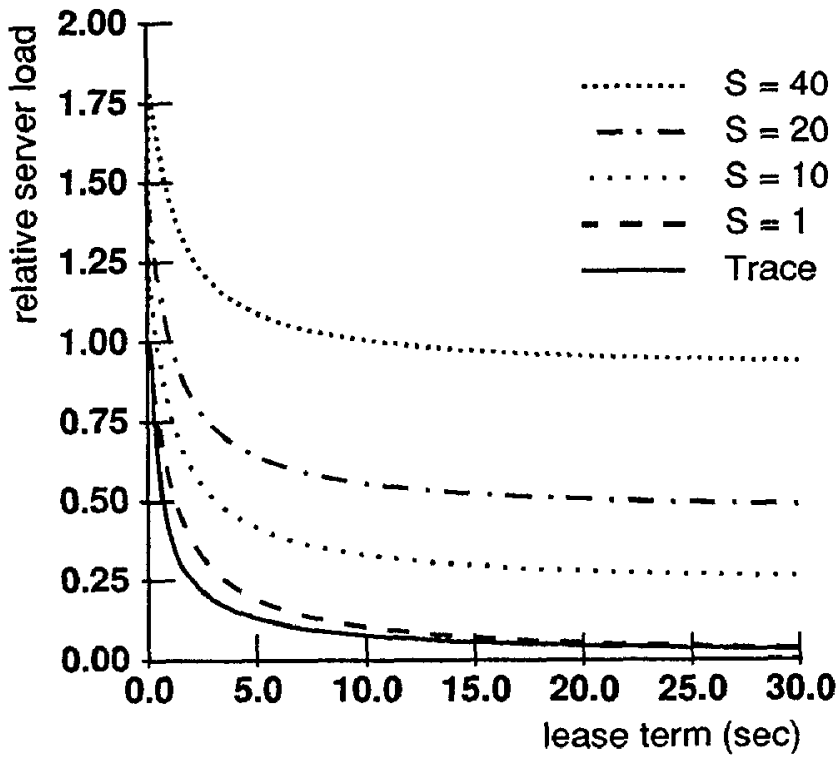
Lease Renewal

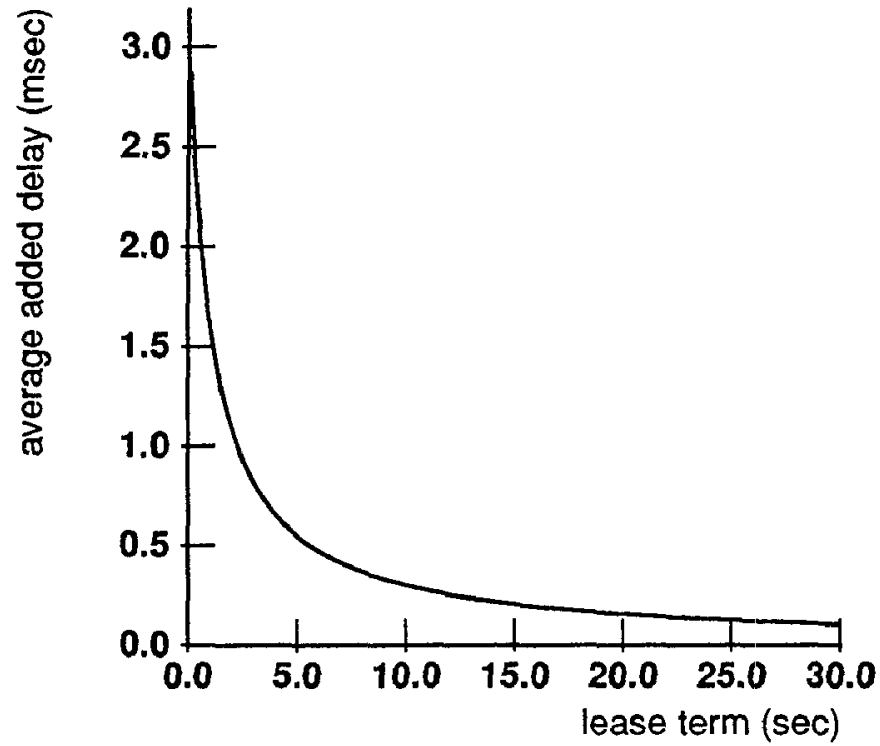
At most one write lease or multiple read leases at any one time



Disadvantages

- lease-holder has total autonomy during lease; revocation?
- writers delayed while read lease holders complete their leases
- more traffic than callback (but less than check on use)
  keepalives for callback only one per server, not per lease





5. Skip Scary Parts

Basic Idea

- ***When write-sharing detected, turn off caching everywhere***
  - All references go directly master copy
- **Resume caching when write-sharing ends**

Original Use

- ***Sprite (circa 1987)*** **(in conjunction with check on use)**

Advantages

- **Precise single-copy semantics (even at byte-level consistency)**
- **Excellent fallback position**
  - Exemplifies good engineering: "Handle average case well; worst case safely"
  - Coda protocol achieves this differently: treats write-sharing as exception (conflict)
- **Good adaptation of caching aggressiveness to workload**

Disadvantages

- **Server maintains state**
- ***Server aware of every use of data (open)***

6. Faith-Based Caching

Basic Idea

- *Blindly assume cached data is valid for a while*
- Periodically check (based on time since last check)
- No communication needed during trust period

Original use

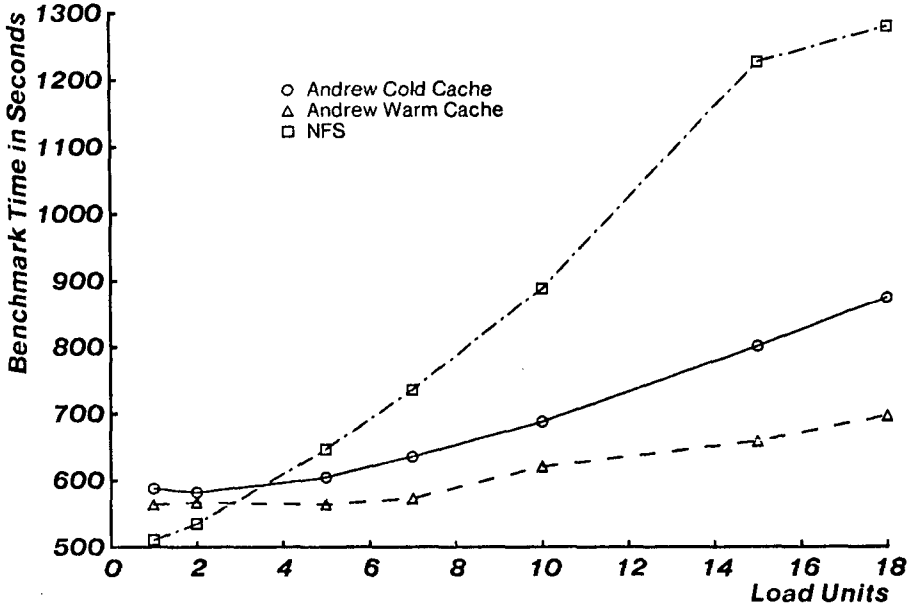
- *Sun NFSv3* file system
  cached file blocks assumed current for X seconds
  X = 3 for files, 30 for directories
- Small variant is a *TTL field* for each object
  used in web caching, gives content creator modicum of control

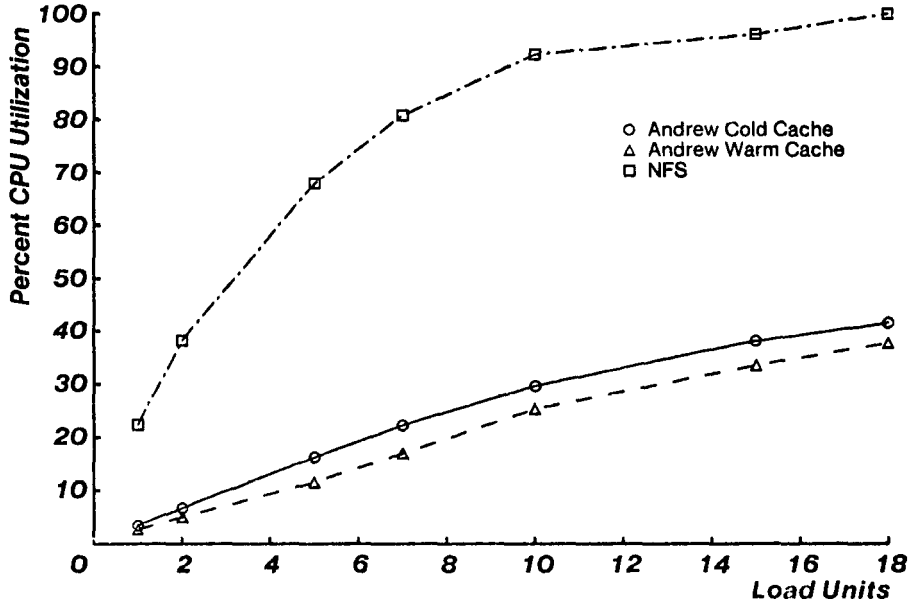
Advantages

- Simple implementation
- *Server is stateless*

Disadvantages

- User-visible inconsistencies sometimes seen (make)
- Blind faith sometimes misplaced!
- Not as efficient as callback-based schemes





7. Pass the Buck

Basic Idea

- *Let the user trigger cache revalidation* (hit "reload")
- Otherwise, all cached copies assumed valid forever
- Equivalent to infinite-TTL faith-based caching

Original use

- Arose in the context of the Web

Advantages

- Trivial to implement, no server changes
- Avoids frivolous cache maintenance traffic

Disadvantages

- Places burden on user
- User may be clueless about level of consistency needed
- Assumes existence of user: pain for write scripts/programs

Cache Consistency Strategies

1. *Broadcast invalidations*
2. *Check on Use*
3. *Callbacks*
4. *Leases*
5. *Skip Scary Parts*
6. *Faith-based Caching*
7. *Pass the Buck*

Many minor variants over the years, but these have withstood the test of time