سیستم‌عامل پیشرفته

محمد داورپناه جزی

گروه مهندسی کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی فولاد

ترم دوم ۹۴-۹۳

فصل پنجم: سیستم فایل توزیع‌شده

- عنصر کلیدی سیستم توزیع‌شده، سیستم فایل آن است.
- در اینجا از شباهت‌ها به سیستم متمرکز صرف‌نظر کرده و تفاوتها را ذکر می‌کنیم.
- سرویس‌های فایل یعنی قابلیت‌های سیستم و نحوهٔ استفاده از آنها (پارامترها و ....).
- سرویس‌ها **WHAT** را می‌گویند نه **How** (چگونگی پیاده‌سازی) و توسط **File Server** پیاده می‌شود.
- تعداد، محل و سرویس‌های **File Server** باید از دید **User** مخفی (**Transparent**) باشد.
- تعداد و سرویس‌های **Server**ها به سیستم وابسته است.
- مثال یک ماشین با سیستم فایل **MSDOS** و **UNIX** روی دو **Server**، یک ترمینال با دو جور پنجره روی آن.

فصل پنجم: سیستم فایل توزیع‌شده

بخش ۱ طراحی و سیستم فایل توزیع‌شده

- دو عنصر اصلی: عملیات اصلی فایل شامل خواندن، نوشتنن، .....
- : عملیات Directory شامل افزودن و حذف، .....
- فایل یعنی چه: در MSDOS, UNIX شامل دنباله‌ای از بایت‌ها (کاراکترها)ست که از دید برنامهٔ استفاده کننده آن معنی و فرمت خاصی دارد و سیستم عامل دخالتی ندارد.
- در Main Frame: معمولاً انواع فایل وجود دارد، دنباله‌ای از رکوردها با خواندن و نوشتن مخصوص.
- رجوع به رکوردها با آدرس واقعی، hash، ایندکس.
- در سیستم توزیع شده بیشتر فایل بعنوان مجموعه بایت‌ها استفاده می‌شود.

فصل پنجم: سیستم فایل توزیع‌شده

- هر فایل دارای attributeهائی است استاندارد مثال: صاحب، تاریخ، سایز، مجوزها.
- فایل immutable را می‌توان بعداز CREATE فقط READ کرد.
- فایل immutable براحتی Cache و Replicate می‌شود.
- ایمنی با روش لیست قابلیت‌ها و لیست دستیابی‌هاست. رک OSI.
- مثال روش Access can Frul List در UNIX .....
- در مدل: Upload-download و Remote Access شکل ۱-۵ ص ۲۴۷.
- در ULDL کل فایل منتقل شده و بطور محلی استفاده می گردد. پیاده‌سازی راحت دو عمل WAR RD، نیاز به فضای کافی، مقداری کار اضافی.
- در RA تعداد زیادی سرویس لازم است. لیست در ص ۲۴۸، عدم نیاز به فضای زیاد و انتقال بی‌مورد.

فصل پنجم: سیستم فایل توزیع‌شده

- سرویس‌های Directory = نامگذاری، جابجای، دایرکتوری تودرتو
- نامگذاری ext، اسم، بعضی نوع با ext است بعضی نوع با # Magic.
- سیستم درخت و گراف شکل ۵-۲ ص ۲۴۹، خط سیستم گراف در حذف A ⟵ B.
- سیستم‌ها در سیستم متمرکز با توقف عملیات و بررسی سیستم امکان دارد ولی در سیستم توزیع‌شده بسیار مشکل است.
- آیا همه ماشین‌ها باید WIEW یکسان از Directory داشته باشند یا نه؟ شکل ۳-۵ ص ۲۵.
- در b دیوها یکی است در c فرق می کند D/E/X/ در ایندو فرق می کند.
- اگر بخواهم Root یکسان برای همه ماشین‌ها داشته باشیم مسیرها باید با Server/path/ شروع شود.

فصل پنجم: سیستم فایل توزیع‌شده

- شفافیت نام
- اولین نوع شفافیت محل است یعنی Path راهنمای محل فیزیکی نباشد.
- /IUI/ECEDPT/B321 خوب نیست /Server1/dir1/Proj1 خوب است.
- تغییر Server بدلیل کمبود جا روی Server1 و وجود جا روی Server2 ایجاد مشکل می کند.
- سیستمی که چنین مشکلی ندارد ..... یا مستقل از محل است ولی پیاده‌سازی مشکل است.
- سه برخورد اصلی در سیستم فایل و دایرکتوری ص ۲۵۱.
- اولی و دومی فایل دستیابی است ولی سومی نیاز به طراحی دقیق سیستم فایل دارد.

فصل پنجم: سیستم فایل توزیع‌شده

- نامگذاری در سطحی
- نام سمبولیک برای کاربر و نام عددی (بیت) برای سیستم، **Mapping** توسط سیستم **dir**.
- با **Server**های مستقل، اسم باینری در هر کدام برای خودش بامعنی است **i-node** در یوتیکس.
- روش کلی‌تر داشتن اسم **Server** و سپس اسم **dir** (یک **Server** می‌تواند به **Server** دیگر رجوع کند).
- راه دیگر استفاده از اسم فیزیکی یا منطقی ماشین‌ها روی شکبه است به روش انتقال پیام یا **Broad Cast**.
- نکته دیگر **map** شدن چند اسم عددی به رسم سمبولیک است که شامل فایل و **Bk up**هایش است بعنوان تحمل‌پذیری خطا خوب است.

فصل پنجم: سیستم فایل توزیع‌شده

- مفاهیم اشتراک فایل
- مثال UNIX Semantics شکل ۵-۴a ص ۲۵۳ هر Read آخرین نوشتن را برمی گرداند.
- در سیستم توزیعی با یک Server بدون File Caching این مدل می‌تواند وجود داشته باشد.
- انجام درخواستهای خواندن و نوشتن به ترتیب دقیق زمانی.
- در عمل حداقل Caching وجود دارد و ایجاد مشکل می کند شکل ۵-۴-b ص ۲۵۳.
- می‌توان زندگی را راحت گرفت و گفت "تغییر کرده که کرده" فقط تغییر کننده بداند و بس تا اینکه فایل Close شود، "بعد بقیه بدانند" به این Session Semantics گویند.
- اگر دو تا همزمان تغییر دهند در Cache، آنکه دیرتر Close کرد نهائی است!
- راحت‌تر اینکه "ول کنیم هر کدام که شد" اینجا UNIX Sem رعایت نشده.

فصل پنجم: سیستم فایل توزیع‌شده

- در UNIX Semهر فایل یک دارد که محل خواندن از یا نوشتن در است برای هر پروسس و فرزندانش.
- در Session Se فرزندان روی ماشین دیگر است این PTR نمی‌تواند Share شود.
- مثال Run>Out< که Run در برنامه a بعد b را اجرا کند UNIX خروجی b. بلافاصله بعد از خروجی a است چون هر دوی a، b فرزندان Runاند و PTR را به ارسال می‌برند.
- در مورد Session Se این کار بسیار پیاده‌سازی‌اش مشکل است و هر سیستمی بطریقی آنرا حل کردن Share
- راه دیگر: همه فایل‌ها Immutableاند فقط Create و Read.
- تغییر نداریم ولی می‌توان فایل را خواند و از رویش فایل جدیدی ساخت که جانشین قبلی می‌شود.
- پس مشکل نوشتن روی فایلی که خوانده شده نداریم.

فصل پنجم: سیستم فایل توزیع‌شده

- مشکلی که داریم جایگزین همزمان یک فایل توسط دو پروسس، آخری نهائی است یا شانسی باشد.
- مشکلی دیگر جایگزینی فایلی که کسی دارد آنرا می‌خواند، به نحوی بگذاریم خواندن ادامه یابد حتی اگر از dir حذف شد.
- مثل UXIN: فایل open را می گذارد و استفاده شود ولو اینکه از کلید dirها حذف شده باشد.
- یا اینکه خواندن را fail کنیم.
- راه چهارم هم Atomic Xachin است که در فصل ۳ گفته شد End Trans عملیات Btrans All or Nothing ترتیب پذیری.
- مثال اضافه کردن ۵. دلار توسط دو پروسس به یک حساب حاوی ۱۰۰دلار بدون ترتیب ممکن است ۱۲۰ یا ۱۱۵ دلار شود ولی با ترتیب حتماً ۱۲۰ دلار می‌شود.
- خلاصه در شکل ۵-۵ ص ۲۵۶.

فصل پنجم: سیستم فایل توزیع‌شده

بخش ۲ پیاده‌سازی سیستم فایل توزیع‌شده

- بحث اینجا نحوهٔ پیاده‌سازی است از دید پیاده‌ساز نه از دید کاربر.
- چگونگی پیاده کردن Caching، تکرار، کنترل همزمانی، .....

- بکارگیری فایل
  - اطمینان از اینکه پراستفاده‌ترین عملیات روی فایل دارای کارآئی بالاست.
  - سنجش‌های استاتیک: اندازه فایل، توزیع اندازه‌ها روی سطح دیسک، توزیع انواع روی سطح دیسک.
  - سنجش‌های دینامیک: از روی Log ساخته شده استخراج می‌شود.
  - بافر انواع عمل‌ها.
  - تعداد فایل‌های باز در هر لحظه.
  - حجم اشتراک در عملیات روی فایل‌ها.

فصل پنجم: سیستم فایل توزیع‌شده

- تلفیق دو سنجش می‌تواند برای بکار گیری سیستم فایل راهنما باشد.
- مسئله دیگر نوع کاربران است تا چه حد آنان نماینده همهٔ کاربرانند.
- در تحقیق مورد بحث کاربران دانشگاه هستند، آیا محققان صنعتی، کاربران اداری و ... هم اینگونه‌اند.
- کسی نمی‌داند تا وقتی که هر سیستمی سنجیده شود.
- مطلب دیگر آثار جانبی سیستم مورد استفاده است مثلاً حداکثر حدود ۸ کاراکتر در اسم فایل.
- اجباراً ۸ تا می گیریم، اگر آزاد بود معلوم نیست کاربران چه می کردند.
- تحقیق کتاب روی UNIX معمولی بوده آیا روی سیستم توزیع شده چگونه است؟ کسی نمی‌داند.

فصل پنجم: سیستم فایل توزیع‌شده

- شکل ۶-۵ ص ۲۵۷ خلاصه شرایط سیستم مطالعه‌شده.
- فایل‌های زیر 10K ⇦ جابجائی کل فایل به صرفه است تا نقل و استمال داده‌ها بین Server, Client.
- کوتاه بودن عمر اکثر فایل‌ها ⇦ مثال فایل میانی کامپلرها ⇦ایجاد فایل موقتی در Client نیمه پائین آوردن ترافیک است.
- کم بودن درصد Sharing ⇦ می‌توان از Caching استفاده کرد چون مشکل کمتری دارد.
- کلاس‌های مختلف فایل با عملیات خاص ⇦ برخورد مختلف روی کلاسهای مختلف مثلاً بانیری‌ها تعمیر نمی‌کنند می‌توانند تکرار شوند، Mailها نیازی به تکرار ندارند.

فصل پنجم: سیستم فایل توزیع‌شده

- ساختار سیستم
- تشریح ساختار و سازمان داخلی File Server و Directory Server .
- سئوال: آیا Client و Server با هم تفاوت دارند.
- در بعضی سیستم‌ها اینکه کدام باشی فرقی نمی کند هر ماشینی می‌تواند مثلاً Server شود.
- در بعضی سیستم‌ها Client و Server برنامه‌های کاربرند و می‌تواند روی یک ماشین باشند.
- در بعضی سیستم‌ها Client و Server اساساً تفاوت دارند و روی ماشین‌های مختلفند و حتی DSهای مختلف.
- دلیلی هم برای ..... هیچکدام بر دیگری نیست.
- موضوع پیاده‌سازی دیگر در مورد فایل‌ها و دایرکتوری‌هاست.
- آیا هر دو روی یک Server باشند یا روی دوتا.

فصل پنجم: سیستم فایل توزیع‌شده

- اگر دو تا باشد دو تا رجوع به دو تا ماشین داریم برای هر دسترسی به فایل.
- مزایای جداسازی: دو تا Dir Server برای Dos و UNIX ولی یک File Server برای نگهداری فیزیکی.
- معایب جداسازی: افزودن ترافیک.
- مثال شکل ۷-۵ ص ۲۵۹.
- در a جستجو برعهده کاربر گذاشته می‌شود در b سیستم اینکار را می کند.
- در b نمی‌توان از RPC استفاده کرد زیرا درخواست به کس دیگر است پاسخ از کس دیگر.
- جستجوی Pathها گران و دقت گیر است، یک راه نگهداری Cache Hint از آنهاست.
- اگر در Cache هست پله پله رفتن لازم نیست و گر نه هست.
- آیا اطلاعات حالت کاربران توسط Server نگهداری شود؟

فصل پنجم: سیستم فایل توزیع‌شده

- **Stateless :**
  - طرز فکر ۱: هر درخواست که آمد انجام بده و پس از خاتمه کلیه اطلاعات مربوطه را پاک کن.
  - طرز فکر ۲: نگهداشتن اطلاعات حالت کاربران در بین درخواست‌های آنان.
  - مثال: فایل‌های open شده: سیستم بداند چه کسی چه فایلی را باز کرده.
  - هر کاربر با یک File Descripter درخواست عمل می‌دهد: جدول تبدیل اسم به آدرس جدول حالت است.
  - Stateless: هر دستور باید خود کفا باشد: نتیجه ⇦ بالا رفتن ترافیک.
  - سقوط Server: اطلاعات جدول‌ها از بین می‌روند، نیاز به ترمیم دقیق دارد.
  - ظاهراً Stateless بهتر بنظر می‌رسد از دید تحمل خطا (حداقل).
  - خلاصهٔ مزایای دو روش در شکل ۸-۵ ص ۲۶۱.
  - بعضی اوقات بدلایل ایمنی، سیستم Stateless را خودمان Stateless می‌کنیم. سیستم ثبت نام.

فصل پنجم: سیستم فایل توزیع‌شده

- **Caching**
- چهار محل حفظ فایل شکل ۹-۵ ص ۲۶۲.
- سرراست‌ترین محل دیسک سرور است، در دسترس همه، فضای کافی، بدون افزونگی.
- مسئله انتقال‌هاست دیسک سرور←شبکه ← حافظه Client ← یک Clint
- Caching کارآئی را بالا می‌برد (در حافظهٔ Server).
- محدودیت اندازه حافظه Cache، واحد Caching؟ (کل فایل یا بلوک سخت‌افزاری).
- کل فایل: نقل و انتقال سریع با کارآئی بالا.
- بلوک‌ها: استفاده بهینه از فضای دیسک و حافظه.
- وقتی Cache پرشد چه کنیم: استفاده از الگوریتم LRU و یک لیست پیوندی و بروز کردن دیسک در صورت لزوم.

فصل پنجم: سیستم فایل توزیع‌شده

- Cache در حافظهٔ Server باعث سازگاری فایل و عدم انتقال از فایل می‌شود ولی ترافیک شبکه را دارد.
- راه دیگر Caching در طرف Client است (حذف ترافیک شبکه)
- مسائل جدید:
- دیک Client یا حافظه Client اینجا روی حافظه بحث می‌شود.

فضای زیاد و سرعت کم فضای کم و سرعت زیاد

- سه راه: Cache در فضای آدرس هر کاربر شکل ۵-۱.b، در بستن Cache پس نویسی می‌شود.

Overhead کم است در صورتی کارآئی دارد که هر پروسس مکرراً بفایل رجوع کند.

سیستم DB در مقابل سیستم وابسته به زبان (توربو C).

راه دوم: Cache کردن در فضای Kernel شکل ۵-۱.....: هر دسترسی یک احضار Kernel.

راه سوم: Cache کردن در فضای مدیر Cache پروسس جدا شکل ۵-۱.b: آزاد کردن Kernel.

فصل پنجم: سیستم فایل توزیع‌شده

- سازگاری Cacheها
- دو نفر همزمان بخوانند، بعد نفر سوم بخواند ⇦سه کپی متفاوت: راه حل قبول Session Sem.
- عیب مهم هم زمان نوشتن است که آخری می‌ماند! UNIX Sem این را قبول ندارد.
- یک راه الگوریتم Write through است که از طریق Cache تغییرات می‌رود روی فایل.
- مشکل این راه Cacheهای کهنه روی ماشینهای دیگر است که بعد ممکن است استفاده شود.
- راه حل: Cache mgr هر بار بوسیله زمان یا شماره ردیف نو بودن یا کهنه بودن را با مقایسه با Server انجام دهد اگر لازم است کپی جدید بیاورد نیاز به یک RPC دارد ولی با نقل و انتقال کم.
- مشکل Wrt tru: همه نوشتن‌ها روی شبکه است انگار Cacheای نیست.
- بعضی طراحان تقلب می کنند: نوشتن‌ها را جمع می کننده هر Δt (۳۰. ثانیه) اعمال می کنند!

فصل پنجم: سیستم فایل توزیع‌شده

- مشکل دیگر فایل‌های Scraleh است (ایجاد - خواندن - حذف) که نیازی به بهنگام‌سازی ندارد این‌ها بهتر است روی Server نباشد بطور کلی.
- البته Δt بالا هم Senantis را بهم می‌زند و وابستگی به زمان ایجاد می کند.
- نیزا به سازش بین دوستی Semantis و Performance داریم.
- Write-on-Close یا همان Session Sem راه حل دیگر است که مشابه سیستم‌های تک CPU است.
- دو پروسس هر یک فایلی را بخوانند اصلاح کنند و بنویسند (در تک CPU).
- یک راه کاملاً متفاوت: الگوریتم متمرکز، بداند چه فایلی را چه کسی برای چه کاری باز کرده.
- خواندن بتواند همزمان باشد ولی نوشتن فقط یکی باشد.
- درخواستهای غیرفایل انجام یا Deny شود یا queue شود.

فصل پنجم: سیستم فایل توزیع‌شده

- یک راه هم ارسال پیام درخواست نشده (Unsoli ated msg) که فایل خاصی را از cache حذف کنند.
- حال خواندن و نوشتن‌های آن فایل مثل سیستم تک CPU می‌شود.
- ولی این باعث تغییر نقش Client و Server می‌شود.
- در Client چند رشته‌ای یک رشته گوش بدهد به این پیام‌ها و گرنه باید از روش وقفه استفاده شود.
- راه دیگر مدیریت Cacheها بجای مدیریت فایل‌های باز است و بزور نگهداشتن آنها.
- طبق معمول هیچ راه حلی که ..۱% کار کند نیست، خلاصه شکل ۱۱-۵ ص ۲۶۷.
- خلاصه: Server Caching ساده‌تر است و مشکل کمتری دارد و مستقل از Client است.
- Client Caching بازدهی بهتری دارد ولی پیچیده است.
- یک راه هم فایل‌های immutable است که قبلاً گفته شد و Semautics آن روشن‌تر از بالائی است.

فصل پنجم: سیستم فایل توزیع‌شده

- تکرار
- به دلایلی که در زیر گفته شده بعضی فایل‌ها روی چند Server کپی می‌شوند.

۱- بالابردن قابلیت اطمینان (چند back up).

۲- در توقف یک Server بازی ادامه دارد و بقیه سرویس می‌دهند (قابلیت دسترسی).

۳-تقسیم بار کاری بین چند Server بخصوص در شرایط بار زیاد (کارآئی).

- موضوع دیگر وجود یا عدم وجود شفافیت تکرار است (دو Extreme: کلاً در اختیار سیستم یا User).
- ارائه سه روش در شکل ۱۲-۵ ص ۲۶۹.
- در a همه کار با کاربر است فایل prog .c را روی یک Server کامپایل می‌کند روی بقیه کپی می‌کند.

فصل پنجم: سیستم فایل توزیع‌شده

- در /mach 1/usr/as/ng 2 ایجاد می‌شود و با دستور CP روی /mach 2/usr/ast/ng 2 کپی mach 3 می‌شود.
- در بازیابی ماشین‌ها به ترتیب چک می‌شود تا از یکی بتوان خواند.
- اسم سمبولیک با شماره نود و اسم باینری فایل map می‌شود.
- در b اول client یک کپی روی یک ماشین (معلوم نیست کدام) درست می‌کند بعد که وقت پیدا کرد کپی‌ها را روی بقیه ایجاد می‌کند بدون اطلاع کاربر. باید مواظب تغییرات بعدی باشد.
- تکرار در قالب گروه شکل C: همه نوشتن‌ها در یک زمان روی همهٔ کپی‌ها در گروه انجام می‌شود.
- برعکس Lazy rep کار ایجاد کپی‌ها در Background نیست بلکه همه در یک زمان است.

فصل پنجم: سیستم فایل توزیع‌شده

- پروتکل‌های بهنگام‌سازی
- چگونه کپی‌های موجود بهنگام شود.
- یک پروسس مسئول ممکن است وسسط کار سقوط کند و بعضی بهنگام نشوند.
- روش .... Primary: یک نسخه اصلی است بقیه کپی.
- تغییرات به نسخه اصلی می‌رود پس از اعمال از آنجا به فرعی‌ها.
- قبل از اعمال روی اصلی در حافظهٔ ماندنی ثبت می‌شود، در صورت سقوط اصلی بعد از آنجا برداشته می‌شود.
- مشکل: اگر اصلی توقف کرد هیچ بهنگام‌سازی نمی‌تواند وارد شود.
- یک راه حل رای گیری (Voting) است.
- برای هر عمل خواندن یا نوشتن باید از نصف به اضافه یک Serverها مجوز گرفته شود.

فصل پنجم: سیستم فایل توزیع‌شده

- با ۵ کپی اگر ۳ تا ورژن ۸ باشند بقیه نمی‌تواند ۹ باشد، این سه تا که ۹ شد می‌شود اصلی.
- روش دقیق‌تر شکل ۵-۱۳ ص ۲۸۱ Nr تعداد برای خواندن Nw برای نوشتن تعداد کل

  $Nr + Nw > N$
- اگر چند Server سقوط کند نمی‌توان Nw مناسب گرفت.
- راه حل: ایجاد Server مجازی بجای آنها فقط برای نوشتن و نه خواندن.
- وقتی Server اصلی برگشت اول خود را update کند بعد برود در مدار سرویس. (رای گیری با ارواح)
- مثال عملی: سیستم فایل روی SUN Network File System ≡ NFS.
- یک محیط ..... که توسط SUN ارائه شد بقیه هم حالا Support می کنند.
- Clientهای MS Dos فایل‌های UNIX را می‌خوانند و می‌نویسند.

فصل پنجم: سیستم فایل توزیع‌شده

- معماری NFS
- استفاده مشترک یک سیستم فایل از طرف گروه دلخواهی از Clientها و Serverها.
- معمولاً همه روی یک LANاند ولی می‌توانند روی WAN هم باشند.
- فرض می‌کنیم روی ماشینهای مختلف باشند ولی در عمل هر ماشین می‌تواند همزمان هم C و هم S باشد.
- هر NSF سرویس‌دهنده دایرکتوری‌های خود را برای دسترسی از راه دور Export می‌کند.
- Dir و همه شاخه‌هایش قابل دسترسی است و در /etc/export قرار دارد.
- با هر Reboot اینها صادر می‌شوند.

فصل پنجم: سیستم فایل توزیع‌شده

- Clientها Dir مورد نیازشان را mount می‌کنند که بخشی از درخت Dir آنها می‌شود.
- اکثر سیستم‌های SUN بدون دیسک‌اند و دایرکتوری‌شان روی Server است.
- اگر Disk Full باشد می‌تواند بطور محلی هم mount کند که درخت دوگانه می‌شود (محلی + رامدور).
- برای برنامه‌های در حال اجرا هیچ فرقی نمی‌کند. فایل مورد استفاده می‌کنند و هیچ کار اضافی دیگر هم نباید بکنند. این مهمترین نکته در سیستم NFS است در رابطه با سیستم فایل.

فصل پنجم: سیستم فایل توزیع‌شده

- پروتکل‌های NFS
- بدلیل Heterojeneus بودن سیستم‌های مورد پشتیبانی با S,C در حال اجرا روی ماشین‌های متفاوت و تحت OSهای متفاوت، ارتباط در اینجا پایه دقیقاً تعریف شود.
- در اینصورت نوشتن Client جدید و صدور آن برای کار با هر Serverای امکان دارد.
- S, C از دید طرف ..... (C , S) یک جعبه سیاه است و پروتکل است که ارتباط را فراهم می کند.
- نحوهٔ انجام درخواست هم به S مربوط است نه به C.
- پروتکل اول: mount کردن را پشتیبانی می کند.
- یک مسیر بصورت پیام از C به S داده می‌شود برای اخذ مجوز mount کردن بدون ذکر محل.
- در صورت درستی مسیر، S یک فایل handle حاوی نوع سیستم فایل، دیسک، node، اطلاعات ایمنی و غیره را برمی گرداند.

فصل پنجم: سیستم فایل توزیع‌شده

- عملیات بعدی با فایل با استفاده از .Han dle است.
- خیلی Cها با boot شدن، اتوماتیک فایل /etc/rc را که یک Shell است اجرا می کند.
- این اجرا باعث mount شدن تعدادی Dir خاص در C می‌شود.
- راه دیگر .UNIX Automoan Fing است.
- به هر C تعدادی Dir راه دور وابسته می‌شود بدون mount شدن یا ارتباط با S آن.
- با boot شدن هر فایل که open شد پیامی به Sهای آن ارسال می‌شود.
- هر کس زودتر جواب داد، Dir آن mount می‌شود.
- دو مزیت: اول در /etc/rc (استاتیک ماونت) یک S خراب باشد C نمی‌تواند boot شود در حالیکه در Automnt اینطور نیست و با اولین جواب کار حل می‌شود.

فصل پنجم: سیستم فایل توزیع‌شده

- دوم: Automnt درجه‌ای بالا از تحمل خرابی دارد چون کافی است فقط یک S جواب بدهد.
- کارآئی بالا می‌رود اگر S با کمترین بار کاری انتخاب شود.
- از طرف دیگر NFS تکرار را پشتیبانی نمی‌کند و این کلاً برعهدهٔ User است.
- Automnt فرض می‌کند کلیهٔ Sها حاوی نسخه‌های مشابه فایل‌اند.
- نتیجه: Automnt عموماً برای فایل‌های Readonly و کد ماشین‌های تا تغییر بسیار کم استفاده می‌شود.

فصل پنجم: سیستم فایل توزیع‌شده

- پروتکل دوم: برای دستیابی به Dir و فایل است
- C پیغام به S می‌دهد برای خواندن یا نوشتن فایل و کار با Dir و یا خواندن مشخصات فایل.
- بجز Open و Close بقیه احضارهای سیستم UNIX توسط NFS قبول می‌شود.
- حذف ایندو دستور هم برای Stat less کردن Sهاست (قبلاً داشتیم).
- سیستم UNIX V روش Remote File Sys یا RFS دارد که State File است.
- عیبش را می‌دانید؟ امکان پیاده‌سازی UNIX Sem نیست.
- مشکل NFS سیستم ..... را بکار برد با امید موفقیت!
- ولی ساده لوحانه بود.

فصل پنجم: سیستم فایل توزیع‌شده

- حالا مشخصات S, C رمزنگاری می‌شود تا از تقلب جلو گیری شود.
- NIS یا Network Info Server کلیدهای شناسائی و ارتباطات لازم را فراهم می کند.
- دو ستون (کلید، مقدار) ذخیره می کند، کلید می گیرد مقدار را برمی گرداند.
- کلیدها را رمزنگاری می کند و با User mane آنها را map می کند.
- اسم ماشین‌ها را هم با آدرس شبکه‌شان map می کند.
- به روش master/slave تکرار می‌شود، خواندن از هر کپی و نوشتن روی master که بعد می‌رود روی slaveها.

فصل پنجم: سیستم فایل توزیع‌شده

- پیاده‌سازی NFS
- فراموش نکنید که کد S, C مستقل از پروتکل NFS است، ولی یک نگاه خود ندارد.
- سه سطح در شکل ۱۴-۵ ص ۲۷۶ نشان داده شده.
- سطح بالائی: سطح احضار سیستم، احضارهائی مثل READ، OPEN، CLOSE را انجام می‌دهد.
- سطح وسطی: سطح سیستم فایل مجازی (VFS) بعد از چک شدن دستور و پارامترها در سطح بالائی، این سطح احضار می‌شود.
- دارای جدولی است از فایل‌های باز، هر فایل در یک سطر، مشابه جدول i-node فایلهای باز UNIX.
- در UNIX عادی i-node با زوج (device – i node#) مشخص می‌شود.

فصل پنجم: سیستم فایل توزیع‌شده

- در VFS یک i-node مجازی داریم می گویند v-node برای هر فایل باز.
- v-node می گویند فایل محلی است یا از راه دور (با اطلاعات کافی برای دسترسی).
- دنبار کردن یک مثال MPUNT ← OPEN ← READ احضارهای سیستم.

۱- مدیر سیستم برنامه mount را احضار می کند و Dir راه دور، Dir محلی برای نصب و سایر اطلاعات را می‌دهد.

۲- برنامه mount اسم مسیر راه دور (Dir) را تجربه کرده و نام ماشین مربوط را استخراج می کند.

۳-با پیامی درخواست File handle را از ماشین مزبور می گیرد (از Server).

۴-در صورت وجود Dir روی آن ماشین و قابل دسترسی بودن، handle مربوطه بر گردانده می‌شود.

۵-یک احضار سیستم MOUNT انجام شده و handle را به Kernel می‌دهد.

فصل پنجم: سیستم فایل توزیع‌شده

۶- Kernel یک v-node می‌سازد برای Dir راه دور و سپس از برنامه NSF Client شکل ۵-۱۴ درخواست r-node (i-node از راه دور) در جدول داخلی‌اش می کند برای نگهداری هندل فایل.

۷- v-node اشاره می کند به r-node در برنامه NSFC یا به i-node یا به i-node در DS محلی شکل ۵-۱۴ از روی v-node محلی یا راه دور بودن فایل‌ها و برای راه دور هندل فایل یافته می‌شود.

۸- در هنگام open فایل راه دور، kernel با تجربهٔ اسم مسیر، Dir محل نصب فایل را می‌فهمد.

۹- از روی v-node آن می‌فهمد راه دور است و اشاره گر به r-node را برمی‌دارد.

۱۰- از کد NSF Client درخواست باز کردن فایل را می کند.

۱۱- کد NSF Client از روی بقیه اسم مسیر و Sهای وابسته به آن Dir نصب شده یک هندل برای فایل می گیرد (ر ک ۶).

فصل پنجم: سیستم فایل توزیع‌شده

۱۲- او برای فایل دور یک r-node می‌سازد در جدولش و آنرا به لایه VFS برمی گرداند.

۱۳-لایه وسطی یک سطر در جدول خودش با V-node اشاره گری به r-node درست می کند پس هر فایل یا Dir باز، اشاره گری در قالب v-node یا به r-node دارد یا به c-node.

۱۴- به احضار کننده یک File-des cripter برای فایل باز شده داده می‌شود.

- File des c از طریق جدول VFS به v-node نگاشت می‌شود.
- هیچ جدولی در طرف S ساخته نمی‌شود.
- S با درخواست هندل را می‌دهد.
- S هندل را می گیرد، چک می کند درست باشد و بکار می‌برد.
- در صورت وجود کلید مجوز در RPC آنرا هم چک می کند.

فصل پنجم: سیستم فایل توزیع‌شده

۱۵- وقتی یک احضار سیستم برای READ داده می‌شود (با استفاده از File-des c) لایه VTIF از جدولش v-node مربوطه را برمی‌دارد.

۱۶- تشخیص می‌دهد که فایل محلی است یا دور و r-node یا i-node مربوط را می‌یابد.

۱۷- VFS تکه‌های 8K هر بار می‌خواند و قبل از مصرف هر بلوک، بلوک بعدی را درخواست می‌کند برای بالا بردن کارآئی، به این مکانیزم READ AHEAD گویند.

۱۸ – در نوشتن هم داده‌ها در Client جمع می‌شود تا 8K شود بعد ارسال می‌شود مگر زبان تشخیص فایل.

- Caching هم برای افزایش کارآئی در Server انجام می‌شود که ترافیک شبکه را کم می‌کند.
- یک Cache حافظه را دارد باضافه بعضی مسائل جدید (Snooping نمی‌شود کرد).
- برای مقابله با این موضوع.

فصل پنجم: سیستم فایل توزیع‌شده

- هر بلوک Cache یک Timer هم دارد که وقتی Expine شد بلوک هم حذف می‌شود.
- ۳ ثانیه برای Clala و ۳۰. ثانیه برای Timer Dir می گذارد.
- وقتی فایل Cache شده را بخواهیم باز کنیم، پیامی به S می‌دهد که آخرین بار یک فایل اصلاح شده.
- اگر تاریخ کپی Cache کهنه‌تر است آنرا حذف می کند و کپی جدید درخواست می کند.
- هر ۳۰. ثانیه یکبار ساعت صفر شده و بلوکهای اصلاح شده به S ارسال می‌شود.
- هنوز از NFS بدلیل عدم پیاده‌سازی دقیق UNIX Sem انتقاد می‌شود.
- فایل جدید ایجاد شده می‌تواند برای ۳۰. ثانیه رویت نشود و این خود مشکل‌زا می‌باشد.
- پس گر چه NFS بصورت عام استفاده می‌شود ولی بعنوان UNIX وصله‌شده شناخته می‌شود زیرا Semantic مربوطه خوش ترکیب نیست و وابسته به زمانبندی‌هاست.

فصل پنجم: سیستم فایل توزیع‌شده

- آنچه فرا گفته شد
- شکل ۱۵-۵ اصول اولیه مورد لزوم برای طراحی سیستم فایل توزیع‌شده را می‌دهد.
- انتخاب W.S. بجای S وقتی امکان دارد
- Caching.
- بهره‌برداری از اطلاعات کاربر و Cache نکردن فایل‌های موقتی.
- طراحی سلسله مراتبی.
- تا می‌توانید اجزاء را محدودتر کنید.
- یک تراکنش 50K بمراتب کارآتر از ۵۰ تراکنش یک ..... است.

فصل پنجم: سیستم فایل توزیع‌شده

بخش ۳ زمینه‌های تحقیقاتی در سیستم‌های فایل توزیع‌شده

- پیشرفت در هر دو زمینه سخت‌افزار و نرم‌افزار در مقایسه با سایر رشته‌ها سرسام‌آور است و تأثیر زیادی هم روی سیستم فایل توزیع‌شده دارد.
- موضوع تأثیر گذار دیگر تغییر خواسته‌ها و تغییر کاربردها است.
- در این بخش سئوالات مطرح در زمینه فوق بررسی می‌شود و زمینهٔ تحقیق برای شما دانشجویان فراهم می‌شود.

فصل پنجم: سیستم فایل توزیع‌شده

- سخت‌فزارهای جدید
- در سیستم‌های موجود دیسک بعنوان محل ذخیره اطلاعات استفاده می‌شود و حافظه اصلی نقش Cache را در Serverها دارد که اجباری نیست، برای بالابردن کارآئی است.
- با پائین آمدن شدید قیمت حافظه، امکان دارد در آینده سیستم‌های فایل با چندین گیگابایت حافظه مجهز شوند.
- پس ممکن است سیستم فایل کلاً در حافظه بماند و به دیسک احتیاجی نباشد که این جهش عظیمی در سیستم فایل است در جهت بالابردن کارآئی و ساده‌سازی ساختار سیستم فایل.
- فعلاً ساختار درخت (UNIX) یا لیست پیوندی (NS Dos) بکار می‌رود ولی در حافظه اصلی می‌تواند فضای پیوسته باشد که باعث بالابردن سرعت و کارآئی می‌شود.

فصل پنجم: سیستم فایل توزیع‌شده

- جابجای بلوک‌ها روی دیسک هزینه دارد ولی در حافظه اینطور نیست.
- ولی مشکل اصلی قطع برق است.
- یک راه back up روی نوار ویدئو است هر نوار معمولی ده دلاری ۵ گیگابایت گنجایش دارد.
- گر چه زمان دسترسی طولانی است ولی امید است قطع برق یکی دو بار در سال باشد.
- یک پیشرفت HW دیسک نوری است.
- ایجاد سوراخ بالیزر، یکبار نوشتن و چند بار خواند (WORM).
- انواع جدید قابل پاک کردن و نوشتن مجدد.

فصل پنجم: سیستم فایل توزیع‌شده

- خصوصیات دیسک نوری

•سرعت پائین • ظرفیت بسیار زیاد • دسترسی مستقیم •نسبتاً ارزان

- در مقابل ویدئو دسترسی مستقیم را دارد ولی قیمت آن گران‌تر است.
- داشتن سیستم فایل در حافظه، کپی فایل در زمان بیکاری سیستم روی دیسک نوری با همان اندازه و ساختار.
- راه دیگر استفاده از حافظه اصلی Server برای فایل و دیسک نوری برای Back up ولی شبکهٔ فایبر برای انتقال اطلاعات ⇦حذف مشکلات شبکهٔ معمولی Client Cacheیا Server Cache
- عیب Client Cache: تغییر فایل پیش یک Client و بوجود آمدن ناسازگاری بین کپی‌ها.
- در حافظهٔ مشترک بین چند CPU وقتی یکی تغییر می‌دهد به بقیه CPUها خبر می‌دهد ولی در سیستم فایل اینطور نبود. چرا نباشد؟

فصل پنجم: سیستم فایل توزیع‌شده

- دلیل آن عدم وجود سیگنال‌های لازم برای اینکار است.
- یک راه حل ساده در شکل ۱۶-۵ ص ۲۸۲ ارائه شده است.
- هر کاربر یک **Bet Map** دارد که هر بیت مربوط به یک فایل است.
- اگر بخواهد فایل Cache شده‌ای را تغییر دهد در **map** خودش بیت آنرا **Set** می کند.
- سیستم یک پاکت روی شبکه می‌فرستد که بیت مربوط را در جاهای دیگر **Set** کند (فایل قفل شود).
- بعد فایل را تغییر دهد و بلوک‌های تغییر یافته را علامت گذاری کند.
- بلوک‌های تغییر یافته را روی بقیه ماشین‌ها کپی کند و سپس **Bet** مربوطه در **map** را صفر کند.
- راه حل خوبی است که فقط در سایه افزودن یک سخت‌افزار ساده امکان‌پذیر شده و آینده اینطور است.

فصل پنجم: سیستم فایل توزیع‌شده

- توسع‌پذیری
- یک روند دیگر توسعه‌پذیری سیستم و بزرگتر و بزرگترشدن سیستم فایل است.
- سیستمی خوب با ..ا ماشین بین ...ا ماشین کند می‌شود و برای ...,.ا ماشین اصلاً نمی کند
- عناصر متمرکز همیشه تنگناهای سیستم هستند.
- یک راه کلی تقسیم کل سیستم به چند پارتیشن با یک Server برای هر پارتیشن است.
- سعی شود هر پارتیشن حداکثر استقلال را داشته باشد.
- پیچیدگی دیگر broad castها هستند n ماشین با یک broad cast در ثانیه می‌شود n^2 تا وقفه که مشکل‌ساز است.
- لیست کردن اجزاء و جستجوی خطی روش خوبی نیست در مقابل hash کردن روش مناسب است
- Semanticsهای دقیق نظیر UNIX برای پیاده‌سازی دقیق مشکلند و نیاز به یک سازش وجود دارد گر چه sem دقیق خوش تعریف است.
- درخت‌های فایل UNIX اگر خیلی بزرگ شد کند می‌شود بهتر است چند بخش شوند.

فصل پنجم: سیستم فایل توزیع‌شده

- شبکهٔ Wide Area
- تمرکز فعلی روی LAN است در حالیکه جهت بسوی ارتباط کشورها و ایجاد ارتباط بین LANهاست.
- PTT فرانسه می‌خواهد در هر خانه یک کامپیوتر بگذارد برای حذف اپراتو ر و دفتر تلفن.
- بعد ممکن است بخواهند ده میلیون کامپیوتر را به یک شبکه وصل کنند.
- چگونه یک سیستم فایل کل فرانسه، کل اروپا یا کل دنیا را سرویس می‌دهد؟
- در سیستم فوق همه ماشین‌ها یکسانند ولی در واقع ممکن است ماشین‌ها هم متفاوت باشند.
- استفاده کننده‌ها در زمانهای مختلف با بودجه‌های مختلف اجباراً ماشین‌های متفاوت دارند.
- چگونه کاراکترهای مختلف ASCII یا ... تطبیق شوند یا مقادیر اعشاری با نمایش‌های مختلف.
- مطلب دیگر تغییرات در کاربردهاست.

فصل پنجم: سیستم فایل توزیع‌شده

- در شاخه تحقیق در دانشگاهها برنامه‌های C روی UNIX نوشته می‌شوند.
- مسلماً خانواده‌های فرانسوی همه چنین کاری نمی‌کنند.
- با توسعه سیستم‌های توزیع‌شده، کاربردهائی مثل Email، بانک E، دسترسی به DBها و سرگرمی‌ها که کاربردها فایل مختلفی دارند.
- تنگنای دیگر ظرفیب خط ارتباطی‌ست، خط تلفن حداکثر 64Kbps است.
- ایجاد شبکه Fiber زمان و هزینه بسیار زیادی می‌خواهد.
- ارسال یک نوار یا CD حاوی کل اطلاعات ممکن است ارزان‌تر از ایجاد شبکه باشد (سیستم تلفن Dir).
- پس باید بین اطلاعات استاتیک (سیستم تلفن) و اطلاعات دینامیک (Email) در سیستم فایل تمایز قائل شد.

فصل پنجم: سیستم فایل توزیع‌شده

- کاربردهای موبایل
- یک شاخه از کاربردهای جدید کامپیوترهای جابجاپذیرند (Mobile Computing).
- Laptop، Note book، کامپیوتر جیبی با سرعت تکثیر می‌شوند.
- در حال رانندگی نه ولی در حال پرواز می‌توان با کامپیوتر کار کرد.
- یک کامپیوتر و یک موبایل اکنون شما را به شبکه وصل می‌کند (اشاره به .....).
- امید است ظرفیت خط بین هواپیما و زمین و همینطور ارتباط online این روزها برای همه جا باشد.
- البته هنوز هم ارتباط off line وجود دارد و بخش اعظمی از کار کاربران off line است (مثال خودمان).
- کاربر وصل می‌شود فایلهائی را down load می‌کند بعد قطع می‌شود و مدتی (ساعت‌ها، روزها) کار می‌کند.
- دوباره که وصل شد فایل‌های او با سیستم فایل باید او تمام شود (مدتی از Caching است).
- مسئله ناسازگاری کپی‌ها خیلی شدت پیدا می‌کند تا در یک سیستم همیشه online.
- آیا دوباره که وصل شد کجای دنیا قرار دارد با چه فاصله‌ای از Server خود مسئله‌ای است.

فصل پنجم: سیستم فایل توزیع‌شده

- تحمل خرابی
- بجز سیستم‌های خاص، در حالت عادی کاربر باید خرابی را بعنوان حقیقت قبول کند چون همه سیستم‌ها متحمل خرابی نیستند ولی عامه مردم این را نمی‌پذیرند (قطع برق، قطع تلفن، .....).
- کاربران عادی انتظار دارند سیستم همیشه کار کند که این امکان‌پذیر نیست.
- چنین سیستمی نیاز به تکرار در اجزاء مختلف (قبلاً گفته شده) دارد.
- سیستم باید بتواند با بخشی از اطلاعات هم کار کند وقتی همهٔ سیستم آماده کار نیست زیر F.T. صددرصد بسیار مشکل است. (شماره دانشجو را بدهد نامش را ندهد در ثبت‌نام).
- کاربران غیر کامپیوتری بدلیل عدم اطلاع توقعاتشان بسیار بالاتر از متخصصین خواهد بود.

فصل پنجم: سیستم فایل توزیع‌شده

- چند رسانه‌ای
- یک روند دیگر بخصوص در سیستمهای R.T. استفاده از صوت و تصویر و فیلم است که روی سیستم فایل اثر می‌گذارد.
- یک سیستم Video on Demand نیاز به حجم بسیار زیادی حافظه و سیستم فایل خاص دارد.

۵ دقیقه فیلم رنگی

ثانیه: ۵×۶۰ — فریم‌ها: $\times ۲۴$ — صفحه نمایش: $\times ۶۰۰ \times ۸۰۰$ — رنگ: $\times ۱۶$ — بیت: $= ۵۵,۲۹۶,\ldots,\ldots$ بایت $= ۶,۹۱۲,\ldots,\ldots$

با یک خط 64Kbps تقریباً

$$\frac{8 \times 12000}{60} = 8 \times 200, \text{ دقیقه} \qquad \frac{1600}{60} \approx 26 \text{ ساعت}$$