دانشکده‌ی مهندسی کامپیوتر

شماره دانشجویی: ۹۹۱۰۵۵۶۱

شایان صالحی

تمرین پنجم

پاسخ مسئله‌ی ۱.

الف

در اینجا می‌دانیم که هر direct index به فضای ۱۰۲۴ بایتی اشاره کرده و از طرفی آز آنجایی که هر پوینتر ۵ بایت جا می‌گیرد برای indirect index می‌دانیم که به یک فضای $۱۰۲۴/۴ = ۲۵۶$ بایتی اشاره می‌کند. همچنین برای 2-level indirect index می‌دانیم دو مرحله از این فضادهی را خواهیم داشت. به این ترتیب برای کل فضایی که می‌توان آدرس‌دهی کرد داریم:

$$\overbrace{۶۴ \times ۱۰۲۴}^{\text{direct index}} + \overbrace{۲۵۶ \times ۱۰۲۴}^{\text{indirect index}} + \overbrace{۲۵۶ \times ۲۵۶ \times ۱۰۲۴}^{\text{2-level indirect index}}$$

$$\text{کل فضایی که می‌توان آدرس دهی کرد} \implies ۲^{۱۰} \times (۲^{۶} + ۲^{۸} + ۲^{۱۶}) \implies$$

ب

از آنجایی که فایل در موقعیت ۳۰۰۰۳۱۲ قرار دارد و بلوک‌های ۱۰۲۴ تایی داریم این موقعیت در خانه $۲۹۳۰ = \lfloor ۳۰۰۰۳۱۲/۱۰۲۴ \rfloor$ قرار دارد. بنابراین این قسمت مربوط به آدرس‌دهی 2-level indirect index می‌شود که نیاز به ۳ عمل خواندن از حافظه یا disk access دارد.

پاسخ مسئله‌ی ۲.

در اینجا ابتدا می‌بایست تاخیر را محاسبه کرد. برای این کار یک تاخیر چرخش داریم که برابر است با:

$$\text{تاخیر چرخشی} \longrightarrow \frac{۱}{(\frac{۱۵۰۰۰}{۶۰})۲} = \frac{۱}{۵۰۰} = ۲\text{ms} \implies$$

از آنجایی که تاخیر seek time دو برابر این مقدار است مقدار آن برابر ۴ میلی‌ثانیه خواهد بود. حال برای تاخیر انتقال فایل داریم:

$$\text{تاخیر انتقال فایل} \longrightarrow \frac{۵۱۲}{۱۰۷} \simeq ۰/۰۱\text{ms} \implies$$

که زمان کنترلر ۱۰ برابر آن بوده که برابر یک دهم میلی‌ثانیه خواهد بود در نهایت تاخیر خواندن و نوشتن به این صورت خواهد بود:

$$\text{Read / Write latancy} \implies ۴ + ۲ + ۰/۱ + ۰/۰۱ = ۶/۱۱\text{ms}$$

پاسخ مسئله‌ی ۳.

الف

درست، از آنجایی که در فایل سیستم FAT مکان ذخیره‌سازی فایل‌ها به صورت زنجیره‌ای نگه‌داری می‌شود نمی‌توان دسترسی تصادفی خوب و سریع در آن داشته باش و برای این کار نامناسب است.

ب

درست، می‌توان از روش گفته شده برای ذخیره کردن مکان به صورت سلسه مراتبی استفاده کرد. البته نکته حائز اهمیت آن است که معمولا فایل‌های و فولدرها شامل اطلاعات دیگر است که این روش به تنهایی کافی نبوده است.

ج

نادرست، این دو معادل یکدیگر نبوده است. سکتور واحد فیزیکی کوچکتری است که به حجم ثابتی از داده‌ها در هارددیسک اشاره می‌کند. در حالی که بلوک یک واحد ذخیره‌سازی با حجم متغیر در سطح فایل است و کاربر و وظیقه متفاوتی دارد.

د

نادرست، در اینجا برای فایل سیستم FAT نمی‌توان hard link داشته باشه و دلیل آن به این خاطر است که در این فایل‌سیستم نمی‌توان یک inumber چندین نام مختلف داشته باشد.

ه

نادرست، علاوه بر محدودیت رو حجم اندازه فایل‌های، FAT محدودیت‌هایی در حوزه تعداد پونترهای مستقیم و غیرمسقیم در inode ها قرار می‌دهد.

پاسخ مسئله‌ی ۴.

الف

از آنجایی که در فایل سیستم FAT هستیم و ورودی جدول‌ها ۱۶ بیت هستند، ۱۶ پوینتر داریم که می‌توانیم حداکثر به $۲^{۱۶}$ مکان آدرس‌دهی کنیم که برای دیسک با ۱۳۱۰۷۲ سکتور ۵۱۲ بایتی این مقدار کافی نیست.

ب

در اینجا می‌دانیم که حجم دیسک ما برابر با

$$۱۳۱۰۷۲ \times ۵۱۲ \simeq ۶۴\text{MB}$$

است بنابراین می‌توان با استفاده از یک فایل سیستم در همین فضای FAT با ورودی ۳۲ بیت این مشکل را حل کرده و تمامی فضا را آدرس‌دهی کرد.

پاسخ مسئله‌ی ۵.

الف

از آنجایی که در هر بلاک ۳۲ بایت داریم که فضای اشغالی هر پوینتر ۴ بایت است، می‌توانیم در هر بلاک $۳۲/۴ = ۸$ پوینتر داشته باشیم. حال با توجه به پوینترهای غیرمستقیم یک‌مرحله‌ای و دومرحله‌ای داریم:

$$\implies \overbrace{۲ \times ۸ \times ۳۲}^{\text{indirect index}} + \overbrace{۱ \times ۸ \times ۸ \times ۳۲}^{\text{2-level indirect index}} = ۲۵۶۰$$

پس برای آدرس‌دهی فضای ۳۲۰۰ بایتی به $۶۴۰ = ۲۵۶۰ - ۳۲۰۰$ بایت دیگر نیاز داریم که از آنجایی که هر پوینتر مستقیم ۸ بایت را فضای دهی می‌کند می‌بایست ۸۰ پوینتر مستقیم به سیستم اضافه کنیم.

ب

می‌دانیم که در SSD به خاطر محدودیت حجم نوشتن داده‌ها بر روی خانه‌های حافظه تمایل داریم که تمام خانه‌ها به صورت یکنواخت از طرف سیستم استفاده شود. به این منظور نیاز است که فایل حال موقع استفاده و بازنویسی مجدد در قسمت جدید و کمتر استفاده شده حافظه نوشته شوند و به صورت بلاک‌های بزرگ نیز عمل پاک کردن انجام شود. با توجه به این توضیحات فایل سیستم log-structured می‌تواند با بازنویسی جدید فایل‌های در قسمت‌های کمتر استفاده شده این قابلیت را به ما دهد. در صورتی که در فایل سیستم inode based این کار بسیار سخت‌تر انجام شده و فایل‌های پرتغییر روی بخش خاصی از حافظه اجرا می‌شود و استفاده از حافظه را غیریکنواخت کرده که این مطلوب نیست.

ج

آز آنجایی که در اینجا پنج درایو اصلی و پنج درایو بکاپ از آنها داریم، دو درایو خراب شده می‌تواند دو حالت داشته باشند:

- اگر هر دو از حافظه‌های اصلی باشند، با کپی کردن مقادیر آنها از حافظه بکاپ می‌توانیم به راحتی مشکل را بازیابی کنیم.
- اگر یکی از قسمت اصلی و دیگری از بکاپ باشد و هردو نیز مربوط به یک درایو باشد. (اگر مربوط نباشند که همانند قسمت قبل می‌شود.) در این صورت به خاطر وجود مکانیزم RAID 5 می‌توانیم با XOR های تمام ۴ درایو دیگر اطلاعات این درایو را بازیابی کنیم.

پس در هر صورت امکان ریکاوری کردن فراهم است.

پاسخ مسئله‌ی ۶.

در اینجا می‌خواهیم به سه روش گفته شده بپردازیم:

روش Linked :

در این روش اتفاقی که می‌افتد آن است که هر بلاک به بلاک بعدی اشاره می‌کند و برای پیدا کردن یک آدرس از بلاک بعدی می‌بایست آدرس بلاک فعلی را بخوانیم. در اینجا می‌خواهیم به بلوک فایل دهم برسیم پس نیاز داریم که ۱۰ بار از دیسک عمل خواندن را داشته باشیم.

روش Contiguous :

در این روش تمامی بلاک‌ها پشت سرهم بوده و با دانستن اندازه هر بلاک و آدرس بلاک اول می‌توان به هر بلاک دسترسی پیدا کرد. پس در اینجا برای خواندن بلاک دهم تنها یک خواندن از دیسک نیاز داریم زیراکه با دانستن آدرس بلاک اول می‌توانیم مستقیما به آن دسترسی پیدا کنیم.

روش Indexed :

در این روش می‌دانیم که خود جدول ایندکس‌های در مکانی دیگر ذخیره شده و این امکان را به ما می‌دهد که با یکبار خواندن آن از حافظه آدرس تمام بلاک‌ها را پیدا کنیم و از روی آن بخوانیم. پس در این روش به دو بار خواندن نیاز داریم، یک بار برای خواندن جدول ایندکس‌ها و بار دیگر برای خواندن خود بلاک دهم.