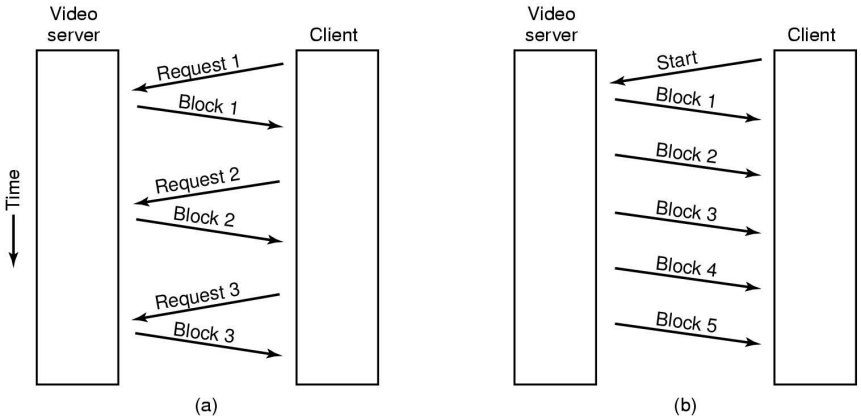
سیستم فایل چند رسانه‌ای

برای مدیریت دقیق یک سیستم چند رسانه‌ای، سیستم فایلی که استفاده می‌شود متفاوت از سیستم‌های سنتی است و باید از تکنیک‌های خاصی استفاده کرد. به عبارت دیگر مدل های سنتی به دلایل زیر به خوبی بر روی Serverهای ویدئو عمل نمی‌کنند. اول اینکه سرویس دهندگان ویدئو باید قابلیت این را داشته باشند که بلاک های داده ای حجیم را بدون تأخیر خوانده و بر روی خروجی خود ارسال کنند. از سوی دیگر کاربری که در سمت دیگر برای کار با سیستم های سنتی قرار دارد، باید خود زمانبندی دقیقی از درخواست های خواندن را برای سرویس دهنده ویدئو ارسال کند، به طوریکه اگر می‌خواهد Jitter اطلاعات دریافتی، یا به عبارت بهتر نرخ فریم های دریافتی یکنواخت باشد و احساس پرش در داده های دریافتی نکند باید درخواست های خواندن را خود تنظیم کرده و به فواصل زمانی منظمی برای سرویس دهنده ویدئو ارسال کند، که این امر عملاً ممکن نخواهد بود. معمولاً سیستم های فایل قدیمی را به عنوان سیستم های (Pull Server) می شناسیم. برای مدیریت سیستم فایل در چند رسانه‌ای از روش جدیدی که آنها را با نام Push Server می شناسیم استفاده می‌کنیم.در یک Push Server ، یک فرآیند کاربری یا یک User Process، یک فراخوان سیستمی را اجرا خواهد کرد. اجرای فراخوان سیستمیstart باعث می شود که سرویس دهنده ویدئو در پاسخ به این فراخوان سیستمی فریم های مورد نظر را با نرخی مناسب خوانده و ارسال کند.



نکته مهم در این مبحث، تفاوت های موجود در یک سیستم (Pull Server) به عنوان نمادی از یک سیستم قدیمی و یک سیستم (Push Server) که به عنوان ابزاری برای پیاده سازی سریع و مناسب سرویس دهنده ویدئو خواهد بود. در یک Push Server از سمت کاربر تنها یک فراخوانی سیستمی Start به سمت سرویس دهنده ویدئو ارسال می شود و در پاسخ بلاک های متوالی، خوانده شده و متوالیاً به سمت کاربر (Client) ارسال خواهد شد. اما در یک Pull Server به ازای هر بلاک باید یک درخواست از سوی کاربر به سمت سرویس دهنده ویدئو ارسال شود.

آن چه که حائز اهمیت است و در شکل نشان داده شده است، این است که برای پایان یافتن عملیات انتقال داده ها سیگنال دیگری به نام سیگنال Stop باید از سمت کاربر به سمت سرویس دهنده ویدئو فرستاده شود و با اجرای این فراخوان سیستمی سرویس دهنده عملیات (Transfer) یا انتقال را متوقف خواهد کرد. در سیستم Pull Server بلاک های داده ای، به سمت Client ارسال می شود و نرخ پردازش آنها و نمایش آنها تا حدی در اختیار Client خواهد بود. بنابراین استفاده از سیستم‌های Push Server برای سیستم‌های فایل چندرسانه ای مورد توجه قرار می‌گیرد.

توابع کنترل VCR

نکته دیگر در سیستم های چندرسانه ای بحث توابع کنترل VCR است. توابع کنترل VCR توابعی هستند که کاربر به کمک آنها می‌تواند کنترل تصویر نمایش داده شده را بر عهده گیرد. از جمله مهمترین آنها می‌توان به عملیات Pause، Forward،Rewind و مشابه آنها اشاره نمود. برای پیاده سازی یک عمل Pause در سیستمی که مبتنی بر Push Server است تکنیک چندان پیچیده‌ای مورد نیاز نیست. این عمل، عمل ساده‌ای است و برای پیاده سازی آن تنها کافیست یک درخواست خاص برای سرویس دهنده تعریف شود و Client در هنگام درخواست عمل Pause توسط شخص، این سیگنال را به سمت Server ارسال کند. در هنگام ارسال این سیگنال و پس از آنکه این سیگنال توسط سرویس دهنده کنترل شد، منابعی که برای انجام این عملیات در اختیار سرویس دهنده بودند باید آزاد شوند و باید دقت داشت که استفاده از چنین تکنیکی این

خطر را به همراه خواهد داشت که پس از آزاد سازی منابع و پس از ارسال مجدد سیگنال Play، امکان دریافت مجدد منابع وجود نداشته باشد. برای عملیات Rewind یعنی بازگشت به عقب نیز سرویس دهنده تنها کاری که باید انجام دهد این است که مجدداً از فریم صفر را به سمت Client ارسال کند. عملیات Forward و Backward مقداری مشکل تر هستند، به خصوص وقتی که بر روی بلاک‌های مربوط به فیلم در حال انتقال عملیات فشرده سازی انجام شده باشد. بنابراین توصیه می شود که چنانچه از چنین سیستمی استفاده می‌کنیم Client فایل‌های دریافتی یا داده های دریافتی را ذخیره کند.

دیدگاه دیگر این است که فایل های مختلفی از سمت سرویس دهنده ذخیره شود که هر یک از این فایل‌ها نرخ‌های متفاوت فشرده‌سازی را شامل هستند. هنگامی که عملیات Forward و Backward استفاده می شود امکان خواندن از فایل های مختلف وجود داشته باشد تا عملیات فشرده سازی تأثیری بر روی نرخ انتقال نداشته باشد. نکات منفی که بر روی سیستم های (Push Server) قابل تأمل است این است که:

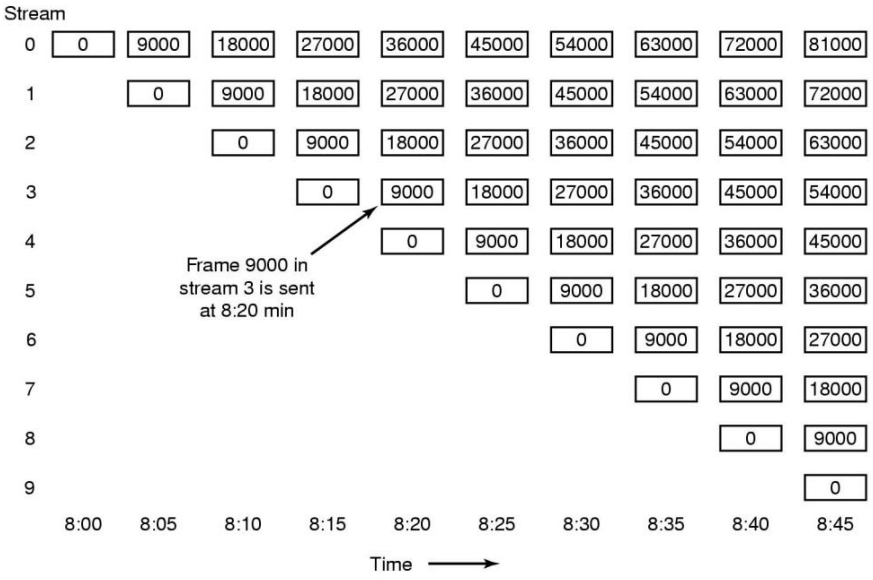
۱- اینگونه سیستم ها فضای دیسک بیشتری را نیاز دارد.

۲- عملیات Fast-forward و Fast-Backward نمی تواند به سادگی و در هر سرعت دلخواهی بر روی آنها انجام شود.

۳- پیچیدگی زیادی برای حرکت بین مدهای مختلف نمایشی در این گونه سیستم ها به سیستم تحمیل خواهد شد.

برای رفع این مشکلات و ارائه ی یک سیستم بهتر از مفهوم Near Video On Demand استفاده می کنیم. Near Video on Demand یا انتقال ویدئو بر حسب تقاضا یا بهتر بگوئیم نزدیک به انتقال ویدئو بر حسب تقاضا مفهوم جدیدی است که گرچه سعی می کند مفاهیم سیستم‌های چند رسانه‌ای را رعایت کند، اما در این سیستم ها یک کاربر، هر زمان هر عملیاتی را که بخواهد نمی‌تواند از سرویس دهنده مربوط به چندرسانه‌ای تقاضا کند.

Near Video On Demand



در این شکل می بینید که چطور ۹ استریم متوالی در حال اجراست و این استریم ها هر یک در زمان هایی به فاصله ۵ دقیقه آغاز شده است و داده‌های مشابهی را نسبت به نقطه آغازین استریم های دیگر دارا هستند. نکته دیگری که در این چیدمان باید مورد توجه قرار گیرد این است چگونه می‌توان عملیات مربوط به کنترل VCR را در سیستم های Near Video On Demand پیاده سازی نمود؟ تکنیک های خاصی برای این مفهوم مورد توجه است، اگر بخواهیم عملیات Backward را آغاز کنیم به عبارت دیگر بخواهیم کنترل را به نقطه ای قبل از نقطه Play جاری منتقل نماییم باید عملیات خاصی را داشته باشیم. این عملیات عیناً در عملیات حرکت به جلو یا Forward نیز صادق است. برای انجام عملیات Pause کار چندان مشکلی نداریم و تنها کافی است که استریم را Drop کرده و دیگر داده مربوط به آن را نمایش ندهیم. حال نحوه پیاده سازی عملیات مربوط به حرکت به عقب و حرکت به جلو در یک سیستم Near Video On Demand را بررسی خواهیم کرد.

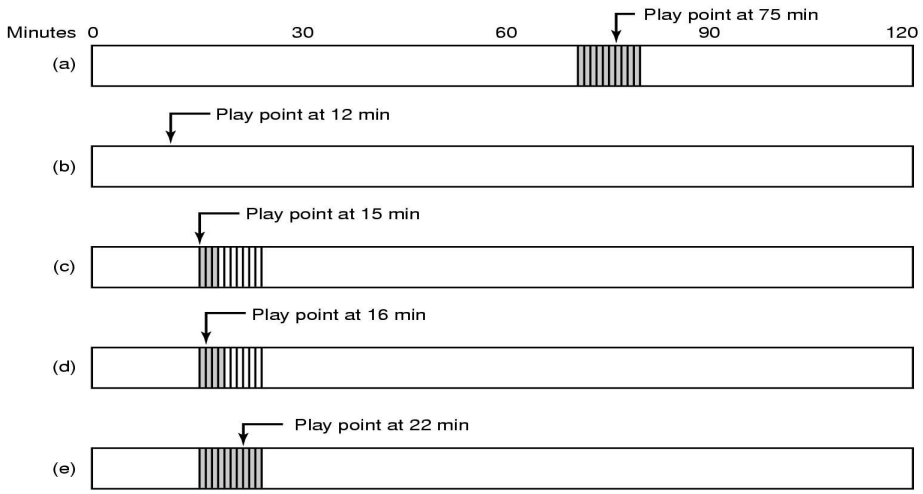
در این سیستم هر Client باید یک بافر داشته باشد که این‌بافر وظیفه ذخیره سازی تعدادی از فریم های ارسالی را بر عهده دارد. به عنوان مثال عنوان می‌شود که بافر مربوط به هر Client باید بتواند T دقیقه قبل و بعد از نقطه جاری را ذخیره کند. بدیهی است که ذخیره سازی T دقیقه قبل کار مشکلی نیست و تنها نیاز به فضای اضافه دارد. اما برای ذخیره کردن T دقیقه آینده که هنوز داده‌های آن به این ماشین ارسال نشده است باید تدبیر دیگری اندیشید. بنابراین برای مدیریت کردن چنین پدیده ای Client ها باید توانایی این را داشته باشند که دو استریم را همزمان بخوانند. منظور از استریم جویبار داده یا مسیرهای ارتباطی داده‌هاست.

نکته دیگر در سیستم‌های Near Video On Demand این است که استریم‌های متعددی در این سیستم‌ها تعبیه می‌شود و استریم ها به این شکل عمل می کنند که هر استریم در بازه های متناوب و منظم تولید و آغاز می شود. به عنوان مثال اگر کاربری بخواهد در زمان ۸:۱۵ دقیقه شروع به مشاهده یک فیلم کند، باید Client مربوط به آن کاربر استریم مربوط به ۸:۱۵ دقیقه را بخواند و در این هنگام این Client باید به طور همزمان استریم مربوط به ۱۰:۸ دقیقه را نیز ذخیره کند تا امکان ذخیره سازی ۵ دقیقه قبل و بعد برای وی در هر لحظه مهیا باشد.

بدیهی است در لحظه ۸:۲۰ دقیقه استریم مربوط به ۸:۱۵ دقیقه حذف خواهد شد. به عبارت دیگر داده‌های ذخیره شده مربوط به این استریم از سمت ماشین کاربر حذف خواهد شد. نکته ای که باید در اینجا به آن اشاره کرد این است که استریم‌های متعدد در سیستم وجود دارد، اما کاربردر هر زمان نمی‌تواند هر چیزی را مشاهده کند. به عبارت دیگر اگر ما فرض کردیم که استریم‌ها در هر ۵ دقیقه آغاز می‌شوند، آنگاه کاربر نخواهد توانست در ساعت ۸:۰۳ دقیقه به تماشای یک فیلم بنشیند، به عبارت دیگر نمی‌تواند در ساعت ۸:۰۳ دقیقه فیلمی را از ابتدا نگاه کند. بنابراین در چنین شرایطی کاربر باید منتظر بماند تا لحظه‌ای که یک استریم جدید آغاز می شود.

بررسی نحوه پیاده سازی حرکت به عقب و حرکت به جلو در Near Video On Demand

بدیهی است اگر در هنگام عملیات حرکت به عقب نقطه مورد نظر خارج از بافر باشد یعنی با فرض اینکه هر Client به طول2T نسبت به نقطه جاری ، بافر را ذخیره کرده است، چنانچه نقطه مورد درخواست خارج از بافر باشد مسئله متفاوت خواهد بود از هنگامی که نقطه مورد درخواست در داخل بافر است. اگر نقطه مورد درخواست در داخل بافر باشد به سادگی عملیات انتقال انجام خواهد شد. اما اگر این نقطه در خارج از بافر باشد در سیستم Near Video On Demand فرض خواهد شد که یک استریم خصوصی و منحصراً برای این Client ایجاد می شود.



در شکل a می‌بینید که کاربری در حال مشاهده فیلمی است که در بازه‌های ۵ و با استریم هایی که فاصله زمانی آنها ۵ است نمایش داده شده است.اگر play point یعنی نقطه‌ای که کاربر در حال مشاهده آن است را فرض کنیم در زمان ۷۵ دقیقه است، آنگاه می دانیم که داده‌های مربوط به زمان ۷۰ تا ۸۰ در بافر مربوط به این client وجود دارد. حال اگر در شکل بعد play point را به نقطه ۱۲ منتقل کنیم با توجه به اینکه ۱۲ کاملا از بافر مربوط به play point اولیه دور است، نیاز به یک استریم خصوصی داریم.

استریم خصوصی در اینجا دائمی نخواهد بود، به عبارت دیگر در نقطه ۱۲، استریم خصوصی برقرار می‌شودتا پخش فاصله زمانی بین ۱۲ تا ۱۵ از روی آن صورت گیرد و همزمان داده‌های مربوط به بازه زمانی ۱۵ به بعد از استریم مربوط به دقیقه ۱۵ به بافر منتقل می شود. بعد از ۳ دقیقه یعنی در زمان ۱۵ سیستم می تواند از بافر که تا دقیقه ۱۸ را در خود دارد تغذیه شودو بنابراین دیگر نیازی به استریم خصوصی نیست، لذا در این زمان می توان استریم خصوصی را حذف کرد. همزمان با پخش، بافر از استریم Near Video On Demand که الان در دقیقه ۱۸ است شروع به پر شدن می‌کند. همانطور که در شکل d مشخص است بعد از یک دقیقه play point در دقیقه ۱۶ است و بافر داده‌های مربوط به بازه زمانی ۱۵ تا ۱۹ را در خود دارد در مورد بافر دوست داشتیم در هر لحظه بتوانیم به اندازه T واحد قبل و بعد را در بافر داشته باشیم. بعد از گذشت ٦ دقیقه دیگر، بافر همانطور که در شکل e نشان داده شده است پر شده است و play point در دقیقه ۲۲ است. با اینکه play point در وسط بافر نیست ولی در صورت لزوم این مسئله را نیز می توان برطرف کرد.

قرار گرفتن فایل ها بر روی دیسک در یک سیستم چند رسانه‌ای

دراین مبحث، نحوه قرار گرفتن فایل ها بر روی دیسک در یک سیستم چند رسانه ای بررسی خواهد شد. به عبارت دیگر مفهوم File placement در سیستم های چند رسانه ای باید مورد بررسی قرار گیرد. سیستم های چندرسانه ای و فایل های چند رسانه ای ، دارای چند خاصیت هستند که با توجه به این خواص باید نحوه قرار گرفتن آنها بر روی دیسک را بررسی کرد.

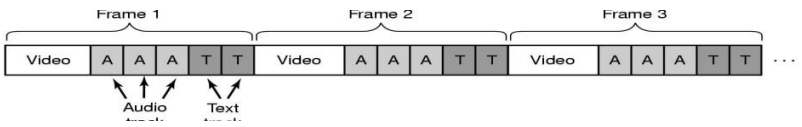
۱- این فایل ها معمولاً بزرگ هستند.

۲- این فایل ها معمولاً یکبار نوشته شده و چندین بار خوانده می شوند.

۳-معمولاً دسترسی به فایلهای چند رسانه ای مانند ویدیو یک دسترسی ترتیبی است و دسترسی تصادفی اگرچه وجود دارد اما کمتر مورد توجه است.

این خصوصیات باعث می شود که برای مدیریت این فایل‌ها بتوان از تکنیک های دیگری که آنها را از سیستم های عامل عمومی و قدیمی متمایز می کند استفاده کرد.

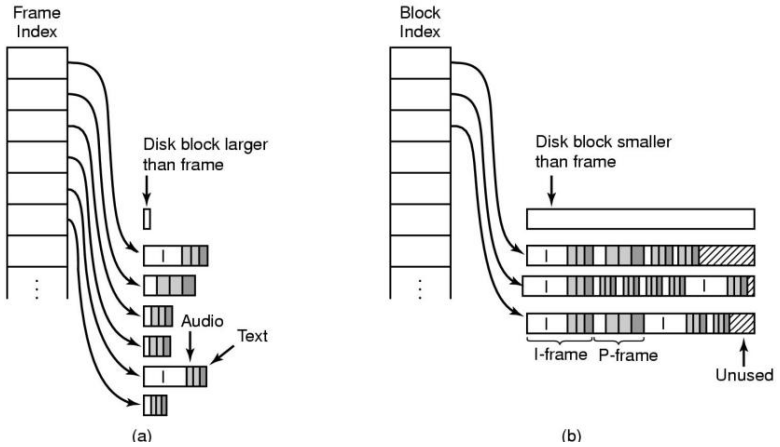
نحوه‌ی قرار گرفتن فایل های چند رسانه‌ای بر روی یک دیسک باید به گونه ای باشد که ایجاد جویبار داده‌ها بر روی درخواست‌های ارسال شده حتی‌الامکان بدون هیچ گونه Jitter باشد. به عبارت دیگر این خاصیت یکی از خواص بسیار مهمی است که به شدت تحت تأثیر نحوه‌ی قرارگرفتن فایلهاست. به عبارت دیگر وجود Jitter در نمایش چنین فایل هایی از آنجا ناشی می‌شود که ممکن است در داخل یک فریم یک عمل seek اتفاق افتد. اگر به خاطر بیاورید منظور از عمل seek در یک دیسک، جا به جایی هد است، به عبارت دیگر زمانی که طول می کشد که هد مربوط به دیسک از یک سیلندر به سیلندر دیگر منتقل شود را seek time می‌نامیم، لذا انجام عمل seek در داخل یک فریم مجاز نخواهد بود. بنابراین باید دقت نمود که برای داشتن چنین خاصیتی، دیگر نمی توان فایل ها را چند پاره کرده و بر روی نقاط مختلف دیسک قرار داد، بنابراین سیستم پیوسته ی فایل، راه حل خواهد بود. نکته ی دیگری که باید به آن دقت شود و یکی از پیچیدگی‌های این نوع سیستم هاست، این است که نمایش همزمان ویدیو و متن و صدا کار چندان ساده ای نیست. همانطور که قبلاً عنوان شده بود، این سه مؤلفه، مؤلفه های یک فیلم هستند. یکی از تکنیک هایی که برای حل پیچیدگی این مسأله مطرح می شود، استفاده از فایل های جداگانه برای هر یک از این مؤلفه هاست. این راه حل، چندان راه حل مناسبی نیست، به این دلیل که باعث ایجاد seek در خواندن یک فریم، بین فایل‌های مختلفی می شود که داده های مختلف مربوط به یک فریم را نگهداری می کنند. بنابراین راه حل بهتر این است که این داده ها همگی در یک فایل و به صورت interleave در داخل فایل ذخیره شوند.



با توجه به شکل، در فریم ۱، ویدئو، صدا و متن و همین طور در فریم ۲، کلیه این داده ها وجود دارند. استفاده از چنین سازماندهی، چندین نکته در بر خواهد داشت. اول اینکه باعث زیادشدن عملیات I/O مربوط به دیسک می شود، زیرا در خیلی از موارد انواع و اقسام صدا و متن‌هایی که بر روی یک فایل وجود دارد توسط کاربر مورد نیاز نیست و آنها را فقط خوانده و دور می اندازد. دوم اینکه به دلیل حجم بالای اطلاعات مربوط به صدا و متن که در ویدئو ادغام شده‌اند نیازمند فضای بافرینگ بزرگتری هستیم. اما نکته‌ی دیگر، این است که استفاده از این روش به کلی انجام عملیات seek در داخل یک فریم را حذف می‌کند، از سوی دیگر با داشتن چنین ساختاری برای یک فایل، دسترسی تصادفی و انجام عملیات Fast-Forward و Fast-Rewind نیازمند ساختمان های داده‌ای دیگریست. نکته ی دیگری که باید به آن اشاره کرد این است که برای خواندن فیلم های مختلف قطعاً استفاده از این تکنیک باعث می شود که ما نیازمند seekهای متعددی باشیم که ممکن است در هنگام نمایش بیش از یک فیلم اثرات مخرب بر روی Jitter هر یک از این فیلم‌ها داشته باشد.

استراتژی سازماندهی فایل

برای سازماندهی فایل ها در یک سیستم چند رسانه‌ای دو مدل پیشنهاد می‌شود. این دو مدل، مدلهای غیرپیوسته هستند، یعنی در این مدلها فریم های مربوط به فیلم ها لزوماً پشت سر هم قرار نمی‌گیرند. این دو مدل small block و large block می‌باشند.



Small Disk block

با توجه به شکل (a)، در مدل small block اندازه‌ی بلاکهای مربوط به دیسک، کوچکتر از میانگین اندازه‌ی فریم هاست. در این حالت فرض می شود که فریم‌های مربوط به یک فیلم و حتی فریم های مربوط به صدا و متن و ویدئو نیز از یکدیگر جدا هستند. در این سیستم یک frame index وجود دارد، که با توجه به شکل، به ازای هر فریم یک ورودی دارد که در داخل آن ورودی، مکان قرار گرفتن آن فریم ذخیره شده است. بنابراین با توجه به آنچه که در شکل قرار دارد، در این سیستم به ازای درخواست یک فریم، مجموعه ی اندازه های متفاوتی از داده ها خوانده می شود و فرض بر این است که اطلاعات هر فریم در کنار یکدیگر ذخیره می شود. کوچک بودن اندازه ی این بلاک‌ها باعث می شود هیچ گاه فضای مصرفی بلاک هدر نرود. بطور حتم در این ساختار، اندازه فریم در شاخص فریم مورد نیاز است. زیرا اندازه‌ی فریم های مختلف با یکدیگر متفاوت است و برای اینکه سیستم بداند به ازای هر فریم چه تعداد داده ی پشت سر هم را باید بخواند، باید اندازه ی فریم را بداند.

Large Disk block

در این مدل فرض می شود که بلاک دیسک به اندازه ای بزرگ است که در آن بیش از یک فریم جای خواهد گرفت. در این سیستم به جای frame index از مفهوم block index استفاده می شود. Block index مشخص خواهد کرد به ازای هر بلاک چه فریم هایی در آن قرار گرفته و در کجا قرار دارد و به کمک block index می توان مشخص نمود که کدام فریم، فریم آغازین این بلاک است. با توجه به شکل (b)، بلاک‌های مختلفی که در هر بار درخواست خواندن، خوانده می شوند، اندازه‌ی یکسانی دارند.

برای نقد large disk block موارد زیر را عنوان می کنیم:

به عنوان اولین نکته در large disk block باید عنوان شود که، چنانچه فریم بعدی در داخل بلاک جاری جا نشود چه باید کرد؟

یکی از تکنیک های مطرح، این است که بقیه ی بلاک را خالی گذاشت. با توجه به شکل در بالا چنین اتفاقی افتاده است. بنابراین استفاده از این کار باعث می شود که در وسط خواندن یک بافر نیاز به هیچ عمل seek نباشد. از سوی دیگر این عمل باعث اتلاف فضای حافظه خواهد شد. دومین تکنیکی که در هنگامی که یک فریم در یک بلاک جا نمی شود می توان بکار برد آن است که فریم را شکسته و قسمت دوم را در بلاک دیگر قرار داد. این تکنیک باعث ذخیره فضای دیسک و کاهش کارایی سیستم خواهد شد.

یکی از بهترین روشها ترکیب این دو روش با یکدیگر است، یعنی بتوان یک موازنه (tradeoff) بین small block و large block ها ایجاد کرد. استفاده از مفهوم frame index باعث مصرف حافظه‌ی اصلی بیشتری خواهد شد و فضای تلف شده بر روی دیسک ها در عین حال کاهش خواهد یافت. اما استفاده از block index ی که در آن فریم ها شکسته نمی شوند، استفاده از میزان حافظه‌ی اصلی، یعنی RAM کاهش خواهد یافت در عین حال فضای دیسک تلف خواهد شد اما اگر block index را در حالتی در نظر بگیریم که اجازه ی شکستن فریم ها بر روی بلاک ها داده شود، گرچه فضای استفاده از

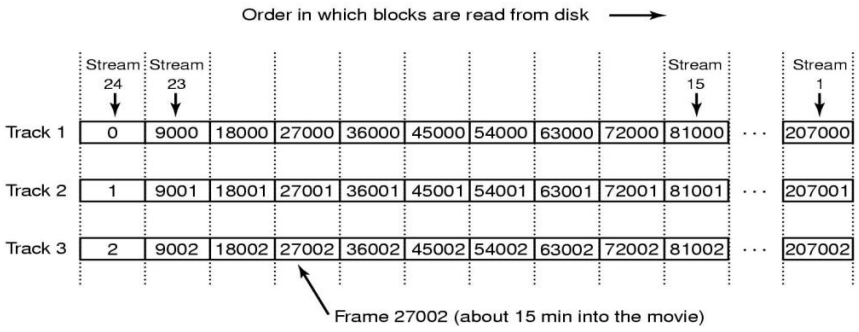
RAM کم خواهد بود و هیچ اتلاف دیسکی نداریم، اما تعداد seekهای زیادی ممکن است در داخل بلاک های مختلف ایجاد شود و این امر باعث کاهش یا افزایش Jitter به عنوان یک معیار quality of service می شود. از سوی دیگر باید دقت کرد که مدیریت فضای دیسک در مدل small block کمی پیچیده تر است، زیرا باید بتوانیم یک مجموعه ای از بلاک های پشت سر هم را مدیریت کنیم.

برخی از نکات این دو روش را به صورت تیتروار مقایسه می‌کنیم :

در small block ها، بافرینگ به خوبی انجام شده و در عین حال اندازه بافر می تواند متغیر باشد. در large blockها، دیسک می تواند با حداکثر سرعت عملیات را انجام دهد. در small block ها طول زمانی، مقدار ثابتی دارد و میزان playing time مورد نیاز معمولاً میزان مشابهی خواهد بود. در large blockها طول داده ای میزان ثابتی خواهد داشت و بلاک های داده ای اندازه‌ی یکسانی خواهند داشت.

در خاتمه مبحث، نحوه چیدمان داده ها و فایل‌ها بر روی یک دیسک، به حالتی که near video on demand موجود است دقت کنید. در حالت near video on demand یک فیلم مشخص با بازه های زمانی مختلفی شروع به نمایش می شد.

اگر فرض کنیم که این فیلم بر روی یک فایل پیوسته قرار گرفته است به ازای هر استریم که در سیستم وجود دارد یک عمل seek مورد نیاز بود و این عمل seek مشکلاتی را ایجاد می کرد. بنابراین پیشنهاد می شود که چیدمان فایل‌هایی که تحت near video on demand قرار می گیرند به صورت شکل زیر باشد.



با توجه به شکل، track1 شامل داده های مربوط به ۰، ۹۰۰۰، ۱۸۰۰۰، ۲۷۰۰۰ و به همین ترتیب خواهند بود. این تفاوت فاصله از آنجایی ناشی می شود که فرض شده است که تفاوت بین دو استریم ایجاد شده ۵ دقیقه است و بنابراین با در نظر گرفتن یک نرخ فریم ثابت می توان به این نتیجه رسید که با آغاز هر استریم چه فریم هایی باید خوانده شود. محاسبه نرخ فریم در این مثال بر عهده دانشجویان قرار گرفته می‌شود. بدیهی است که با داشتن چنین ساختاری با خواندن track1 داده اولیه برای بسیاری از استریمها و یا به عبارتی برای تمامی استریم‌ها مهیا می شود، و به ازای استریم‌های مختلف نیازمند انجام عمل seek نخواهیم بود. این روند برای track2 و همچنین برای track3 اتفاق خواهد افتاد. بنابراین با این چیدمان ابتدا track1 و سپس track2 و به همین ترتیب track3 خوانده خواهد شد و کلیه استریم‌ها می‌توانند داده مربوط به خود را بردارند.