بهینا انصاری راد **به نام خدا**

سوال ۱: وقتی توی شبکه اطلاعات خراب یا نویزی می‌شن، چجوری درستشون می‌کنن؟

وقتی اطلاعات از یه جایی به یه جای دیگه توی شبکه فرستاده می‌شن، ممکنه یه‌سری اتفاقا مثل نویز، تداخل یا خراب شدن کابل باعث بشه بخشی از اون اطلاعات به‌درستی نرسه یا خراب شه. حالا سیستم‌های شبکه برای اینکه مطمئن شن این اطلاعات سالم رسیدن، از یه‌سری ترفند و تکنیک استفاده می‌کنن.

1- اول از همه: چک می‌کنن ببینن اطلاعات سالم هست یا نه (تشخیص خطا)

- **CheckSum:**
  یه عدد خاصی از کل اطلاعات حساب می‌کنن (مثل جمع یه‌سری عدد). این عدد رو هم می‌فرستن همراه اطلاعات. طرف مقابل دوباره همون حسابو می‌کنه، اگه عددش فرق داشت یعنی اطلاعات توی راه خراب شده.

- **Parity Bit (بیت توازن):**
  مثلاً می‌گن باید توی هر گروه از اطلاعات، تعداد ۱ها همیشه زوج باشه. اگه این شرط به‌هم بخوره، می‌فهمن که یه چیزی خراب شده.

- **CRC:**
  یه مدل پیشرفته‌تر از همون چک‌کردنه. خیلی دقیق‌تره و حتی یه ذره تغییر توی اطلاعات رو هم تشخیص می‌ده.

2. بعدش: اگه فهمیدن خراب شده، می‌رن سراغ درست کردنش (اصلاح خطا)

- **کدهای تصحیح خطا (ECC):**
  یه‌سری بیت اضافه همراه داده‌ها می‌فرستن که اگه مشکلی پیش بیاد، خودشون بتونن داده رو اصلاح کنن، بدون اینکه نیاز باشه دوباره از اول بفرستن.

- **ارسال دوباره:**
  اگه نتونستن اصلاحش کنن، یه پیغام می‌فرستن به فرستنده که «هی! اطلاعاتت خراب شده، دوباره

بفرستش!»

3- گاهی هم باید اطلاعات خراب رو تمیز یا بازیابی کنن (Data Cleaning & Restoration)

- **پاک‌سازی داده (Data Cleaning):**
  اطلاعات ناقص یا اشتباه رو پیدا می‌کنن و یا حذف می‌کنن، یا درستش می‌کنن.

- **جای‌گذاری مقادیر معقول (Imputation):**
  اگه یه بخشی از اطلاعات گم شده باشه، یه مقدار منطقی می‌ذارن جاش (مثلاً میانگین بقیه مقادیر).

- **نرمال‌سازی (Normalization):**
  اطلاعات رو هم‌سطح می‌کنن تا همه چیز منظم باشه.

- **بک‌آپ و RAID:**
  اگه اطلاعات خیلی مهم بوده و کلاً خراب شده، میان از نسخه‌ی پشتیبان استفاده می‌کنن. بعضی سیستم‌ها مثل RAID باعث می‌شن حتی اگه یه دیسک خراب بشه، بتونی اطلاعاتو از بقیه در بیاری.

4. گاهی هم مشکل از خود شبکه یا سخت‌افزارهاست

- اگه کابل خراب باشه، یا کارت شبکه مشکل داشته باشه، ممکنه اطلاعات به‌درستی نرسه. اون‌وقت باید برن دنبال عیب‌یابی شبکه.

- یا ممکنه هارد دیسک یا رم سیستم خراب شده باشه که باعث بشه داده‌های ذخیره‌شده هم مشکل‌دار بشن.

2- پلتفرم هاي استريم ويديو به دليل جنس كاربرد، از ارتباط بر بستر UDP استفاده می کنند. أنها این خرابی ها را چگونه مدیریت می کنند؟

درسته که سرویس‌های استریم ویدئو (مثل نتفلیکس، یوتوب، آپارات و...) معمولاً برای ارسال داده‌های ویدئویی از **UDP** استفاده می‌کنند. دلیلش هم اینه که UDP سرعت بالاتری داره و سربارش (overhead) کمتره، که برای استریمینگ که نیاز به جریان پیوسته داده داره خیلی مهم در ادامه بیشتر به این موضوع باز می کنیم.

یکی از اصلی‌ترین راه‌ها، **بافرینگ** هست. وقتی شما یه ویدیو رو پلی می‌کنید، پلیر ویدئو به جای اینکه همون لحظه داده‌ها رو پخش کنه، یه مقدار ازشون رو توی یه حافظه موقت (بافر) ذخیره می‌کنه. این کار چند تا فایده داره:

- **جبران تاخیر:** اگه شبکه برای چند ثانیه دچار مشکل بشه یا بسته‌ای گم بشه، ویدئو از بافر پخش میشه و شما متوجه وقفه‌ای نمی‌شید.
- **صاف کردن جریان:** حتی اگه بسته‌ها کمی نامرتب هم برسن، بافر به پلیر فرصت میده که اون‌ها رو مرتب کنه و بعد پخش کنه.

وقتی این دایره چرخان "بافرینگ" رو می‌بینید، در واقع همون لحظه‌ایه که بافر خالی شده و پلیر داره سعی می‌کنه دوباره یه مقدار داده جمع کنه.

۲. Forward Error Correction (FEC)

یکی دیگه از روش‌های هوشمندانه، **تصحیح خطای پیش‌رو (FEC)** هست. توی این روش، فرستنده (سرور استریم) علاوه بر داده‌های اصلی، یه سری داده‌های اضافی (Redundancy) هم می‌فرسته. این داده‌های اضافی جوری محاسبه میشن که اگه یه سری از بسته‌های اصلی گم بشن، گیرنده (پلیر شما) بتونه با استفاده از اون داده‌های اضافی، بسته‌های گم‌شده رو بازسازی کنه.

تصور کنید یه معلم، علاوه بر جواب اصلی سوال، یه سری سرنخ اضافه هم بده که اگه یه قسمت از جواب اصلی رو نشنیدید، با اون سرنخ‌ها بتونید جواب رو حدس بزنید.

۳. Interleaving

این روش بیشتر توی سیستم‌های صوتی و تصویری استفاده میشه. به جای اینکه داده‌های یک لحظه خاص رو پشت سر هم بفرستن، اون‌ها رو **به هم میبافن (interleave)**. یعنی چی؟ فرض کنید یه ویدئو رو به قطعات

خیلی ریز تقسیم می‌کنن. به جای اینکه قطعه ۱، ۲، ۳، ۴ رو پشت سر هم بفرستن، می‌تونن بفرستن قطعه ۱، ۵، ۹، ۱۳ و بعد ۲، ۶، ۱۰، ۱۴ و...

حالا اگه یه بسته از این‌ها توی مسیر گم بشه، شما فقط یه قسمت خیلی کوچیک از تصویر یا صدا رو از دست میدید و چون بسته‌های اطرافش از زمان‌های دیگه هستن، مغز شما به راحتی اون قسمت رو پر می‌کنه و شما متوجه نمی‌شید. مثل این می‌مونه که یه تیکه کوچیک از پازل رو گم کنید، اما چون بقیه‌ی پازل رو دارید، می‌تونید تصویر کلی رو ببینید.

۴. ABS (Adaptive Bitrate Streaming)

این روش خیلی مهمه و شاید شما بیشتر از همه باهاش سروکار داشته باشید. تو **Adaptive Bitrate Streaming**، سرور ویدئو نسخه‌های مختلفی از یه ویدئو رو با کیفیت‌ها و بیت‌ریت‌های (میزان داده در ثانیه) متفاوت آماده می‌کنه (مثلاً کیفیت ۴۸۰p, 720p, 1080p و...).

پلیر شما مرتباً وضعیت شبکه رو چک می‌کنه. اگه سرعت اینترنتتون خوب باشه، بالاترین کیفیت ممکن رو درخواست می‌کنه. اگه سرعت افت کنه، پلیر به طور خودکار به سرور میگه که کیفیت پایین‌تری از ویدئو رو براش بفرسته. این کار باعث میشه که حتی با اینترنت نوسانی هم استریم بدون قطع و وصل شدن (با کیفیت پایین‌تر) ادامه پیدا کنه.

این سیستم هم کمک می‌کنه تا تأثیر گم شدن بسته‌ها کمتر حس بشه، چون با کاهش کیفیت، حجم داده‌ها کمتر میشه و احتمال رسیدن بسته‌ها بیشتر میشه.

3- نویزها وخرابی های زمان استقرار (اطلاعات در جایی ذخیره شده اند اما ممکن است خراب شوند) چگونه رفع می شود فایل سیستم های توزیع شده چه مزیتی دارند ؟

وقتی اطلاعاتی روی هارد دیسک، SSD یا هر حافظه دیگه‌ای ذخیره میشن، ممکنه به دلایل مختلفی مثل نقص سخت‌افزاری، اشکالات نرم‌افزاری یا حتی خطاهای انسانی، دچار مشکل بشن و اصطلاحاً خراب بشن. برای مقابله با این موضوع، از روش‌های مختلفی استفاده میشه:

1-(Checksums) و (Hashes):

- وقتی داده‌ای ذخیره میشه، یه امضای دیجیتال (checksum یا hash) براش محاسبه و همراه با خود داده ذخیره میشه.
- بعداً وقتی داده خونده میشه، دوباره امضاش محاسبه میشه و با امضای اولیه مقایسه میشه.
- اگر این دوتا با هم فرق کنن، یعنی داده خراب شده. این روش بیشتر برای **تشخیص** خرابی کاربرد داره.

2- تصحیح خطای پیش‌رو (Forward Error Correction - FEC) و کدهای افزونگی (Redundancy Codes):

- در این روش‌ها، علاوه بر داده اصلی، یه سری اطلاعات اضافی و بازسازی‌کننده (Parity Data) هم ذخیره میشه.
- اگر بخشی از داده‌ها (یا حتی یک هارد دیسک کامل) خراب بشن یا از بین برن، میشه با استفاده از این اطلاعات اضافی، داده‌های اصلی رو بازسازی کرد.
- RAID (Redundant Array of Independent Disks) یک مثال معروفه که چندین هارد دیسک رو ترکیب می‌کنه و با استفاده از Parity Data، حتی با از دست رفتن یک یا چند دیسک هم امکان بازیابی اطلاعات رو فراهم می‌کنه.
- Erasure Coding هم روش پیشرفته‌تریه که با ذخیره اطلاعات اضافی کمتر، مقاومت بالاتری در برابر از دست رفتن چندین بخش از داده رو فراهم می‌کنه.

3-(Data Scrubbing):

- این یک فرایند دوره‌ای هست که در پس‌زمینه اجرا میشه و تمام داده‌های ذخیره‌شده رو اسکن می‌کنه.
- هدفش پیدا کردن و ترمیم خرابی‌های کوچیک و پنهان (Silent Data Corruption) قبل از اینکه به مشکل جدی تبدیل بشن، هست.
- این کار با استفاده از چکسام‌ها و کدهای افزونگی انجام میشه تا در صورت تشخیص خرابی، داده‌ها تعمیر بشن.

پشتیبان‌گیری (Backup) و اسنپ‌شات (Snapshots):

- **پشتیبان‌گیری:** گرفتن کپی از داده‌ها و ذخیره اون‌ها در یک مکان جداگانه و امن. این روش در صورت بروز فجایع بزرگ (مثل حمله باج‌افزار یا خرابی کامل سیستم) راهی برای بازیابی کامل اطلاعاته.
- **اسنپ‌شات:** یک "تصویر لحظه‌ای" از وضعیت فایل سیستم در یک نقطه زمانی خاصه. اینها برای بازیابی سریع به نسخه‌های قبلی فایل‌ها (مثلاً اگه فایلی به اشتباه پاک یا ویرایش شد) خیلی مفید هستن.

فایل سیستم‌های توزیع‌شده (Distributed File Systems - DFS) چه مزیتی در این مورد دارن؟

فایل سیستم‌های توزیع‌شده (مثل HDFS, GlusterFS, Ceph) ذاتاً برای کار با حجم عظیمی از داده‌ها و تحمل خطا طراحی شدن و در برابر خراب شدن داده‌ها مزایای قابل توجهی دارن:

1- تکرار داده (Data Replication) :

- مهمترین مزیت اینه که DFSها هر قطعه از داده رو به طور پیش‌فرض روی چندین سرور (نود) مختلف کپی می‌کنن. مثلاً اگه تنظیم شده باشه که هر داده ۳ بار تکرار بشه، کپی اون روی ۳ سرور جداگانه ذخیره میشه.
- مزیت: اگر یک یا حتی دو سرور کاملاً از کار بیفتن یا داده‌هاشون خراب بشن، هنوز نسخه‌های سالم داده روی سرورهای دیگه وجود داره و سرویس‌دهی بدون وقفه ادامه پیدا می‌کنه. این ویژگی به قابلیت دسترسی بالا (High Availability) و تحمل خطا (Fault Tolerance) کمک می‌کنه.

2- استفاده از Erasure Coding برای بهره‌وری بیشتر:

- بسیاری از DFSهای مدرن از Erasure Coding استفاده می‌کنن. این بهشون اجازه میده با ذخیره اطلاعات افزونگی کمتر نسبت به روش صرفاً تکرار، به همون سطح از تحمل خطا دست پیدا کنن.
- **مزیت:** فضای ذخیره‌سازی کمتری مصرف میشه و در عین حال، در برابر از دست دادن چندین دیسک یا نود مقاومت بالایی دارن.

ترمیم خودکار (Self-Healing):

- فایل سیستم‌های توزیع‌شده معمولاً مکانیزم‌های خودکار برای تشخیص و ترمیم خرابی دارن.
- وقتی یک نود یا یک قطعه داده‌ای خراب میشه، سیستم به طور خودکار نسخه‌های سالم اون داده رو از نودهای دیگه پیدا می‌کنه و روی نودهای سالم جدید (که جای نود خراب رو گرفتن) دوباره تکرار می‌کنه تا تعداد نسخه‌های مورد نظر حفظ بشه.
- **مزیت:** نیاز به دخالت دستی مدیر سیستم کمتر میشه و سیستم به صورت خودکار خودش رو از مشکلات بازیابی می‌کنه.

مقیاس‌پذیری (Scalability):

- اگر حجم داده‌ها زیاد بشه یا نیاز به مقاومت بیشتری در برابر خطا داشته باشیم، به راحتی میشه سرورهای جدیدی رو به این سیستم‌ها اضافه کرد.
- **مزیت:** سیستم انعطاف‌پذیره و می‌تونه با رشد داده‌ها و نیازهای کسب‌وکار تطبیق پیدا کنه.

4- با توجه به اطلاعاتی که درباره نویز و پاک‌سازی نویز (denoise) داری، به طور میانگین چند درصد از اطلاعاتی که می‌فرستیم قابل بازیابی نیست؟ و چند درصد از خطاها اصلاً قابل تشخیص نیستن؟

چقدر از اطلاعات غیر قابل بازیابی هستن؟

سیستم از یه روش پیشرفته به اسم کد اصلاح خطای رید-سالامون (Reed-Solomon Error Correction) استفاده می‌کنه. این روش مثل این می‌مونه که وقتی اطلاعات رو می‌فرستیم، یه سری اطلاعات اضافه هم کنارشون می‌فرستیم تا اگه مشکلی پیش اومد، بشه اطلاعات اصلی رو بازسازی کرد.

پارامترهای این سیستم به این صورته:

- حداکثر خطای قابل اصلاح: هر بلاک (بسته) اطلاعات می‌تونه تا ۱۰۰ "سیمبل" (معادل ۸۰۰ بیت) خطا رو اصلاح کنه.
- بلاک‌ها چطورن؟ هر بلاک شامل ۵۵ سیمبل داده اصلی و ۲۰۰ سیمبل اطلاعات اضافی (برای اصلاح خطا) هست.

معنیش چیه؟ اگه تعداد خطاهای توی یه بلاک بیشتر از ۱۰۰ سیمبل باشه، اون بلاک اطلاعاتی دیگه قابل بازیابی نیست و اصطلاحاً خراب میشه.

از نتایج آزمایش‌ها (با ۱ درصد نویز در خط انتقال) مشخص شده که:

- سیستم تونسته از ۲۹۹۵ بیت خطا با موفقیت بازیابی کنه.
- تمام خطاها (۲۸۷۳ خطا) تشخیص داده شدن و اصلاح شدن.
- یعنی در این شرایط هیچ داده‌ای از دست نرفته (۰٪ از اطلاعات غیرقابل بازیابی بوده).

چند درصد از خطاها غیر قابل تشخیص هستن؟

سیستم رید-سالامون مکانیزم‌های قوی برای تشخیص خطا داره. یعنی معمولاً می‌فهمه که کی یه خطا اتفاق افتاده.

چه خطاهایی رو تشخیص میده؟

- وقتی تعداد خطاها کمتر یا مساوی ۱۰۰ سیمبل باشه، حتماً تشخیص داده میشن.
- وقتی جای خطاها مشخص باشه (مثلاً اگه سیستم بدونه دقیقاً کجا مشکل پیش اومده)، باز هم تشخیص داده میشن.

اما کی ممکنه خطاها تشخیص داده نشن (که خیلی نادر هستن)؟

- وقتی تعداد خطاها خیلی زیاد باشه و از ۲۰۰ سیمبل هم بیشتر بشه. در این حالت، ممکنه سیستم دیگه نتونه خطا رو تشخیص بده.
- خیلی خیلی به ندرت، ممکنه الگوی خطا جوری باشه که سیستم فکر کنه اطلاعات سالمه، در حالی که خراب شده (به این میگن "valid codeword"). این اتفاق فوق‌العاده نادره.

عملکرد کلی سیستم و خلاصه‌ی نهایی:

- سیستم برای اصلاح خطاها، حدود ۷۸٪ اطلاعات اضافی ارسال می‌کنه (یعنی ۲۰۰ سیمبل از ۲۵۵ سیمبل هر بلاک). این حجم اطلاعات اضافی زیادیه، اما برای اینکه مطمئن باشیم داده‌ها سالم می‌رسن، لازمه.
- سیستم می‌تونه تا ۱۰۰ سیمبل (۸۰۰ بیت) خطا رو در هر بلاک تشخیص بده و اصلاح کنه.
- با توجه به آزمایش‌ها، با ۱ درصد نویز در خط انتقال، سیستم به خوبی و با قابلیت اطمینان بالا کار می‌کنه.
- درصد اطلاعات غیرقابل بازیابی برای سطوح نویز تا ۱ درصد، خیلی کمه (نزدیک به صفر درصد) هست.
- خطاهای غیرقابل تشخیص هم به خاطر مکانیزم‌های قوی تشخیص خطا، فوق‌العاده نادر هستن.
- به طور خلاصه، سیستم بین قابلیت اصلاح خطا و حجم اطلاعات اضافی که باید فرستاده بشه، تعادل خوبی برقرار کرده تا انتقال داده‌ها قابل اطمینان باشه.

5-آیا سیستم فایل‌های توزیع‌شده می‌تونن جلوی همه نوع خرابی داده رو بگیرن؟ نقش نسخه‌های پشتیبان (Replica) توی این سیستم‌ها چیه؟ اصلاً به چه دردی می‌خورن؟

خرابی‌های منطقی (Logical Corruption):

- تعریف: این نوع خرابی به دلیل خطای نرم‌افزاری در برنامه کاربردی، خطای انسانی (مثلاً پاک کردن یا تغییر ناخواسته فایل توسط کاربر) یا حمله بدافزار (مثل باج‌افزار) رخ میده. داده‌ها از نظر فیزیکی روی دیسک سالم هستن، اما محتواشون غلط یا ناقص شده.

- وضعیت در DFS: سیستم‌های فایل توزیع‌شده با استفاده از کپی‌های متعدد (Replica) یا Erasure Coding، در برابر خرابی فیزیکی دیسک‌ها یا از دست رفتن نودها مقاوم هستن. اما اگر یک فایل خراب (مثلاً توسط بدافزار) روی یک نود ذخیره بشه، DFS این نسخه خراب رو روی بقیه نودها هم کپی می‌کنه. در واقع، DFS تضمین می‌کنه که نسخه‌های کپی شده "دقیقاً یکسان" باشن، نه اینکه "درست" باشن.

- نتیجه: DFS نمی‌تونه به تنهایی جلوی این نوع خرابی‌ها رو بگیره، چون هدف اصلیش حفظ دسترس‌پذیری و یکپارچگی فیزیکی داده‌هاست، نه محتوای منطقی اون‌ها.

حذف تصادفی یا مخرب:

- تعریف: کاربر فایلی رو به اشتباه حذف می‌کنه یا یک مهاجم اطلاعاتی رو پاک می‌کنه.
- وضعیت در DFS: اگر یک فایل از طریق رابط کاربری DFS (مثل دستور `delete`) حذف بشه، DFS همونطور که در Replication گفته شد، مطمئن میشه که از روی تمام کپی‌ها پاک بشه. پس بازم DFS به تنهایی اینجا کمکی نمی‌کنه.

خب بریم سراغ قسمت دوم سوال, نقش نسخه‌های پشتیبان (Replica) توی این سیستم‌ها چیه؟ اصلاً به چه دردی می‌خورن؟

تحمل خطا (Fault Tolerance):

- اصل کار: مهم‌ترین دلیل وجود Replicas. هر قطعه از داده (مثلاً هر بلاک در HDFS) روی چندین نود/سرور مختلف ذخیره میشه (مثلاً ۳ کپی).
- کاربرد: اگر یکی از سرورها از کار بیفته (هاردش خراب بشه، نودش خاموش بشه، برقش بره و...)، سیستم می‌تونه از کپی‌های موجود روی سرورهای دیگه استفاده کنه و سرویس‌دهی بدون وقفه ادامه پیدا می‌کنه.
- مزیت: در برابر خرابی‌های سخت‌افزاری (مثل نقص دیسک، خرابی RAM، مشکلات شبکه) و از دست رفتن نودها بسیار مقاوم هستن. این هسته اصلی فلسفه DFS برای اطمینان از دردسترس‌بودن همیشگی داده‌هاست.

قابلیت دسترسی بالا (High Availability):

- اصل کار: چون چندین کپی از داده‌ها وجود داره، همیشه حداقل یک کپی قابل دسترس هست، حتی اگر برخی نودها در دسترس نباشن.
- کاربرد: اطمینان از اینکه برنامه‌ها و کاربران می‌تونن در هر لحظه به داده‌ها دسترسی داشته باشن و سیستم دچار قطعی (Downtime) نشه.

- مزیت: سرویس‌دهی مداوم و بدون وقفه، که برای کسب‌وکارها حیاتیه.

توزیع بار (Load Balancing):

- اصل کار: وقتی چندین کپی از یک فایل وجود داره، درخواست‌های خواندن (Read Requests) می‌تونن بین نودهایی که اون کپی‌ها رو دارن، توزیع بشن.
- کاربرد: به جای اینکه یک نود خاص همیشه مسئول پاسخگویی به درخواست‌های مربوط به یک فایل خاص باشه، بار کاری بین چندین نود پخش میشه.
- مزیت: افزایش کارایی و سرعت دسترسی به داده‌ها، چون هر نود فشار کمتری رو تحمل می‌کنه و تأخیر کاهش پیدا می‌کنه.

اسکرابینگ داده (Data Scrubbing) و خودترمیم‌گری (Self-Healing):

- اصل کار: DFS ها معمولاً به صورت دوره‌ای (مثلاً هفتگی) تمام کپی‌های داده رو چک می‌کنن (با استفاده از چکسام‌ها). اگر یک کپی خراب شده باشه (مثلاً Silent Data Corruption)، سیستم به صورت خودکار کپی‌های سالم رو پیدا می‌کنه و اون کپی خراب رو با یک نسخه سالم بازسازی می‌کنه. همچنین، اگر یک نود از کلاستر خارج بشه، سیستم به صورت خودکار کپی‌هایی که روی اون نود بودن رو روی نودهای سالم دیگه بازسازی می‌کنه تا تعداد کپی‌های لازم حفظ بشه.
- کاربرد: پیدا کردن و تعمیر خرابی‌های پنهان داده‌ها قبل از اینکه به مشکل بزرگ‌تری تبدیل بشن و همچنین حفظ تعداد کپی‌ها در صورت از دست رفتن نودها.
- مزیت: نگهداری و ترمیم خودکار سیستم، کاهش نیاز به دخالت دستی مدیر سیستم.

۶-چگونه می توان با استفاده از فایل سیستم های توزیع شده گلوگاه را حذف کرد؟

گلوگاه I/O دیسک (Disk I/O Bottleneck):

- توضیح: یک هارد دیسک یا SSD واحد، محدودیت خاصی در سرعت خواندن و نوشتن (Input/Output Operations Per Second - IOPS و Throughput) دارد. اگر تعداد زیادی درخواست همزمان به این دیسک برسد، دیسک نمی‌تواند پاسخگوی همه باشد و عملیات‌ها کند می‌شوند.
- مثال: فرض کنید یک اپلیکیشن نیاز به خواندن و نوشتن هزاران فایل کوچک در ثانیه دارد، اما هارد دیسک فقط می‌تواند چند صد عملیات را انجام دهد.

گلوگاه CPU/حافظه (CPU/Memory Bottleneck):

- توضیح: یک سرور تنها، محدودیت‌هایی در قدرت پردازش (CPU) و میزان حافظه (RAM) دارد. اگر برنامه‌ای به منابع CPU یا RAM زیادی نیاز داشته باشد، یا تعداد کاربران همزمان زیاد شود، سرور دچار مشکل می‌شود.

گلوگاه شبکه (Network Bottleneck):

- توضیح: سرعت کارت شبکه (NIC) و پهنای باند شبکه بین سرور و کلاینت‌ها محدودیت دارد. اگر حجم زیادی از داده‌ها باید از طریق شبکه منتقل شوند، شبکه می‌تواند به گلوگاه تبدیل شود.
- مثال: در یک سرور فایل سنتی، اگر ۱۰۰ کاربر همزمان بخواهند یک فایل بزرگ را دانلود کنند، پهنای باند شبکه سرور می‌تواند محدود کننده باشد.

چطور فایل سیستم‌های توزیع‌شده (DFS) گلوگاه‌ها را حذف می‌کنند؟

DFSها با رویکرد "مقیاس‌پذیری افقی" (Horizontal Scaling) این گلوگاه‌ها را از بین می‌برند. به جای اینکه یک سرور را قدرتمندتر کنیم (Scale Up)، چندین سرور (Node) معمولی را به هم متصل می‌کنیم تا به صورت یک سیستم واحد کار کنند.

۱- حذف گلوگاه I/O دیسک با توزیع داده‌ها (Data Distribution & Parallel I/O):

روش کار: DFSها فایل‌های بزرگ را به قطعات کوچک‌تر (Blocks) تقسیم می‌کنند و این قطعات را روی دیسک‌های مختلف در سرورهای مختلف (نودهای داده) پخش می‌کنند. همچنین، از مکانیزم تکرار داده (Replication) استفاده می‌کنند، یعنی هر قطعه از داده را روی چندین دیسک در نودهای متفاوت ذخیره می‌کنند.

چطور گلوگاه را حذف می‌کند؟

- I/O موازی: وقتی یک برنامه نیاز به خواندن یک فایل بزرگ دارد، می‌تواند قطعات مختلف آن فایل را به صورت همزمان از دیسک‌های مختلف روی نودهای متفاوت بخواند. این یعنی به جای اینکه یک دیسک به تنهایی بار I/O را تحمل کند، بار بین ده‌ها یا صدها دیسک توزیع می‌شود و سرعت کلی خواندن و نوشتن بسیار بالا می‌رود.
- از بین بردن نقطه واحد شکست: اگر یک دیسک یا یک نود خراب شود، چون کپی داده‌ها روی دیسک‌های دیگر وجود دارد، سیستم بدون وقفه به کار خود ادامه می‌ده

۲- حذف گلوگاه CPU/حافظه با توزیع پردازش (Distributed Processing & Resource Aggregation):

- **روش کار:** در DFSها، فقط داده‌ها توزیع نمی‌شوند، بلکه عملیات پردازشی هم می‌توانند نزدیک به جایی که داده‌ها ذخیره شده‌اند (Data Locality) انجام شوند. همچنین، منابع CPU و RAM همه نودها با هم جمع می‌شوند.

چطور گلوگاه را حذف می‌کند؟

- پردازش موازی: به جای اینکه یک سرور واحد تمام پردازش‌ها را انجام دهد، هر نود بخشی از پردازش را روی داده‌هایی که روی خودش ذخیره شده‌اند، انجام می‌دهد. این یعنی قدرت پردازش (CPU) و ظرفیت حافظه (RAM) سیستم، با اضافه شدن هر نود جدید، افزایش می‌یابد.
- کاهش فشار روی یک نقطه: هیچ نود واحدی مسئول تمام کارها نیست، بار پردازشی به صورت هوشمندانه بین نودها توزیع می‌شود.

۳- حذف گلوگاه شبکه با افزایش پهنای باند تجمعی (Aggregate Network Bandwidth):

- **روش کار:** DFSها به جای اینکه فقط از یک کارت شبکه و لینک شبکه واحد استفاده کنند، از شبکه‌های متعدد و پرسرعت بین نودها بهره می‌برند. هر نود دارای کارت شبکه مستقل خود است.

چطور گلوگاه را حذف می‌کند؟

- پهنای باند تجمیعی: وقتی داده‌ها از نودهای مختلف خوانده می‌شوند، از پهنای باند شبکه کلی کلاستر (خوشه) استفاده می‌شود، نه فقط پهنای باند یک نود. این به معنای پهنای باند بسیار بیشتر برای انتقال داده‌هاست.
- کاهش ترافیک شبکه (Data Locality): با انجام پردازش‌ها در همان نودی که داده‌ها قرار دارند، نیاز به جابجایی داده‌های حجیم از طریق شبکه کاهش می‌یابد. این باعث می‌شود که فشار کمتری روی شبکه بیاید و گلوگاه‌های شبکه کمتر اتفاق بیفتند.

۶-cdn ها یا شبکه های تحویل محتوا را با فایل سیستم های توزیع شده مقایسه کنید. این شبکه ها چگونه می توانند مدت زمان دسترسی کاربر به محتوا را کاهش دهند و فایل سیستم های توزیع شده چه نقشی در این بین دارند ؟

CDNها چگونه مدت زمان دسترسی کاربر به محتوا را کاهش می‌دهند؟

CDNها برای کاهش مدت زمان دسترسی (Latency) کاربر به محتوا، بر اساس دو اصل اصلی کار می‌کنند:

کاهش فاصله فیزیکی:

- سرورهای لبه (Edge Servers) یا نقاط حضور (CDN :(PoPsها شبکه‌ای از سرورها را در نقاط جغرافیایی استراتژیک در سراسر جهان مستقر می‌کنند. این سرورها به کاربر نهایی نزدیک‌تر هستند تا سرور اصلی (Origin Server) که محتوای اصلی روی آن قرار دارد.
- مسیریابی هوشمند: وقتی کاربر درخواست محتوا می‌دهد (مثلاً باز کردن یک وب‌سایت)، درخواست او به نزدیک‌ترین سرور CDN از نظر جغرافیایی هدایت می‌شود. این سرور محتوا را به جای سرور اصلی که ممکن است هزاران کیلومتر دورتر باشد، به کاربر تحویل می‌دهد.
- نتیجه: با کاهش مسافت فیزیکی، زمان رفت و برگشت درخواست (Round-Trip Time - RTT) که به آن تأخیر (Latency) می‌گویند، به حداقل می‌رسد و کاربر محتوا را با سرعت بسیار بالاتری دریافت می‌کند.

کشینگ (Caching):

- ذخیره‌سازی موقت: سرورهای لبه CDN، کپی‌هایی از محتوای پرکاربرد (مانند تصاویر، ویدئوها، فایل‌های CSS و JavaScript) را به صورت موقت در حافظه خود ذخیره می‌کنند (کش می‌کنند).
- تحویل فوری: وقتی کاربر درخواست همان محتوا را می‌دهد، اگر آن محتوا در کش سرور لبه موجود باشد، فوراً از همانجا به کاربر تحویل داده می‌شود و نیازی به رفتن به سرور اصلی نیست.
- نتیجه: این کار باعث می‌شود بار روی سرور اصلی (Origin Server) هم کاهش یابد و تجربه کاربری بهتری برای کاربران فراهم شود.

علاوه بر این‌ها، CDNها از تکنیک‌های دیگری هم استفاده می‌کنند:

- فشرده‌سازی فایل‌ها: کاهش حجم فایل‌ها (مثل Gzip) برای انتقال سریع‌تر.
- بهینه‌سازی پروتکل‌ها: استفاده از پروتکل‌های جدیدتر و بهینه‌تر مانند HTTP/2 و HTTP/3.
- توزیع بار (Load Balancing): پخش درخواست‌ها بین سرورهای CDN برای جلوگیری از شلوغی یک سرور خاص.
- امنیت: بسیاری از CDNها قابلیت‌های امنیتی مانند محافظت در برابر حملات DDoS را نیز ارائه می‌دهند.

فایل سیستم‌های توزیع‌شده (DFS) چه نقشی در این بین دارند؟

فایل سیستم‌های توزیع‌شده به طور مستقیم برای کاهش مدت زمان دسترسی کاربر نهایی به محتوا طراحی نشده‌اند، بلکه وظیفه اصلیشان ذخیره‌سازی و مدیریت قابل اعتماد و مقیاس‌پذیر داده‌ها در پشت صحنه (Backend) است.

نقش DFS در اکوسیستم CDN:

DFSها می‌توانند به عنوان سرور اصلی (Origin Storage) برای CDNها عمل کنند. یعنی:

1. پایگاه داده مرکزی و مطمئن: سرور اصلی که CDN از آن محتوا را دریافت می‌کند (برای کش کردن یا برای محتوایی که در کش نیست)، خودش می‌تواند یک فایل سیستم توزیع‌شده باشد.
2. ذخیره‌سازی مقیاس‌پذیر برای CDN: فرض کنید شما یک پلتفرم ویدیویی بزرگ هستید (مثل نتفلیکس یا آپارات). حجم ویدئوهای شما بسیار زیاد است و نیاز به یک سیستم ذخیره‌سازی بسیار مقاوم، پرسرعت و مقیاس‌پذیر دارید تا این ویدئوها را ذخیره کنید. یک DFS (مثل HDFS یا Ceph) می‌تواند این نیاز را برطرف کند.
3. تغذیه محتوا به CDN: وقتی CDN نیاز به کش کردن یک ویدئوی جدید دارد یا کاربری درخواست محتوایی را می‌دهد که در کش سرور لبه CDN نیست، CDN به سرور اصلی (که می‌تواند یک DFS باشد) مراجعه می‌کند تا محتوا را دریافت کند.
4. پشتیبانی از پردازش‌های پشت صحنه: DFSها همچنین برای پردازش‌های بزرگ داده (Big Data Analytics) که روی محتواهای شما انجام می‌شود (مثلاً برای تحلیل رفتار کاربران، شخصی‌سازی محتوا و...) نقش حیاتی دارند. این پردازش‌ها روی داده‌های خام و اصلی که در DFS ذخیره شده‌اند، انجام می‌شوند.

مثال:

تصور کنید یک شرکت بازی‌سازی دارید. فایل‌های بازی‌ها (مثل بافت‌ها، مدل‌ها، کدها) حجم بسیار بالایی دارند. این فایل‌ها در ابتدا روی یک فایل سیستم توزیع‌شده (DFS) مثل HDFS ذخیره می‌شوند تا مقاومت در برابر خطا و مقیاس‌پذیری لازم برای توسعه‌دهندگان و فرآیندهای داخلی شرکت فراهم شود.

وقتی قرار است این بازی‌ها به دست کاربران در سراسر دنیا برسد، فایل‌های بازی (یا آپدیت‌های آن) روی CDN بارگذاری می‌شوند. حالا کاربری در آمریکا که می‌خواهد بازی را دانلود کند، آن را از نزدیک‌ترین سرور CDN در آمریکا دریافت می‌کند، نه از سرور اصلی شرکت که ممکن است در آلمان باشد. این کاهش فاصله و استفاده از کش CDN، سرعت دانلود را برای کاربر به شدت بالا می‌برد