Multimedia 2: Technologien

SS 2007

Christian Breiteneder

Institut für Softwaretechnik und Interaktive Systeme
Arbeitsbereich Interaktive Multimediale Systeme

Zur Organisation

- Zeit: DO, 9.00 - 10.45
- keine Vorlesung am 29.3.2007 (Ersatztermin im April)
- homepage der Vorlesung unter *teaching* auf www.ims.tuwien.ac.at
- Unterlagen (in *Englisch*) download von homepage der VL
- schriftliche Prüfung: 29. Juni 2007

Contents

- Part 1: Media and Video Servers 4.5 E
- Part 2: Image und Video Retrieval 4.5 E
- Part 3: MPEG-7 und MPEG-21 2E
- Part 4: Digital TV and Interactive TV 2 E
- Part 5: Multimedia Standards 1 E

References

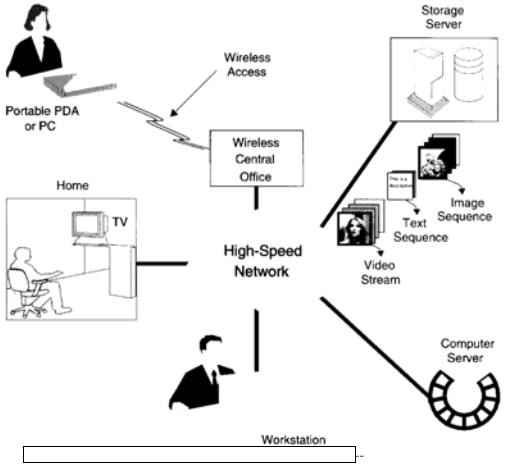
1. Video and Image Processing in Multimedia Systems. B. Furht, S.W. Smoliar, H. Zhang, Kluwer Academic Publishers, 1995.
2. Multimedia Programming. Objects, Environments and Frameworks. S.Gibbs, D.C. Tsichritzis, Addison-Wesley, ACM Press, 1995.
3. Multimedia Servers. Applications, Environments, and Design. D. Sitaram and A. Dan, Morgan Kaufmann Publishers, 2000.
4. The Essential Guide to Digital Set-top Boxes and Interactive TV. G.O'Driscoll, Prentice Hall, 2000.
5. Multimedia: Computing, Communications and Applications. R.Steinmetz, K.Nahrstedt, 2nd edition, Prentice Hall, 1999.
6. The Handbook of Multimedia Information Management. W. Grosky, R. Jain, R. Mehrotra (eds.), Prentice Hall, 1997.

Part I
Media Server Technology

Overview

- Introduction
- Application Scenarios
- Media Server Architectures and Components
  - BG: RAID Architectures
- Scheduling
- The Storage Subsystem

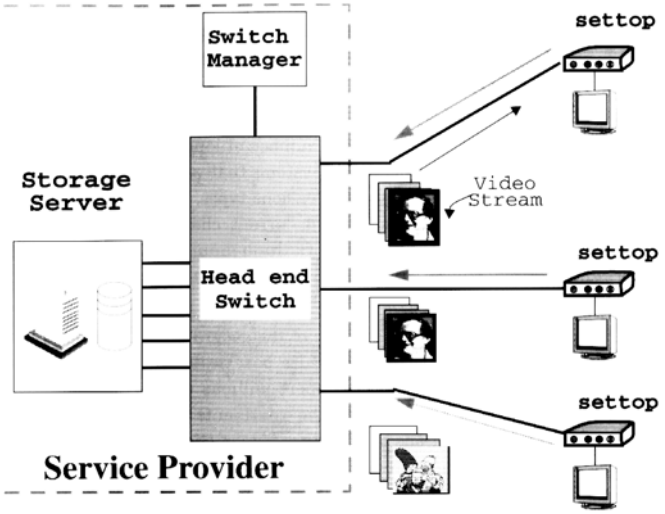
Introduction



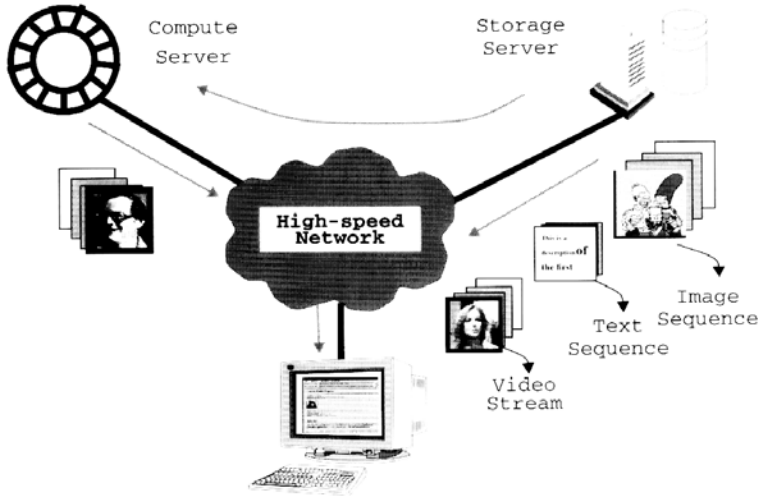
Introduction

- requirements of media servers are uniquely different from those of existing network-based servers
  - real-time nature of mm data requires QoS guarantees (guaranteed periodic retrieval and transmission of data)
  - such guarantees can be provided by *admission control* and *resource reservation*
  - *data layout* and *scheduling algorithms* must be designed to ensure that resource reservation is valid throughout a session duration

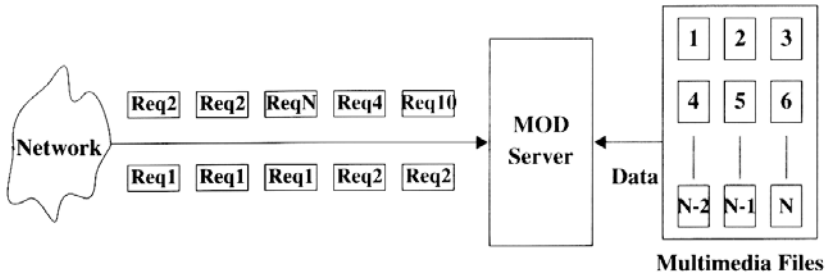
Application Scenarios



Application Scenarios



Request-Response Model



Properties of Request Arrival

spatial locality

- signifies that some data items are likely to be accessed more often than others
- arrival process can be modeled using Zipf's law
- exploited for data placement in storage hierarchy of different storage types

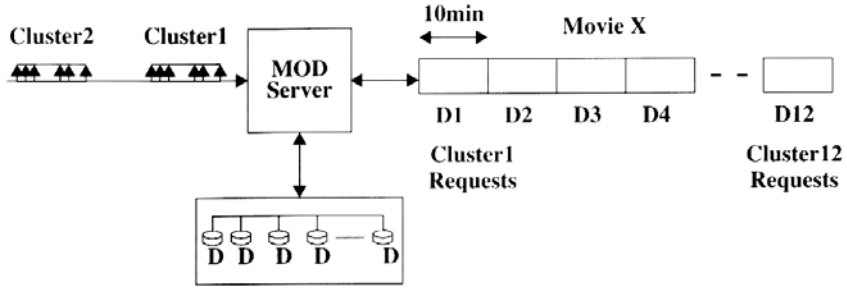
temporal locality

- request arrival process may be temporally clustered
- important in deciding the service and network model for the server

On-Demand Service Types

- Pay-Per-View (PPV)—access time decided by the server; server multicasts at fixed times
- Near Video-On-Demand (NVD) or Shared Viewing with Constraints (SVC)
  - observation: request arrivals are likely to exhibit significant temporal and spatial locality
  - the server processes and accepts requests in groups
  - a new client may face a variable admission latency
  - client becomes part of a multicast group

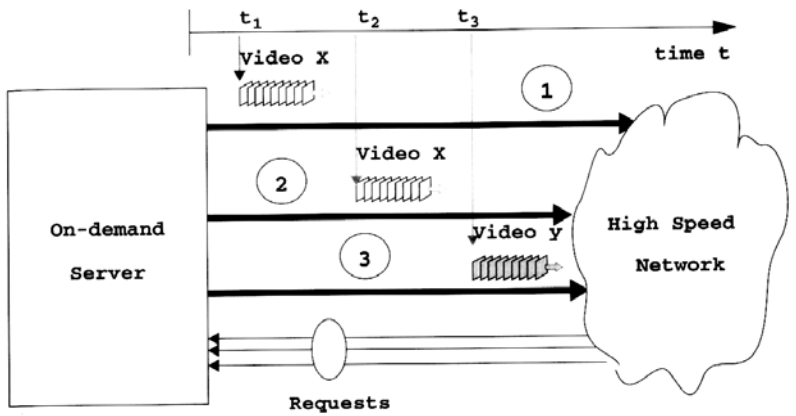
Shared Viewing with Constraints



On-Demand Service Types

- True-Multimedia-On-Demand (TMOD) or Dedicated Viewing
  - each access request is treated independently by the server
  - paradigm for personalized, interactive mm delivery
  - serious implications on the throughput

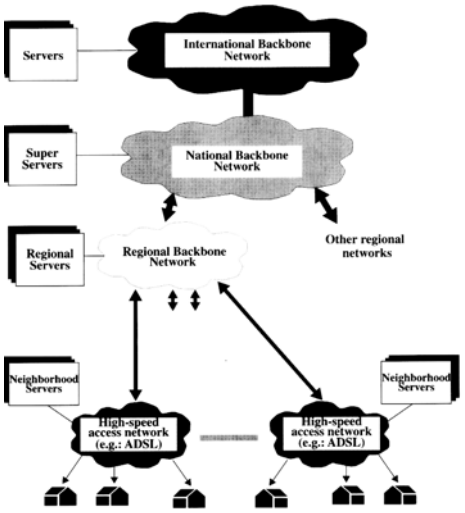
Dedicated Viewing Service



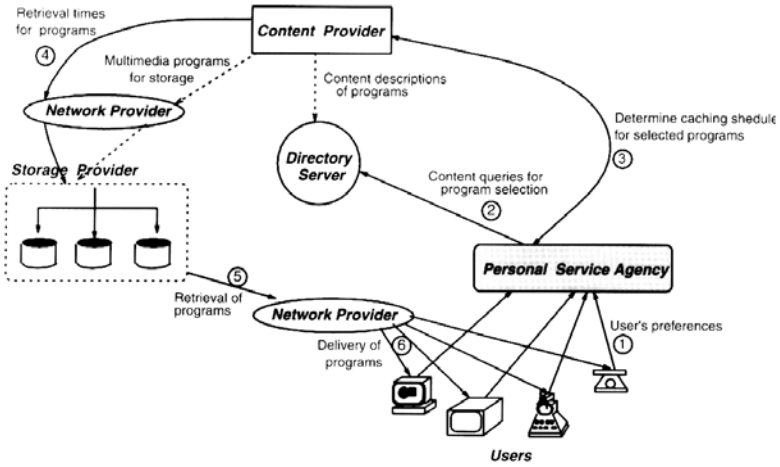
Network Model

- storage costs is the major factor of the total cost of a MODS
- 3 ways to scale network and storage costs
  - hierarchical solutions
  - caching
  - sharing

Hierarchical Network Model



Service Architecture



Media Server Design Issues

- application requirements
  - time sensitivity
  - open interfaces
  - client, network and server capabilities
- business deployment requirements
  - scalability
  - reliability
  - dynamic adaption to work load
- architectural requirements
  - topology
  - cost performance

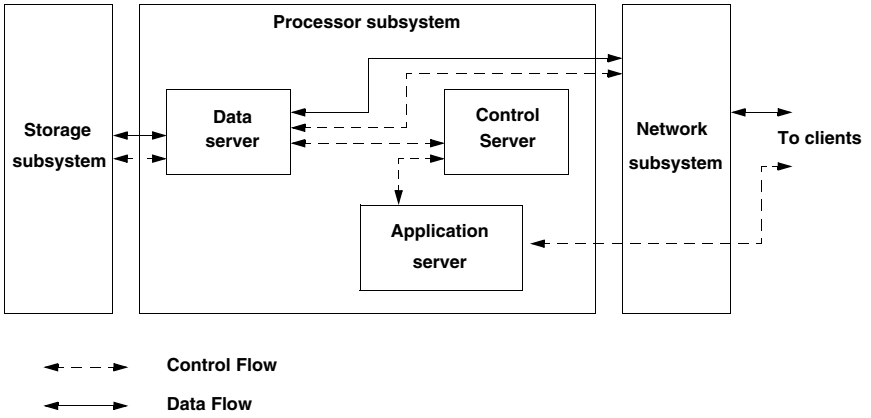
Performance Metrics

to compare and classify storage server architectures performance metrics are defined:

- concurrency—max. number of clients that can independently access a mm document
- access latency (and operation latency)—the amount of time a client has to wait after sending a request (to effect an interactive operation); less than a few seconds (< 1 sec); should be independent of server load
- storage capacity and storage, network throughput per $
- scalability
- extensibility—multiple application scenarios; extendable to support different service models

Media Server Architecture

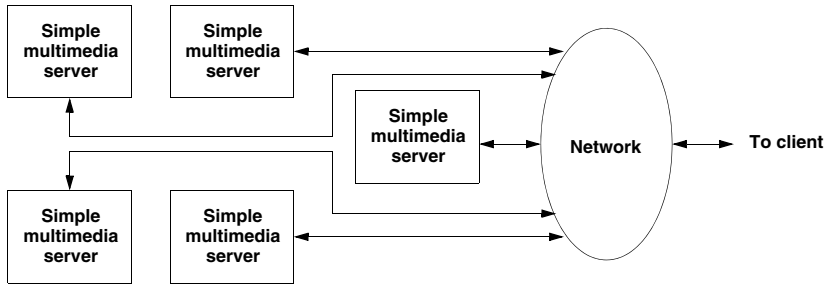
simple media server architecture



Simple Media Server

- *application server* (generally one per application) receives application commands from the client and converts them into media server commands.
- *control server*
  - admission control
  - optimization to increase server efficiency
  - hide configuration complexity from the application server
- *data server* responsible for actual data retrieval and delivery

Distributed Media Server

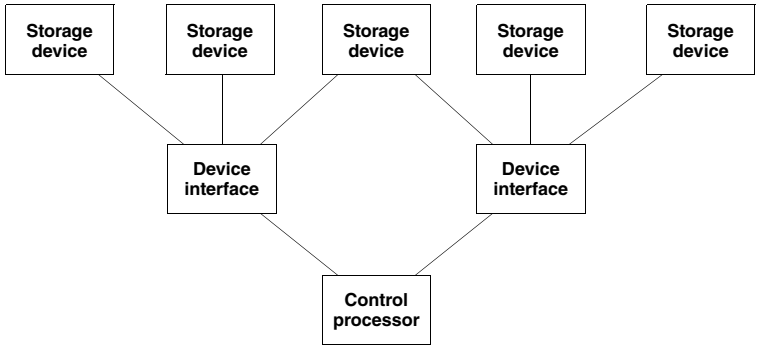


- subcomponents can be arranged in many different ways; need to be orchestrated to act as a single server

Distributed Media Server

- danger of load imbalance among individual servers
- *load-balancing policy* may require additional hw and complexity in sw: data path to copy files, control server
- large-scale servers constructed out of smaller components do enjoy some benefits from economies of scale, e.g. batching of multiple client requests for the same media component or *interval caching*

Media Server Hardware



- device interfaces—data buffering and speed matching; relieving the processor

BG: RAID

Redundant Arrays of Inexpensive Disks

- proposed in the 1980s as a way to use parallelism between multiple disks to improve aggregate I/O performance
- today they appear in the product lines of most major computer manufacturers
- 2 orthogonal concepts are employed:
  - data striping for improved performance
  - redundancy for improved reliability

Disc Technology Trends

- much of the motivation for disk arrays comes from the current trends in disk technology
- magnetic disk drives have been improving rapidly by some metrics and hardly at all by other metrics
- rapid improvements—increased density (disk capacities stay constant or increase while disk sizes decreased; increase in the transfer rate of disk drives)
- little improvements—seek times (from 20ms to 5-8ms today); rotational speeds (from 3600 rev. to 10.000 revolutions)

Disk Array Basics

- data striping distributes data transparently over multiple disks to make them appear as a single large disk
- striping improves aggregate I/O performance by allowing multiple I/Os to be services in parallel:
  - multiple independent requests—can be serviced in parallel by separate disks resulting in decreases of queuing times
  - single multiple-block requests—can be serviced by multiple disks acting in coordination resulting in increases of the effective transfer rate

Disk Array Basics

- the more disks in the array, the larger the potential performance benefits
- unfortunately, a large number of disks lowers the overall reliability of the disk array
- RAID organizations can be distinguished based on 2 features:
  - the granularity of data interleaving
  - the method and pattern in which the redundant information is computed and distributed across the array

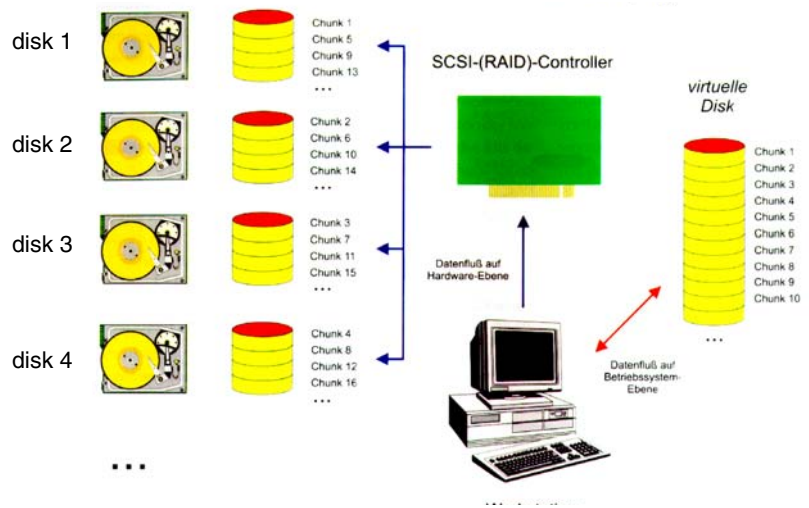
Disk Array Basics

- redundancy brings up 2 orthogonal problems:
  - selecting the method for computing redundant information; most redundant arrays use parity, some Hamming or Reed-Solomon codes
  - selecting a method for distributing the redundant information across the disk array (redundant information on a small number of disks vs. redundancy uniformly distributed across all disks
- concepts of data striping and redundancy are simple; however, selecting between the many possible schemes involves complex trade-offs between reliability, performance and costs

RAID Level 0

- a non-redundant disk array; consequences:
  - lowest cost of any RAID organization
  - best write performance
- surprisingly, not the best read performance; redundancy schemes can perform better on reads by selectively scheduling requests on the disks with the shortest expected seek and rotational delays
- widely used in supercomputing environments where performance and capacity are the primary concerns

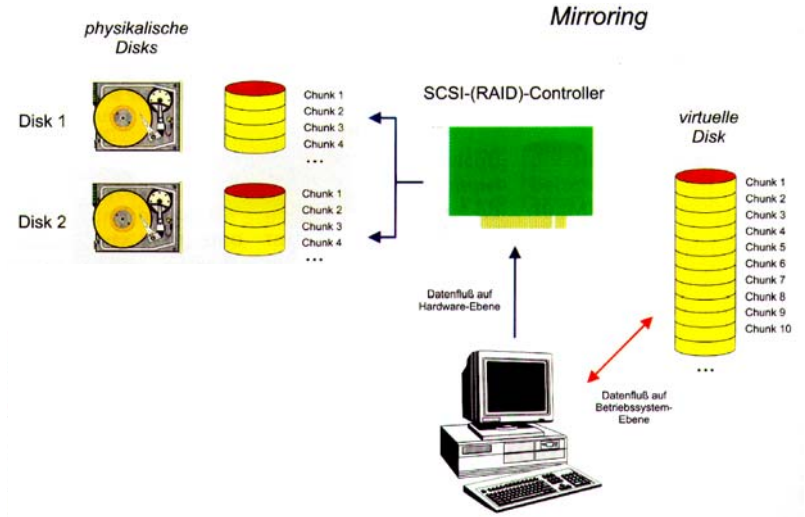
RAID Level 0



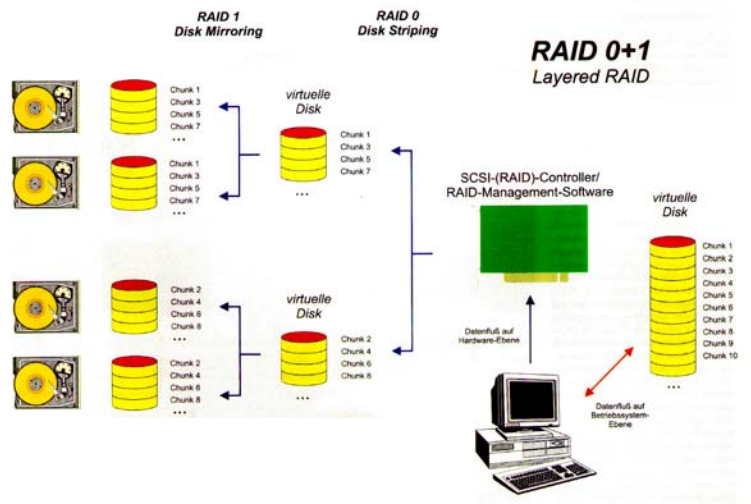
RAID Level 1

- *Mirroring* or *shadowing* uses twice as many disks as a non-redundant array
- always 2 copies of information
- mirroring is frequently used in database applications where availability and transaction rate are more important than storage efficiency

RAID Level 1



Layered RAID



RAID Level 2

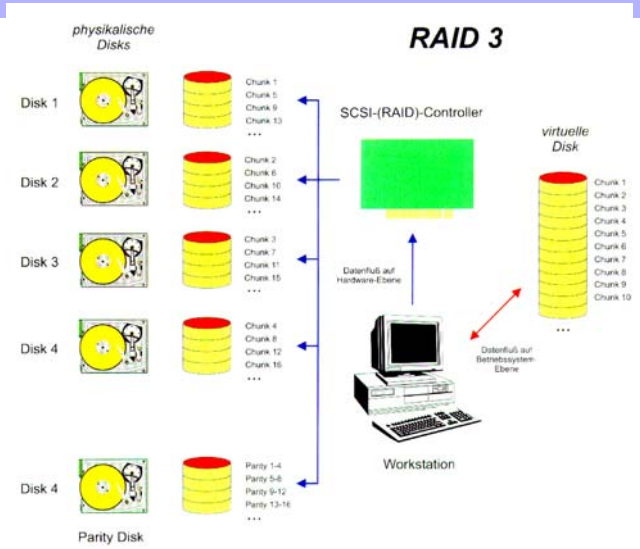
- memory-style ECC—memory systems have provided recovery from failed components with much less cost than mirroring by using Hamming codes
- Hamming codes contain parity for distinct overlapping subsets of components (e.g., 4 data disks and 3 redundant disks); number of redundant disks proportional to the log of the number of disks
- if a single component fails, multiple redundant disks are needed to identify the failed disk, but only one is needed to recover the lost information

RAID Level 3

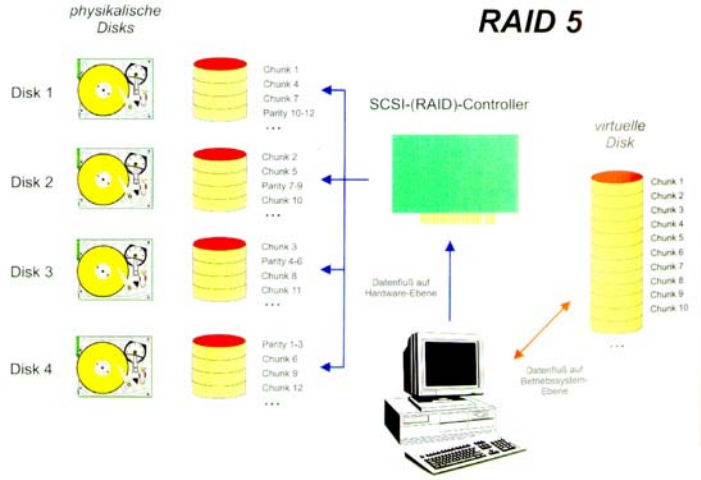
Bit-Interleaved Parity

- single parity disk—disk controller can easily identify which disk has failed
- data is conceptually interleaved bit-wise over the data disk
- each read request accesses all data disks, each write request accesses all data disks and the parity disk
- only one request can be serviced at a time
- frequently used in applications that require high bandwidth but not high I/O rates
- easier to implement than levels 4, 5 and 6

RAID Level 3

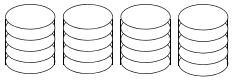


RAID Level 5

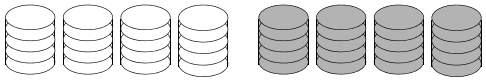


RAID Levels 0 through 3

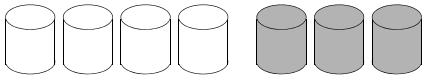
Non-redundant (RAID Level 0)



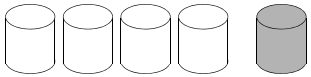
Mirrored (RAID Level 1)



Memory-Style ECC (RAID Level 2)

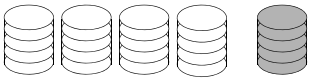


Bit-Interleaved Parity (RAID Level 3)



RAID Levels 4 through 6

Block-Interleaved Parity (RAID Level 4)



Block-Interleaved Distributed Parity (RAID Level 5)

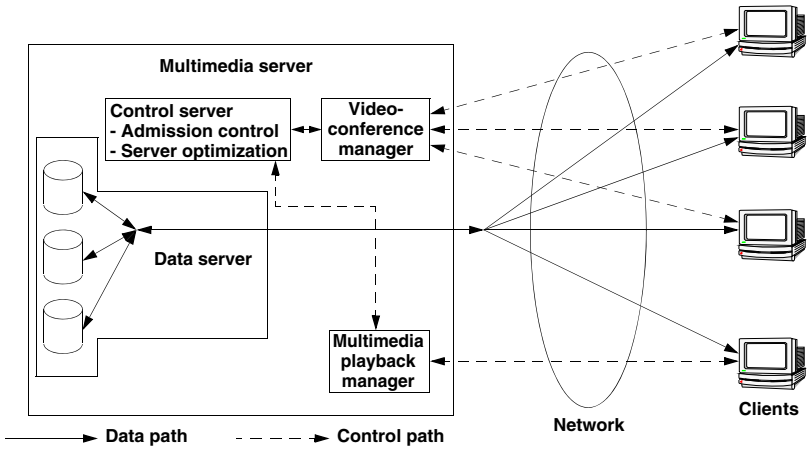
P+Q Redundancy (RAID Level 6)

Media Server Architectures

- Large-Scale Servers—What is the problem?
  - storage I/O bottleneck
  - network I/O bottleneck
- architectures of large-scale servers
  - distributed storage-based clustered mm servers
  - shared memory MIMD multiprocessor machines
  - massively parallel SIMD machines

Media Server Software

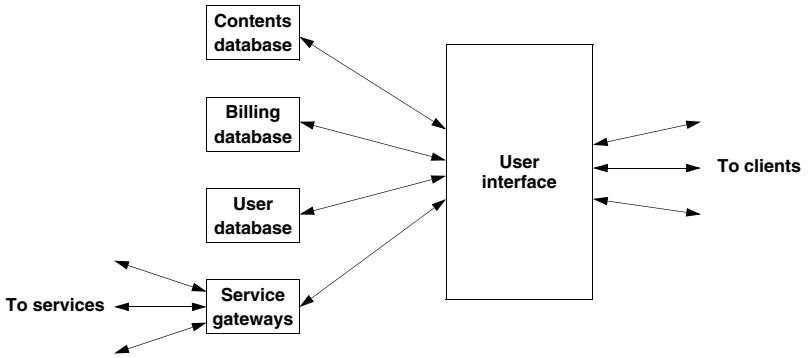
QoS components



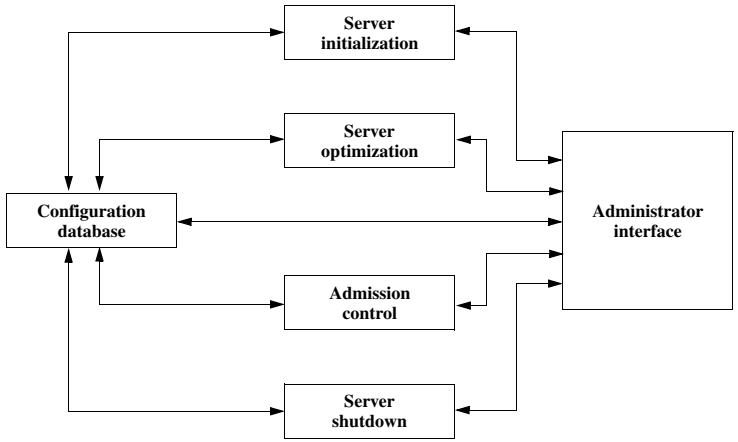
QoS Components

- the *media server* on the previous page is supporting 2 applications; to obtain resource reservations, both application servers interact with the control server.
- applications specify their requirements in application-specific terms: e.g. the videoconference manager may specify a particular bit-rate for video or a max. allowable transmission delay.
- the *control server* has to support a common set of requests that is general enough to allow most applications to specify their requirements; interaction between server optimization and admission control components (*resource manager*)

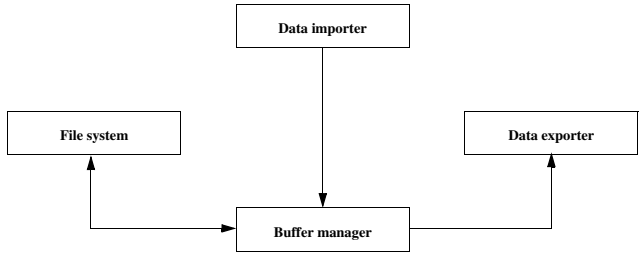
Application Server



Control Server

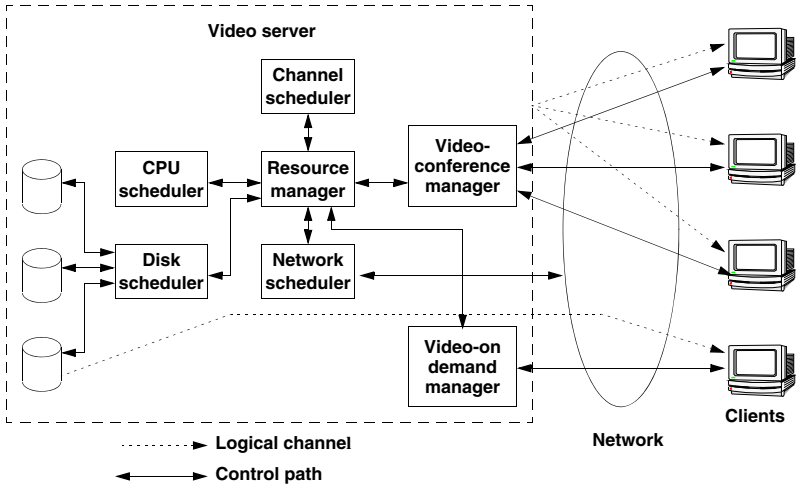


Data Server



- data importer—management of media input devices (e.g. a camera, video player);

Client Session Scheduling



QoS Specification

- QoS requirements may be *explicitly specified* by the application (e.g., for video conferencing stored with device information) or
- *implicitly specified* in an associated metafile, in a specific part of the file or as associated attributes (e.g., for VOD with the media file).
- most flexible implementation: implicit specification as well as allowing applications to override.

Explicit vs. Implicit QoS Specifications

explicit

- simpler implementation
- allows applications to override; useful for dynamic adjustment of QoS requirements

implicit

- simpler process of application development
- provides a migration path for non-mm applications, because server can provide QoS support
- storage of associated attributes may be required

QoS Parameters

- bandwidth requirements—delivery of specific data rates (may not be constant, average data rate)
- peak bandwidth requirements—maximum duration of peak bandwidth
- burst size or workahead—maximum amount of data by which the stream can exceed a strictly constant rate
- delay—maximum delay
- loss probability—maximum tolerable loss probability (control information more sensitive)

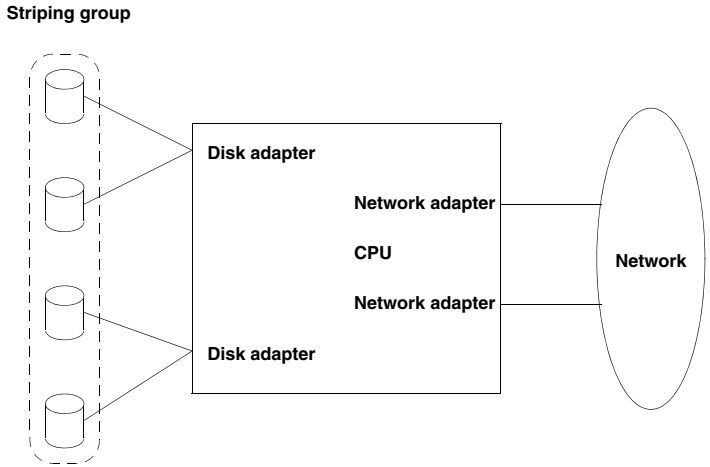
Capacity Estimation

- statistical admission control—controlling admission of user requests to guarantee the combined QoS requirements of all concurrent requests
- naive method: resource manager includes a table containing the rated bandwidth of each component
- calibration—estimation of bandwidth capacity of server components by measurement

Server Calibration

- The calibration process takes as input the configuration of the video server and runs a benchmark the server can support without violating QoS requirements.
- the calibration process may not be able to distinguish between the capacities of certain individual components (e.g. those in a pipeline configuration)
- such components can be aggregated into logical components

Server Calibration Process



Server Calibration Process

1. represent components of the system as a component graph (may be a directed graph)
2. aggregate components whose capacities cannot be distinguished by measurement; these are components for which the ratio of data flowing between them is fixed; 2 major cases:
   - data always flows either through both components or through neither (disk adapter and disk in the example)
   - semantics of configuration (e.g., disk striping)
3. regard one set of nodes in the graph as source (e.g., reduced nodes that include storage devices) and another set as destination (e.g., network adapters); for each pair enumerate all paths connecting a selected source and destination pair;

Server Calibration Process

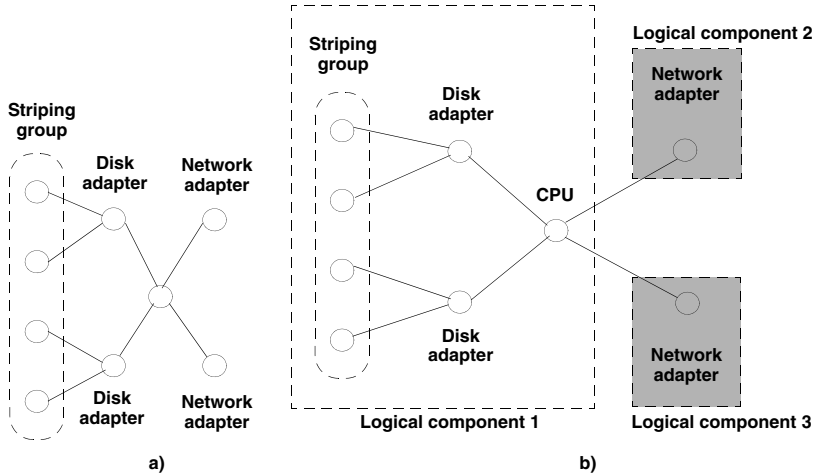
4. compute the weight of each path as the fraction of the overall data flow between pairs
5. to measure the capacity of the logical components, systematically play back data streams by selecting appropriate pairs as well as controlling the data rate;
6. the bandwidth capacity of each logical node is estimated by saturating it
   - ❑ order the paths passing through this node in terms of their weights
   - ❑ select path with highest weight and drive it with a maximum flow (saturation possibly due to a bottleneck at another node)

Server Calibration Process

   - ❑ additional flow through this selected node may be created by selecting remaining paths in order of their weights and driving each path to a maximum flow before selecting the next path in order

7. the resulting bandwidth capacity of the selected node is the sum of the concurrent flows through all paths.

Server Calibration Process



Logical Channel Setup

- ❑ the resource manager views the media server as an interconnected set of logical components
- ❑ in order to set up a logical channel for a request, the logical components must be selected, followed by reservation of resources on these components.
- ❑ selecting components for servicing a request is nontrivial; multiple sets of components can service a given request

Component Selection

- criteria for component selection
  - optimization
  - component characteristics
  - server topology—component selection may be based on component interconnection
- even after application of multiple criteria, there may be multiple paths with different sets of components
- final selection: load balancing; various strategies, e.g., path-based component selection policy

Client Request Scheduling

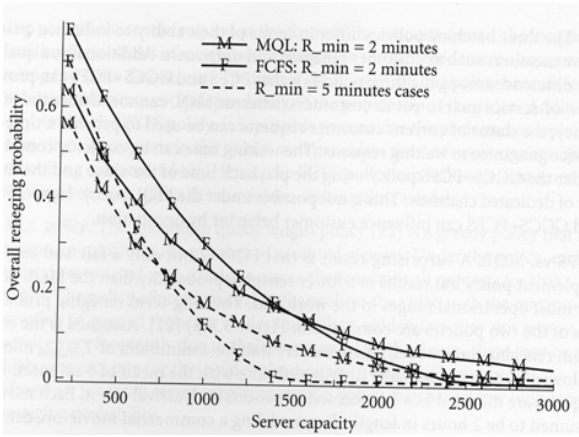
Scheduling Issues

- VCR control operations—a new logical channel is required upon resume; other operations (e.g. fast forward) also result in unpredictable demand in resource requirements
  - contingency channel policy—policy for handling such requests; it sets aside a small number of statistically shared contingency channels for unpredictable demands
  - algorithms for reoptimization or alleviating the higher demand for one resource by use of another available type

Scheduling Issues

- common data streams—sharing all or parts of a logical channel among multiple clients (batching policies, multicast)
  - data-centered approach
  - server capacity proportional to number of media items
  - interval caching
- time-varying workload

Client Behavior



- minimum reneging time / maximum waiting time guarantee

Channel Scheduling

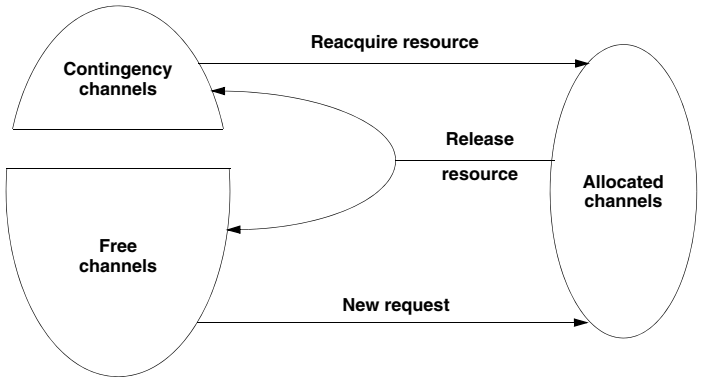
Objectives:

- minimize long-term reneging probability
- minimize short-term peak reneging probability
- minimize average waiting time
- fairness—a scheduling policy is fair if the reneging probability for all video requests is equal
- resume delay

VCR Control Operations

- pause and resume request implementation is fairly simple; major complexity: ensuring sufficient bandwidth upon resume
- other operations (fast-forward and reverse)
  - special files—bandwidth requirement different, discrete set of speeds, storage overhead
- fast playback—increased client implementation complexity, not supported by current decoding standards
- scan—selected frames are transmitted from server to client (suited to pull systems)

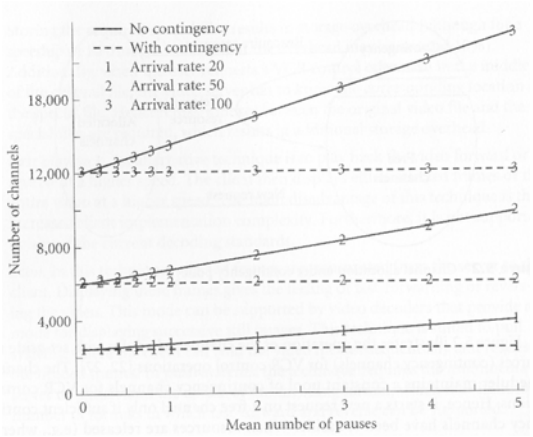
Bandwidth Allocation for VCR Control



Contingency Channel Policy Performance

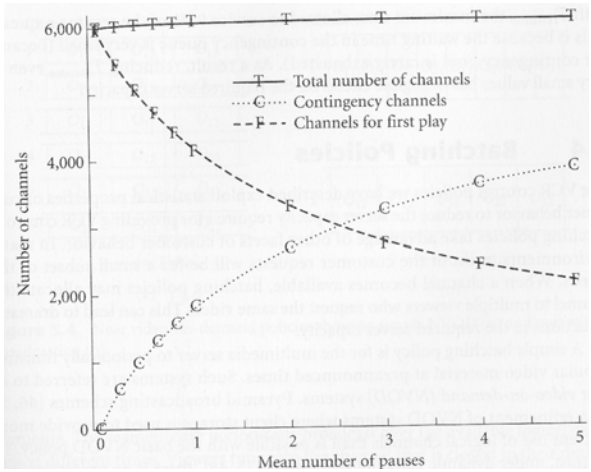
- case study
  assumes the workload depicted on page 63 and a system that stores 100 videos. The arrival rate is 50 requests/min. corresponding to a server with an average of 6000 active customers. Customers are assumed to pause once per video and the average duration of pauses is assumed 15 min.
  - reneging probability less than 5%
  - 95% of VCR control requests are to be satisfied in less than 30 seconds

Contingency Channel Policy Performance



required server capacities

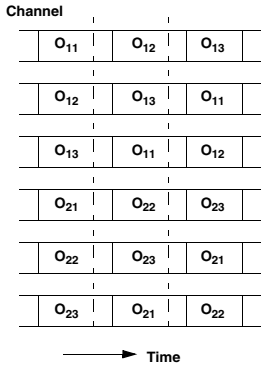
Contingency Channel Policy Performance



required contingency capacity

Batching Policies

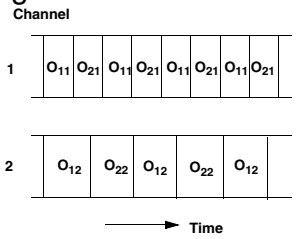
- simple NVOD



- in simple NVOD the maximum time a customer has to wait is $N_{objects} * T_{play} / N_{logical.server.chan}$ (33% of playing time)

Batching Policies

Pyramid Broadcasting



- the server bandwidth is divided into $N_{pyr.ch}$ logical channels; (example 2 log. channels, each with 3 times the bandwidth needed for playback)
- each video is subdivided into $N_{pyr.ch}$ segments whose sizes increase by $\alpha_{pyr.ch}$

Pyramid Broadcasting

- if $N_{pyr.ch}$ is increased, the size of the first segment decreases;
- the i-th segment of all videos is repeatedly broadcast on channel i;
- to play video i, the client starts download and play of O_{i1} when available
- scheme requires sufficient local storage of the order of the size of the largest segment
- value of $\alpha_{pyr.ch}$: for uninterrupted display, it must be possible to start download of the next segment before display of the current segment is completed
  max value of $\alpha_{pyr.ch}$: $N_{logical.server.chan} / (N_{objects} * N_{pyr.ch})$

Dynamic Batching Policies

- NVOD policies assume knowledge of access patterns of video objects
- to deal with uncertainties, batching policies are employed that can dynamically detect popular videos and serve multiple users in a single stream
- dynamic batching policies
  - FCFS policy—single request queue; time-of-service guarantee; fair
  - MQL policy—max. queuing length; greedy policy; each video has unique queue; high reneging rate for cold videos
  - GGCS-FCFS policy—group guaranteed server capacity; attempts to minimize the average wait time for requests

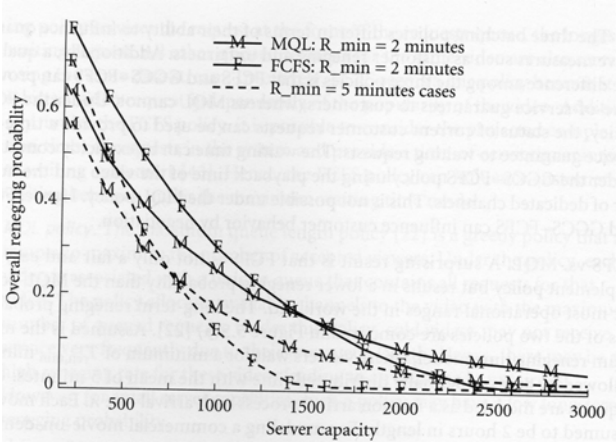
Impact of Dynamic Batching Policies

- the 3 policies differ in terms of their ability to influence quantitative measures such as customer reneging and unfairness
- FCFS and GGCS-FCFS can provide time-of-service guarantees, while MQL can not.
  - in FCFS request status can be used
  - in GGCS-FCFS waiting time can be computed (playback-time, number of channels)

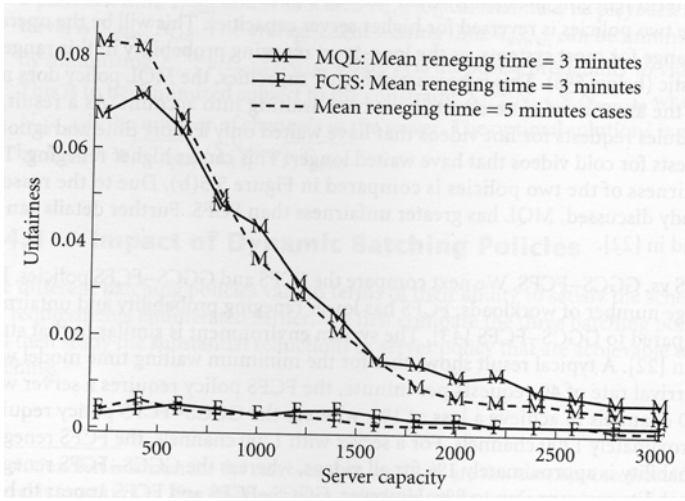
Impact of Dynamic Batching Policies

- FCFS vs. MQL—FCFS results in lower reneging probability than MQL;
- example (next pages)
  assumed minim. reneging time model, where viewers wait for a minimum of $T_{ren,min}$ followed by an exponentially distributed time with 5 min mean; user requests are modeled as a Poisson arrival process with arrival rate λ; movie length is 2 hours; λ is varied between 2 to 25 per minute (300 to 3000 users)
- for small server capacity, MQL has a lower long-term reneging probability

FCFS/MQL Reneging Probability



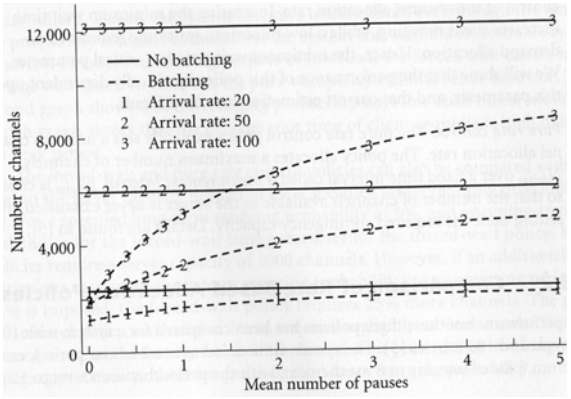
FCFS/MQL Unfairness



Impact of Dynamic Batching Policies

- FCFS vs. GGCS-FCFS—for a large number of workloads, FCFS has a lower reneging probability and unfairness; e.g., for an arrival rate of 40 requests per minute, the FCFS requires 1000 channels to achieve a loss of 1% whereas the GGCS-FCFS requires approx. 1200 channels;
- reduction in server capacity (batching vs. no batching)—depending on the workload the reduction can be as high as 70%;
  - reduction in required server capacity decreases as the number of VCR control operations increases; however, the reduction is greater for higher arrival rates (30% for highest rate)

Batching vs. No Batching



Scheduling in System Components

- CPU scheduling for continuous media differs from scheduling in other contexts, because continuous delivery requires scheduling of periodic requests with hard deadlines
- classical papers on scheduling of periodic tasks studied tasks with the following characteristics:
  - the only tasks with hard real-time deadlines are periodic tasks
  - the deadline for a task is the next time a request for that task is generated
  - the tasks are independent of each other
  - execution time for a task is constant and does not depend upon time or execution sequence

Scheduling in System Components

  - nonperiodic tasks do not have deadlines
  - tasks can be preempted before completion
- these characteristics apply better to CPU scheduling than that of other components;
- such tasks can be scheduled using either a static priority policy or a dynamic priority policy:
  - rate monotonic scheduling—static; associates 1 of a fixed set of priority levels to each task; tasks with greater frequency are assigned higher priority; tasks are scheduled in priority order

Scheduling in System Components

  - deadline scheduling—dynamic priority-based policy; deadline for a task is the time at which the next request for the task will be generated; the policy sets the priority of each task to its deadline; the task with the lowest priority (deadline) is scheduled first
- the two policies assume that nonperiodic requests are scheduled in the background;
  - could result in unnecessarily long response times
  - no guarantee can be provided for these requests

Hierarchical Scheduling

- a hierarchical scheduling policy partitions the processor among task groups; each group either further partitions the processor among subgroups or schedules tasks in the group;
- this allows each application to implement an appropriate scheduling policy
- in general, the lower-level policies that schedule between tasks are application-dependent
- an important requirement is the isolation of different subgroups from each other (partitioning independent of workload)