
Distributed Video Systems
Chapter 7
Parallel Video Servers
Part 2 - A Push-Based Parallel Video Server

Jack Yiu-bun Lee
Department of Information Engineering
The Chinese University of Hong Kong

Contents

Jack Y.B. Lee

- 7.1 Introduction
- 7.2 Inter-Server Scheduling
- 7.3 Performance Modeling and Analysis
- 7.4 Asynchronous Grouped-Sweeping Scheme
- 7.5 Sub-Schedule Striping
- 7.6 Performance Evaluation

7.1 Introduction

Jack Y.B. Lee

- System Architecture
 - ◆ Video Distribution Architecture
 - Proxy-At-Client
 - ◆ Server Striping Policy
 - Space Striping
 - ◆ Video Delivery Protocol
 - Server Push
- Design Challenges
 - ◆ Co-ordination of server transmissions
 - ◆ Video playback continuity
 - ◆ Buffer requirement
 - ◆ Scalability

7.2 Inter-Server Scheduling

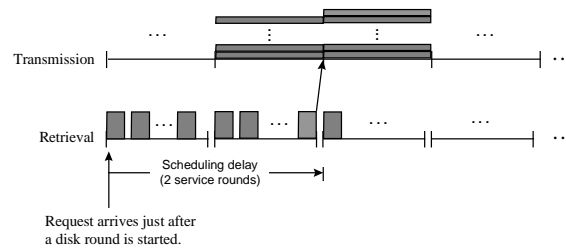
Jack Y.B. Lee

- Problem
 - ◆ Centralized scheduling cannot be done because the servers are independent and connected by a network only.
- Key to Perform Scheduling
 - ◆ Knowledge of a global time or clock!
- Solution
 - ◆ Make use of a distributed clock-synchronization algorithm such as NTP [Mills 1991] to partially synchronize the server clocks.
 - ◆ Perform scheduling locally and independently at each server according to the local clock.

7.2 Inter-Server Scheduling

Jack Y.B. Lee

- Concurrent-Push Algorithm
 - ♦ All servers transmit video data continuously to a video client concurrently.
 - ♦ Let video playback bit-rate be R_V , and there are N_S servers. Then the per-server transmission rate would be R_V/N_S to maintain a correct aggregate rate.
 - ♦ Scheduling at a server:



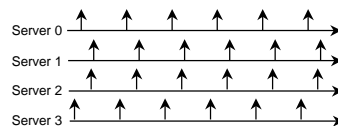
7.2 Inter-Server Scheduling

Jack Y.B. Lee

- Concurrent-Push Algorithm
 - ♦ Transmission from all servers:



- In reality, transmission is most likely done in packets:

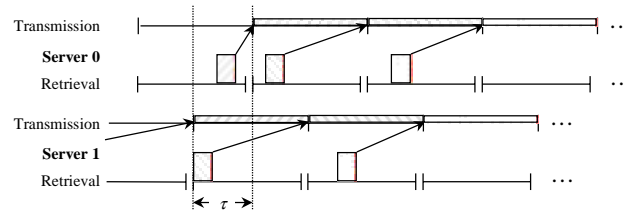


Note that exact synchronization is not possible due to *clock jitters* among servers.

7.2 Inter-Server Scheduling

Jack Y.B. Lee

- Concurrent-Push Algorithm
 - ◆ Server Clock Jitter



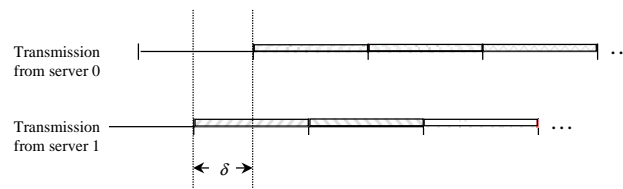
- The amount of clock jitter depends on the clock-synchronization protocol, the network parameters, etc., but is bounded.
- Current protocols can easily synchronize the server clocks to within 100ms on a LAN.

7.2 Inter-Server Scheduling

Jack Y.B. Lee

- Transmission Jitter
 - ◆ Let $T_{i,j}$ be the time server i ($0 \leq i < N_S$) starts transmitting the (jN_S+i) th block of a video stream.
 - ◆ Definition of transmission jitter:

$$\delta = \max\{|T_{i,j} - T_{k,j}| \mid \forall i, k, j\}$$

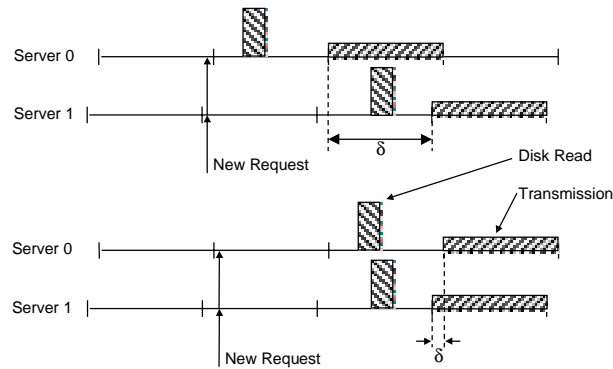


7.2 Inter-Server Scheduling

Jack Y.B. Lee

- Transmission Jitter

- ♦ Looks the same as clock jitter, isn't it?
- ♦ Consider these two cases:



A small clock jitter can lead to a big transmission jitter!

7.2 Inter-Server Scheduling

Jack Y.B. Lee

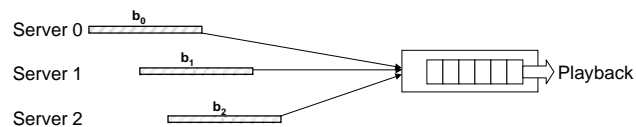
- Transmission Jitter

- ♦ Worst-Case = $\delta \leq T_F$

where $T_F = \frac{N_S Q}{R_V}$ i.e. time to send one video block of Q bytes

- ♦ Problem

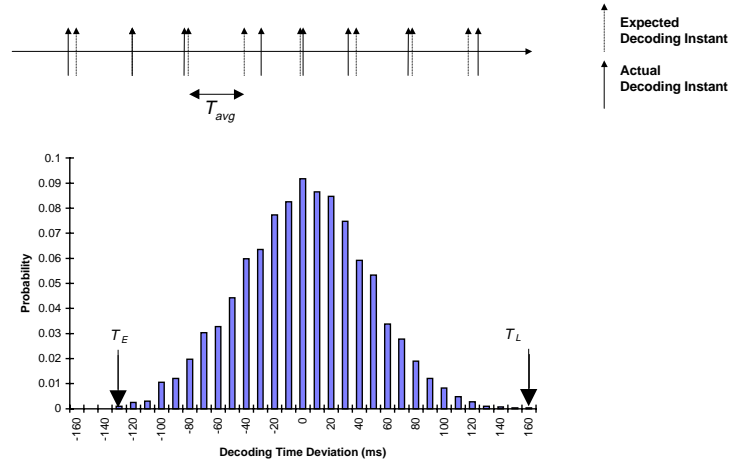
- The bound increase with N_S (no. of servers).
- Transmission jitter affects client buffer requirement.



7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Video Block Consumption Model
 - ◆ Bounded variations with an average rate.



7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Video Block Consumption Model
 - ◆ Bounded variations with an average rate.
 - Decoding-time deviation bounds:
 - Max. lag in decoding: $T_L = \max\{T_{DV}(i) \mid \forall i \geq 0\}$
 - Max. lead in decoding: $T_E = \min\{T_{DV}(i) \mid \forall i \geq 0\}$
 - Peak-to-Peak Decoding-time Deviation:

$$T_{DV} = T_L - T_E$$
 - Time between consumption of any two video blocks i, j is:

$$\underbrace{\max\{(j-i)T_{avg} - T_{DV}, 0\}}_{\text{Min. time interval}} \leq t \leq \underbrace{(j-i)T_{avg} + T_{DV}}_{\text{Max. time interval}}$$

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Client Buffer Requirement
 - ◆ Buffer Management
 - Total $L_C = Y + Z$ buffers, with Y prefilled before playback starts.
 - The L_C buffers are managed as a circular buffer.
 - A client receives video transmissions from N_S servers simultaneously. Hence Y must be multiples of N_S .
 - ◆ Video Block Groups
 - Group n consists of video blocks nN_S to $(n+1)N_S - 1$.
 - ◆ Objective
 - To find the minimum number of buffers Y needed such that video playback continuity can be guaranteed despite *delay and delay jitters, server clock jitters, and decoding-time variations.*
 - To find a similar Z to prevent client buffer overflow.

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Client Buffer Requirement
 - ◆ Buffering for continuity (i.e. underflow)
 - Among the N_S servers, let the earliest transmission for the first round start at time t_0 , then the last transmission for the first round must start at time $t_0 + \delta$.
 - Therefore the time for video block group i to be completely filled, denoted by $F(i)$, is bounded by

$$((i+1)T_F + t_0 + f^-) \leq F(i) \leq ((i+1)T_F + t_0 + \delta + f^+) \quad (10)$$

where f^+ ($f^+ \geq 0$) and f^- ($f^- \leq 0$) are used to model the maximum transmission time deviation due to randomness in the system, including transmission rate deviation, CPU scheduling, bus contention, etc.

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Client Buffer Requirement

- Buffering for continuity (i.e. underflow)

- Assume the client starts playing video after filling the first y groups of buffers (i.e. $Y=yN_S$);
- The playback time for video block group 0 is simply given by $F(y-1)$; and for an arbitrary group i becomes:

$$\{iN_S T_{avg} + F(y-1) + T_E\} \leq P(i) \leq \{iN_S T_{avg} + F(y-1) + T_L\} \quad (11)$$

- For continuity, a video block must arrive before playback:

Worst case: $\underbrace{\max\{F(i)\}}_{(10)} < \underbrace{\min\{P(i)\}}_{(11)}$

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Client Buffer Requirement

- Buffering for continuity (i.e. underflow)

- Substituting and solving gives y as:

$$y > 2 + \frac{f^+ - f^- - T_E}{T_F}$$

- Hence Y can be obtained from

$$Y = \left\lceil 2 + \frac{f^+ - f^- - T_E}{T_F} \right\rceil N_S$$

Note the dependency on N_S

- Buffering to prevent overflow:

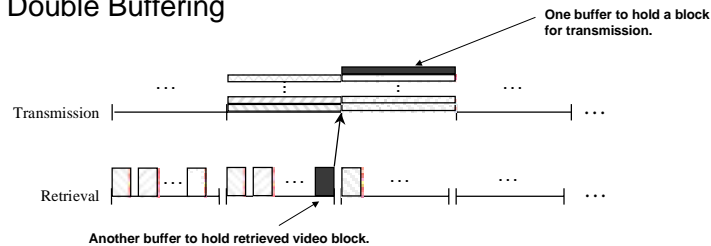
$$Z = \left\lceil 2 + \frac{f^+ - f^- + T_L}{T_F} \right\rceil N_S$$

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Server Buffer Requirement

- ◆ Double Buffering



- ◆ Assume each additional server can increase the system capacity by Λ clients, then the per-server buffer requirement is given by

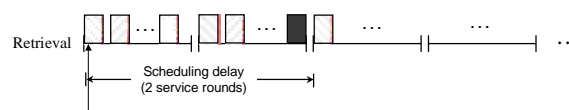
$$B_{server} = 2\Lambda N_s Q$$

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- System Response Time

- ◆ The time from the user requests for a new video session to the time actual video playback starts.
 - ◆ Sums of scheduling delay and prefill delay.
 - ◆ Scheduling delay
 - the time from a client sending a new-session request to the time transmission starts at the server.
 - Worst-case scheduling delay is $D_s = \frac{2N_s Q}{R_v}$



Request arrives just after a disk round is started.

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- System Response Time

- ◆ Prefill delay

- the time from the server starts transmission to the time the first y groups of client buffers are fully filled with data.
 - Worst-case can be determined from (10):

$$D_p = \max\{F(y-1)\} - t_0$$

or

$$D_p = yT_F + \max\{\delta\} + f^+ = (y+1)T_F + f^+$$

$$= \left(3 + \left\lceil \frac{f^+ - f^- - T_E}{T_F} \right\rceil \right) T_F + f^+$$

- Note that $T_F = \frac{N_S Q}{R_V}$

Hence the response time is also proportional to N_S .

7.3 Performance Modeling and Analysis

Jack Y.B. Lee

- Summary of Results

- ◆ Server Buffer Requirement:

$$B_{server} = 2\Lambda N_S Q$$

- ◆ Client Buffer Requirement:

$$Y = \left\lceil 2 + \frac{f^+ - f^- - T_E}{T_F} \right\rceil N_S \quad \text{and} \quad Z = \left\lceil 2 + \frac{f^+ - f^- + T_L}{T_F} \right\rceil N_S$$

- ◆ System Response Time:

$$D_s = \frac{2N_S Q}{R_V} \quad \text{and} \quad D_p = \left(3 + \left\lceil \frac{f^+ - f^- - T_E}{N_S Q R_V} \right\rceil \right) \frac{N_S Q}{R_V} + f^+$$

7.4 Asynchronous Grouped-Sweeping Scheme

Jack Y.B. Lee

- Reducing Server Buffer Requirement
 - ♦ By dividing a service round from serving ΔN_S requests to N_S rounds, each serving only Δ requests.
 - ♦ The idea is same as GSS and the buffer requirement is reduced to

$$B_{server} = QN_S \Delta \left(1 + \frac{1}{N_S} \right)$$

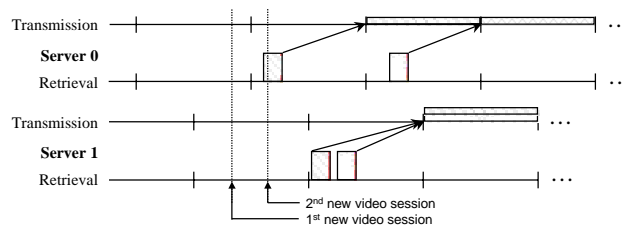
- ♦ Simple huh? Not quite!

7.4 Asynchronous Grouped-Sweeping Scheme

Jack Y.B. Lee

- Inconsistent Group Assignments

- ♦ An Example:



- ♦ The group assignments among servers will become inconsistent and some servers can become overloaded in one group while others are not.

7.4 Asynchronous Grouped-Sweeping Scheme

Jack Y.B. Lee

- Inconsistent Group Assignments

- Solution: Admission Scheduling

- An admission scheduler is used to control the admission of all new video sessions.
- Inconsistent group assignment is prevented by delaying the admission of new sessions by

$$\Omega = \left\lceil \frac{\pi N_s}{T_f} \right\rceil + 1$$

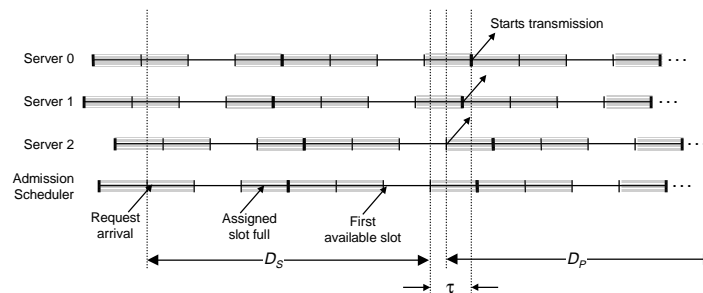
- The idea is to admit a new session to a service round guaranteed to have not yet started in *any* of the servers.

7.4 Asynchronous Grouped-Sweeping Scheme

Jack Y.B. Lee

- Inconsistent Group Assignments

- Solution: Admission Scheduling



- With admission scheduling in place, we can prove that the transmission jitter becomes the same as the clock jitter τ .

7.4 Asynchronous Grouped-Sweeping Scheme

Jack Y.B. Lee

- Inconsistent Group Assignments
 - ◆ Solution: Admission Scheduling
 - Side Benefits
 - Reduced client buffer requirement (δ becomes τ where $\delta \geq \tau$).
 - Reduced prefill delay (same reason).
 - Tradeoff
 - Additional scheduling delay due to the added artificial delay as well as the delay incurred in finding a service round that is not full.
 - The extra scheduling delay depends on the system utilization.

7.5 Sub-Schedule Striping

Jack Y.B. Lee

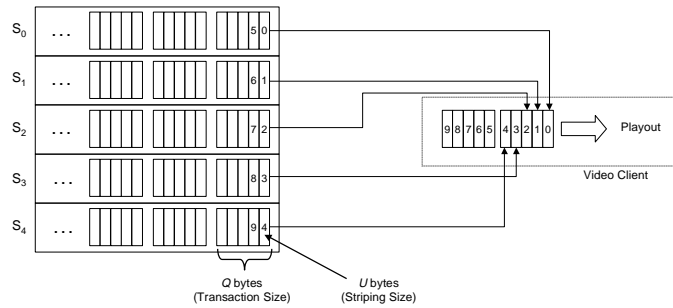
- Motivation
 - ◆ AGSS reduces server buffer requirement substantially but only reduces client buffer requirement slightly.
 - ◆ We can further reduce the client buffer requirement and consequently prefill delay by decoupling striping from disk retrieval.
- Principle
 - ◆ In conventional disk scheduling, each disk transaction retrieves a data block of Q bytes, which contains continuous video data.
 - ◆ It doesn't have to be continuous video data.

7.5 Sub-Schedule Striping

Jack Y.B. Lee

- Principle

- ♦ Striping Size = U bytes
- ♦ Retrieval Size = Q bytes



7.5 Sub-Schedule Striping

Jack Y.B. Lee

- Performance Impact

- ♦ Assuming we maintain $U=Q/N_S$, then:

- Client buffer requirement becomes

$$Y > 1 + \left(\frac{f^+ - f^- - T_E + \tau}{T_{avg}} \right) \quad \text{and} \quad Z > 1 + \left(\frac{f^+ - f^- + T_L + \tau}{T_{avg}} \right)$$

- And prefill delay becomes

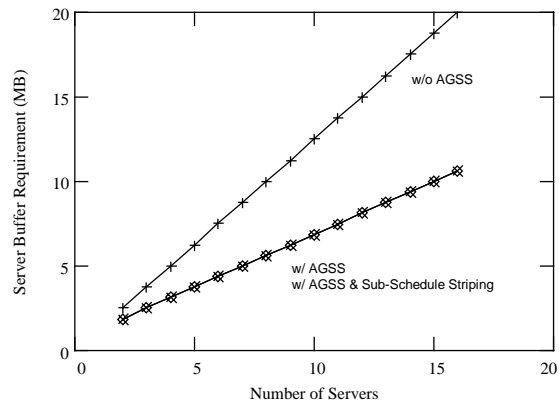
$$D_p = Y T_{avg} + f^+ + \tau$$

- Both are now independent of N_S !
- Any tradeoff?

7.6 Performance Evaluation

Jack Y.B. Lee

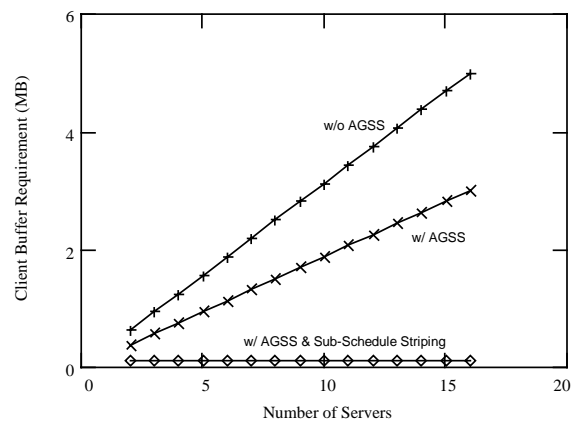
- Server Buffer Requirement



7.6 Performance Evaluation

Jack Y.B. Lee

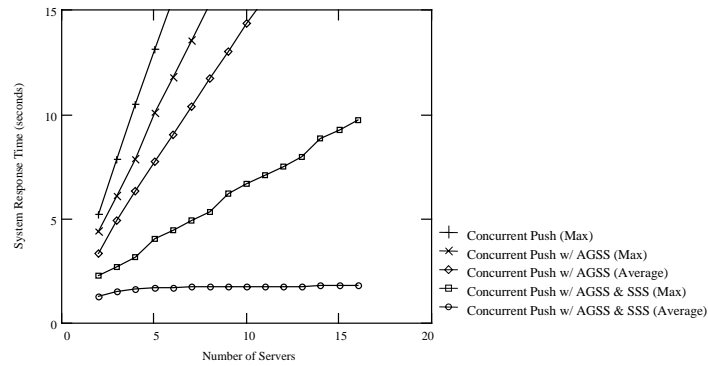
- Client Buffer Requirement



7.6 Performance Evaluation

Jack Y.B. Lee

- System Response Time



7.6 Performance Evaluation

Jack Y.B. Lee

- How scalability is the architecture?
 - ◆ Limited by server memory size:
 - Using servers with 256MB buffer memory, we can scale up to 408 servers, serving 3672 video streams at 90% utilization.
 - Using servers with 1GB buffer memory, we can scale up 14400 video streams with a client-server ratio of 250 (64 servers) at 90% utilization.
 - ◆ Limited by client processing capability:
 - Larger N_S results in smaller striping units.
 - Smaller striping units incurs more processing overhead at the client since resequencing is required.
 - Using 1KB striping units and 64KB transaction size, we can scale to at most 64 servers.