Data Partitioning for Video-on-Demand Services

Shahin Kamali

Joint work with Bairong Lei, Ivan Surya and Khuzaima Daudjee

International Symposium on Network Computing and Applications (IEEE NCA13)

August 22, 2013
Cheriton School of Computer Science
University of Waterloo, Ontario, Canada
Structure

- Video-partitioning problem
Outline

Structure

- Video-partitioning problem
- Algorithms
Outline

Structure

- Video-partitioning problem
- Algorithms
- Theoretical analysis
Outline

Structure

- Video-partitioning problem
- Algorithms
- Theoretical analysis
- Performance evaluation
Videos

- Videos are important ...

YouTube accounts for up to 27 percent of Internet traffic [Sandvine, 2012]. Netflix accounts for about 33 percent of Internet traffic in the USA [Sandvine, 2012].
Videos are important ...

- YouTube accounts for up to 27 percent of Internet traffic [Sandvine, 2012].
Videos are important ... 

- YouTube accounts for up to 27 percent of Internet traffic [Sandvine, 2012].
- Netflix accounts for about 33 percent of Internet traffic in the USA [Sandvine, 2012].
Videos

- Videos are important ...
  - YouTube accounts for up to 27 percent of Internet traffic [Sandvine, 2012].
  - Netflix accounts for about 33 percent of Internet traffic in the USA [Sandvine, 2012].

- How should we store the videos?
  - In a cluster!
Data Partitioning

Server data storage architecture

- A single large server?
  - Holds all video data
  - Services all video-on-demand requests
Server data storage architecture

- A single large server?
  - Holds all video data
  - Services all video-on-demand requests
- Increasing demand outstrips resources
- The whole service will be interrupted if the server fails
Server data storage architecture

- A single large server?
  - Holds all video data
  - Services all video-on-demand requests
- Increasing demand outstrips resources
- The whole service will be interrupted if the server fails
- Distribute videos across multiple servers!
  - Provides scalability and availability for video-on-demand services
Partition a set of videos into $m$ subsets so that each partition is hosted by a server.
Partition a set of videos into \( m \) subsets so that each partition is hosted by a server

Objectives:
- The load on each server is balanced
- The space-capacity of servers is well-used
Some videos are more popular than others
  - There is more demand for them
Some videos are more popular than others
- There is more demand for them
Some videos are more popular than others
- There is more demand for them
- Consider the last week’s view count as the measure of popularity
  - Most videos become popular in the first few weeks
  - The popularity decreases in the rest of videos’ life-span.
Data Partitioning

Load balancing

- Some videos are more popular than others
  - There is more demand for them
- Consider the last week’s view count as the measure of popularity
  - Most videos become popular in the first few weeks
  - The popularity decreases in the rest of videos’ life-span.
- Weight of a video: its video count in the last week.
Here is a bad partitioning:
Data Partitioning

Load balancing (cntd.)

- Balance the load among the servers
  - Minimize the maximum load of the servers
  - Load of a server is the total weight of videos assigned to it
Data Partitioning

Load balancing (cntd.)

- Balance the load among the servers
  - Minimize the maximum load of the servers
  - Load of a server is the total weight of videos assigned to it
- The cost of partitioning: the maximum load of a server

The cost is 10
Load balancing (cntd.)

- Balance the load among the servers
  - Minimize the maximum load of the servers
  - Load of a server is the total weight of videos assigned to it
- The cost of partitioning: the maximum load of a server

The cost is 9
Sorted Round-Robin algorithm

- Determine the number of partitions ($m$)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get $m$ groups of video files
Sorted Round-Robin algorithm

- Determine the number of partitions \( m \)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \( m \) groups of video files

Example:

\(< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >\)
Sorted Round-Robin algorithm

- Determine the number of partitions ($m$)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get $m$ groups of video files

Example:

\[
< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >
\]
Sorted Round-Robin algorithm

- Determine the number of partitions ($m$)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get $m$ groups of video files

Example:

< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >
Algorithms

Sorted Round-Robin algorithm

- Determine the number of partitions ($m$)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get $m$ groups of video files

Example:

$\langle 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 \rangle$
Algorithms

Sorted Round-Robin algorithm

- Determine the number of partitions ($m$)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get $m$ groups of video files

Example:

< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >
Sorted Round-Robin algorithm

- Determine the number of partitions ($m$)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get $m$ groups of video files

Example:

$< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >$
Algorithms

- Sorted Round-Robin algorithm
  - Determine the number of partitions \( (m) \)
  - Sort videos by their weight (past week’s view count) in decreasing order
  - Perform round-robin partitioning to get \( m \) groups of video files

Example:
\[
< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >
\]
Sorted Round-Robin algorithm

- Determine the number of partitions \( m \)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \( m \) groups of video files

Example:
\(< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 1 >\)
Algorithms

Sorted Round-Robin algorithm

- Determine the number of partitions \( m \)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \( m \) groups of video files

Example:

\(< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >\)
Sorted Round-Robin algorithm

- Determine the number of partitions \( m \)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \( m \) groups of video files

Example:

\(< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 1 >\)
Algorithms

- Sorted Round-Robin algorithm
  - Determine the number of partitions ($m$)
  - Sort videos by their weight (past week’s view count) in decreasing order
  - Perform round-robin partitioning to get $m$ groups of video files

Example:

\[
< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 1 >
\]
Sorted Round-Robin algorithm

- Determine the number of partitions ($m$)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get $m$ groups of video files

Example:

$$< 8, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 1 >$$
Sorted Round-Robin algorithm

- Determine the number of partitions \((m)\)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \(m\) groups of video files

Example:

< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >
Sorted Round-Robin algorithm

- Determine the number of partitions \((m)\)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \(m\) groups of video files

Example:

\(< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >\)
Sorted Round-Robin algorithm

- Determine the number of partitions \( m \)
- Sort videos by their weight (past week's view count) in decreasing order
- Perform round-robin partitioning to get \( m \) groups of video files

Example:

\(< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 1 >\)
Sorted Round-Robin algorithm

- Determine the number of partitions \( m \)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \( m \) groups of video files

Example:

\(< 8, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 1 >\)
Algorithms

Sorted Round-Robin algorithm

- Determine the number of partitions \((m)\)
- Sort videos by their weight (past week’s view count) in decreasing order
- Perform round-robin partitioning to get \(m\) groups of video files

Example:

\(< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 1 >\)
Algorithm

- **Sorted Round-Robin algorithm**
  - Determine the number of partitions ($m$)
  - Sort videos by their weight (past week’s view count) in decreasing order
  - Perform round-robin partitioning to get $m$ groups of video files

**Example:**

< 8, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 1 >
**Algorithms**

- **Sorted Round-Robin algorithm**
  - Determine the number of partitions \((m)\)
  - Sort videos by their weight (past week’s view count) in decreasing order
  - Perform round-robin partitioning to get \(m\) groups of video files

**Example:**

\(< 8, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 1 >\)
**Greedy algorithm**

- Determine the number of partitions ($m$)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load

Example:

```
<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>
```
Algorithms

Greedy algorithm

- Determine the number of partitions \((m)\)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  \(\langle 2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4 \rangle\)
Greedy algorithm

- Determine the number of partitions \( m \)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  \[<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>\]
Algorithms

- Greedy algorithm
  - Determine the number of partitions ($m$)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    $$<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>$$

![Server Load Diagram]
Algorithms

- **Greedy algorithm**
  - Determine the number of partitions \( m \)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    \[
    \langle 2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4 \rangle
    \]
Greedy algorithm

- Determine the number of partitions \((m)\)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  \[<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>\]
**Algorithms**

- **Greedy algorithm**
  - Determine the number of partitions \( m \)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    \[ <2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4 > \]
Greedy algorithm

- Determine the number of partitions ($m$)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load

Example:

<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>

Server 1
Server 2
Server 3
Server 4
Server 5
Server 6
Server 7
Server 8

2 4 2 4 2 4
1 4 6 6 6 6 6 6
Greedy algorithm
- Determine the number of partitions ($m$)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  `<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>`

![Server loads diagram](image)
Algorithms

- **Greedy algorithm**
  - Determine the number of partitions \( m \)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    - \(<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>\)

Server 1
Servers' loads
Server 2
Server 3
Server 4
Server 5
Server 6
Server 7
Server 8
\(2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4\)
Algorithms

Greedy algorithm

- Determine the number of partitions \( m \)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  \[ <2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4 > \]
Algorithms

Greedy algorithm
- Determine the number of partitions ($m$)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  $<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>$

![Diagram of server loads]

Server 1: 2, 2
Server 2: 4
Server 3: 2
Server 4: 4
Server 5: 2
Server 6: 4
Server 7: 1, 4
Server 8: 4
Greedy algorithm

- Determine the number of partitions \((m)\)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  \(<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>\)

Data Partitioning for Video-on-Demand Services
Algorithms

- Greedy algorithm
  - Determine the number of partitions \((m)\)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    \(<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>\)
Algorithms

Greedy algorithm
- Determine the number of partitions ($m$)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  $$<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>$$
Greedy algorithm

- Determine the number of partitions ($m$)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load

Example:

\[
\langle 2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4 \rangle
\]
Algorithms

- Greedy algorithm
  - Determine the number of partitions ($m$)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    $<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>$
Algorithms

- Greedy algorithm
  - Determine the number of partitions ($m$)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    $$<2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>$$
Algorithms

Greedy algorithm

- Determine the number of partitions \( m \)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  \[ <2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4> \]
Algorithms

Greedy algorithm

- Determine the number of partitions (m)
- Receive videos one by one in random order
- Assign each video to the partition with minimum load
- Example:
  <2, 4, 2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>

Servers' loads

- Server 1: 4
- Server 2: 6
- Server 3: 8
- Server 4: 8
- Server 5: 10
- Server 6: 6
- Server 7: 5
- Server 8: 6

Data Partitioning for Video-on-Demand Services
Algorithms

- **Greedy algorithm**
  - Determine the number of partitions \((m)\)
  - Receive videos one by one in random order
  - Assign each video to the partition with minimum load
  - Example:
    \(<2, 4, 2, 4, 1, 4, 2, 2, 8, 2, 4, 2, 2, 4>\)

- **Sorted Greedy algorithm**
  - First sort the videos in decreasing order of weight and then apply greedy algorithm.
Theoretical Analysis

Multiprocessor Scheduling problem

- Balancing loads among processing machines!

NP-hard [Garey and Johnson, 1990]

The Greedy algorithm for the partitioning problem with $m$ partitions has an approximation ratio of $2 - \frac{1}{m}$ [Graham, 1972].

The Sorted Greedy algorithm has an improved ratio of $\frac{4}{3} - \frac{1}{3m}$ [Graham, 1972].

Theorem
With respect to load balancing, the Sorted Round-Robin algorithm has an approximation ratio of at most 2, and when the number of servers is sufficiently large, this bound is tight.
Theoretical Analysis

Multiprocessor Scheduling problem

- Balancing loads among processing machines!
- NP-hard [Garey and Johnson, 1990]
Multiprocessor Scheduling problem

- Balancing loads among processing machines!
- NP-hard [Garey and Johnson, 1990]
- The Greedy algorithm for the partitioning problem with $m$ partitions has an approximation ratio of $2 - 1/m$ [Graham, 1972].
Theoretical Analysis

Multiprocessor Scheduling problem

- Balancing loads among processing machines!
- NP-hard [Garey and Johnson, 1990]
- The Greedy algorithm for the partitioning problem with $m$ partitions has an approximation ratio of $2 - 1/m$ [Graham, 1972].
- The Sorted Greedy algorithm has an improved ratio of $4/3 - 1/(3m)$ [Graham, 1972].
Theoretical Analysis

Multiprocessor Scheduling problem

- Balancing loads among processing machines!
- NP-hard [Garey and Johnson, 1990]
- The Greedy algorithm for the partitioning problem with \( m \) partitions has an approximation ratio of \( 2 - \frac{1}{m} \) [Graham, 1972].
- The Sorted Greedy algorithm has an improved ratio of \( \frac{4}{3} - \frac{1}{(3m)} \) [Graham, 1972].

**Theorem**

*With respect to load balancing, the Sorted Round-Robin algorithm has an approximation ratio of at most 2, and when the number of servers is sufficiently large, this bound is tight.*
Theoretical Analysis

Load Balancing

- Approximation ratios for load balancing

<table>
<thead>
<tr>
<th>Greedy</th>
<th>Sorted Greedy</th>
<th>Sorted Round Robin</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2 - \frac{1}{m}$</td>
<td>$\frac{4}{3} - \frac{1}{3m}$</td>
<td>2</td>
</tr>
</tbody>
</table>
Theoretical Analysis

Storage Usage

- It is preferred to have a balanced storage usage
  - Make best use of space capacities of servers
Theoretical Analysis

Storage Usage

- It is preferred to have a balanced storage usage
  - Make best use of space capacities of servers
Theoretical Analysis

Storage Usage

- It is preferred to have a balanced storage usage
  - Make best use of space capacities of servers
- The cost of partitioning: the maximum number of videos in a partition
  - We assume the size of videos is almost the same (e.g., YouTube).
The Sorted Round-Robin is optimum regarding the storage usage.
Theoretical Analysis

Storage Usage (cntd.)

- The Sorted Round-Robin is optimum regarding the storage usage

```
Server 1
Server 2
Server 3
Server 4
Server 5
Server 6
Server 7
Server 8
```

```
8  4  2
4  2  1
4  2  2
4  2  2
4  2  2
4  2  2
4  2  2
4  2  2
```
Theoretical Analysis

Storage Usage (cntd.)

- The Sorted Round-Robin is optimum regarding the storage usage

**Theorem**

*With respect to storage usage, the approximation ratios of both greedy algorithms are at least m (the number of partitions).*
Approximation ratios for load balancing and storage usage balancing

<table>
<thead>
<tr>
<th>Load Balancing</th>
<th>Greedy</th>
<th>Sorted Greedy</th>
<th>Sorted Round Robin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Usage</td>
<td>2 - 1/m</td>
<td>4/3-1/(3m)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>m</td>
<td>m</td>
<td>1</td>
</tr>
</tbody>
</table>
Assume video weights follow Zipfian distribution [Summers et al., 2012]
Assume video weights follows Zipfian distribution [Summers et al., 2012]

- Distribution of view counts (defined by *partitioning skew*)
- Distribution of requests (defined by *request skew*)
Performance Evaluation

Assume video weights follow a Zipfian distribution [Summers et al., 2012]

- Distribution of view counts (defined by partitioning skew)
- Distribution of requests (defined by request skew)

Three identical SunX2100 connected by 1Gbit Ethernet switch

A single client which sends HTTP requests to videos hosted in the three servers

- The requests are generated via httpref.
Load balancing performance

- Partitioning skew is $\theta_p = 0.8$.
- Request skew $\theta_r \in \{0.8, 1.2\}$. 
Performance Evaluation

Load balancing performance

- Partitioning skew is $\theta_p = 0.8$.
- Request skew $\theta_r \in \{0.8, 1.2\}$.

Sorted Round-Robin algorithm performs almost as well the Sorted Greedy algorithm

Data Partitioning for Video-on-Demand Services
Performance Evaluation

- Storage usage performance
  - Partitioning skew is $\theta_p = 0.8$. 

Data Partitioning for Video-on-Demand Services
• Storage usage performance

  • Partitioning skew is $\theta_p = 0.8$.

• Sorted Round-Robin algorithm distributes the videos evenly
Concluding Remarks

Conclusions

- Natural video partitioning algorithms were considered (both theoretically and experimental).
- Sorted Round-Robin has advantage over other algorithms
  - It is simple and fast

Although there are no textbooks on simplicity, simple systems work and complex ones don’t. 

[Jim Gray - 1998 Turing Award Winner]
Concluding Remarks

Conclusions

- Natural video partitioning algorithms were considered (both theoretically and experimental).
- Sorted Round-Robin has advantage over other algorithms
  - It is simple and fast
  - Has an approximation ratio of 2 for load balancing and 1 for storage usage balancing

[Jim Gray - 1998 Turing Award Winner]
Concluding Remarks

Conclusions

- Natural video partitioning algorithms were considered (both theoretically and experimental).
- Sorted Round-Robin has advantage over other algorithms
  - It is simple and fast
  - Has an approximation ratio of 2 for load balancing and 1 for storage usage balancing
  - Performs almost as well as Sorted Greedy for load balancing and is superior for balancing storage usage

Although there are no textbooks on simplicity, simple systems work and complex ones don’t.

[Jim Gray - 1998 Turing Award Winner]
Concluding Remarks

Conclusions

- Natural video partitioning algorithms were considered (both theoretically and experimental).
- Sorted Round-Robin has advantage over other algorithms
  - It is simple and fast
  - Has an approximation ratio of 2 for load balancing and 1 for storage usage balancing
  - Performs almost as well as Sorted Greedy for load balancing and is superior for balancing storage usage

Although there are no textbooks on simplicity, simple systems work and complex ones don’t. [Jim Gray- 1998 Turing Award Winner]

"Bounds on multiprocessing anomalies and related packing algorithms".

Sandvine (2012).
"Sandvine Global Internet Phenomena 1H".

"Methodologies for generating HTTP streaming video workloads to evaluate web server performance".