سیستم‌عامل پیشرفته

محمد داورپناه جزی

گروه مهندسی کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی فولاد

ترم دوم ۹۴-۹۳

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- چگونه با حضور چند پردازش کار انجام می‌شود.
- رشته یا **Thread**.
- در سیستم‌های عامل مرسوم و عادی هر پروسس یک فضای آدرس و یک رشتهٔ کنترل اجرا دارد که در واقع تعریف آن است.
- حالاتی پیش می‌آید که در یک فضای آدرس چندین رشتهٔ کنترل آنرا **Share** کنند و بطور تقریباً موازی اجرا شوند. (مثل اینکه پروسس‌های مختلف هستند.)
- مثال **Fileserver**: مجبور است برای دیسکت صبر کند، اگر چند رشته داشته باشد یکی منتظر شود دیگری می‌تواند اجرا شود.
- با دو پروسس مجزا امکان‌پذیر نیست زیرا باید از یک فضای آدرس هر دو استفاده کنند.
- مثال شکل ۱-۴ ص ۱۷.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- نسبت رشته به پروسس مثل نسبت پروسس به ماشین است ولی استقلال کمتری دارند.
- رشته‌ها قرار است همکاری کنند نه رقابت (پروسس‌ها رقابت هم می‌کردند).
- شکل ۲-۴ ص ۱۷۱ خصوصیات پروسس‌ها، رشته‌ها و تفاوت آنها.
- رشته‌ها برای تلفیق توازی و اجرای ترتیبی و احضارهای سیستم بلوکه کننده.
- حالت a شکل ۳-۴ مثال **Fileserver**.
- **Dispatcher** درخواست را می‌گیرد و توزیع می‌کند بین **Worker**ها (نوشتن آدرس پیام در یک **Word** مرتبط با آن و بیدار کردن رشته).
- یک درخواست خواندن به یک رشته داده می‌شود اگر با بلوک **Cache** اشتراکی موجود بتواند انجام می‌دهد.
- اگر نتواند درخواستی به دیسکت می‌دهد، می‌خوابد، بعد **dispatcher** بکار می‌افتد و بعد یک رشته دیگر را بیدار می‌کند که آماده اجرا است.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- در روش قدیمی اینطور نبود، در صورت انتظار کل پروسس Server و کل ماشین (Dedicated CPU) منتظر می‌شوند تا درخواست اجرا شود.
- با رشته‌ها نتیجه این است که تعداد کار بیشتری در واحد زمان انجام می‌شود.
- راه دیگر استفاده از ماشینهای با حالات محدود است با یک رشته در یک پروسس.
- پس از وصول درخواست اگر شد (با Cache موجود) اجرا می‌شود و گر نه درخواستی به Disk می‌رود.
- بجای بلوکه شدن وضعیت آن در جدولی ثبت و پیغام بعدی پردازش می‌شود.
- اگر یک درخواست جدید است مثل بالا عمل می‌شود اگر پاسخ دیسک است پروسس مربوط با آوردن اطلاعات از جدول بکار ادامه می‌دهد.
- بدین ترتیب ایده اجرای ردیفی را نداریم ولی توازی تقویت می‌شود خلاصه سه حالت شکل ۴-۴ ص ۱۷۳.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- ادامه ش ۳-۴ b حالت تیمی که رشته‌ها همزمان درخواست می‌گیرند یا از یک صف انتخاب می‌کنند می‌توان صف‌های گوناگون برای کارهای گوناگون داشت.
- ش ۳-۴ c حالت Pipelining هر رشته خروجی خود را به دیگری می‌دهد .استفاده در UNIX.
- موارد استفاده رشته‌ها.
- کپی کردن یک فایل توسط یک مشتری به چند) Server یک رشته با هر Server حرف بزند.(
- پردازش سیگنال‌های وقفه مثلاً صفحه کلیدDEL ، BREAKو ... یک رشته مخصوص اینها.
- بجای وقفه دادن به کل پروسس.
- مسائل ذاتاً موازی مثل تولیدکننده و مصرف‌کننده که از یک بافر مشترک استفاده می‌کنند.
- چند پردازنده‌ای که بطور موازی واقعی رشته‌ها را در یک فضای آدرس اجرا می‌کنند.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مفاهیم طراحی در رشته‌ها (Threads Package)
- رشته‌های استاتیک: تعیین رشته‌ها در زمان کامپایل و عدم امکان تغییر، ساده ولی غیرانعطاف‌پذیر.
- رشته‌های دینامیک: ایجاد و حذف رشته‌ها در زمان اجرا. مثال fork
- احضار برای ایجاد رشته حاوی اشاره گر به روال، اندازه Stack و تقدم اجرا.
- یک شناسهٔ رشته برای احضارهای بعدی برمی گرداند که نیاز به این رشته دارند، مثال.fork
- در این مدل هر پروسس با یک رشته شروع می‌شود و می‌تواند رشته‌های دیگر تولید/حذف کند.
- توقف می‌تواند طبیعی باشد) خاتمه (یا Kill شود از بیرون، بعضاً هم متوقف نمی‌شود شکل.4-3

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- استفادهٔ رشته‌ها از حافظهٔ مشترک و نیاز به حفاظت نواحی بحرانی
- استفاده از سمافور دوتائی **mutex** و متغیر شرطی شکل ۴-۵ ص ۱۷۶.
- رشته‌ها متغیرهای محلی، پارامتر و سراسری دارند که آخری اشکال ایجاد می‌کند شکل ۴-۶ ص ۱۷۷.
- یک راه منع آن است بطور کلی، که خوب نیست، عدم تطبیق با سایر محیطها
- راه دوم دادن متغیرهای سراسری مخصوص به هر رشته شکل ۴-۷، افزودن یک سطح دید
- راه دیگر ایجاد متغیرهای سراسری در زمان اجرا
- توابع خاص برای کار با آنها، ص ۱۷۸.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- پیاده‌سازی یک **Threads Package**
- پیاده‌سازی در فضای کاربر (UNIXهای قدیمی) – مدل شکل ۸-۴ ص ۱۷۹.
- مجموعه‌ای از روال‌های مدیریت رشته بین **User** و **Kernel** هستند شکل a
- **Kernel** فقط یک پروسس تک رشته‌ای می‌بیند.
- روی سیستم‌های بدون رشته قابل پیاده‌سازی است.
- در معلق کردن رشته، **Reg**ها ذخیره شده، **Reg**های جدید بار می‌شوند و رشته جدید اجرا می‌شود با سمافور یا **mutex**عین پروسسها
- با سریع بودن نقل و انتقال ثباتها) یک دستور (این روش حداقل ده برابر سریعتر از دیگری است)یعنی.(**Traptokernel**
- هر پروسس می‌تواند الگوریتم زمانبندی خودش را داشته باشد، زیاد شدن رشته‌ها اشکالی ندارد.
- اگر در **Kernel** باشد جداول و **stack** محدودیت دارد.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- این روش علیرغم مزایایش، مسائلی هم دارد:
- چگونگی پیاده‌سازی سیستم Callهای بلوکه شونده (خواندن از یک Pipe خالی).
- اجازه دادن اینکار باعث توقف کل رشته‌ها می‌شود که در یک Processاند.
- هدف اصلی داشتن امکان Blocking ولی جلوگیری از تأثیر رشته بلوکه شده بردیگری.
- یک راه تبدیل Sys Callها به غیربلوکه شونده است که نیاز به تغییر در OS دارد و قابل قبول نیست.
- راه دیگر اول تست کردن و بعد انجام عمل Sys Call مثلاً اول چک کنیم که Pipe خالی نیست بعد Read کنیم.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- نیاز به اضافه کردن روال‌های جدید دارد به نام Jacket چون اطراف Sys Call را می گیرد.
- مسئله دیگر Page faultها هستند که کل پروسس را معلق می‌کند تا صفحه آورده شود زیرا خبری از Thread ندارد.
- مسئله بعدی انحصاری بودن گرفتن CPU توسط Threadهاست که امکان دادن CPU به رشتهٔ دیگر از همان پروسس را نمی‌دهد مگر که رشته خودش CPU را رها کند.
- همگام بود رشته‌ها مسئله دیگر است که نیاز به ساعت وقفه دهنده دارد که در محوطه User داخل Process وجود ندارد.
- در Applicهای CPU باوند اصلاً نیازی به Thread نیست چون برنامه Block نمی‌شود.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- پیاده‌سازی رشته‌ها در Kernel شکل ۴-۸ b
- ایجاد و حذف رشته‌ها توسط Kernel.
- یک جدول رشته‌ها (بجای جدول پروسس‌ها) هر سطر شبیه PCB برای پروسس ها برای هر رشته عملکرد عین پروسس تک رشته‌ای.
- وجود Overhead زیاد برای سیستم عامل، Reuse کردن جداول و Data رشته‌ها بجای شروع از اول، انتخاب رشته جدید برای اجرا فقط داخل یک پروسس نیست بلکه از بین همه است.
- مسائل بلوکه کردن و paging و ..... حل می‌شود.
- به هر حال خرج این روش بیشتر است.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مشکل مشترک: Reentrant نبودن اکثر برنامه‌ها بافر مشترک استفاده نشده رونویسی می‌شود، errno مشترک و . . .
- Mallocدر یونیکس هیچ فرضی برای چند رشته‌ای و حفاظت از نواحی بحرانی ندارد
- ایجاد Jacket برای هر روال که یک سمافور روی کل Libray بگذارد در هر زمان فقط یک رشته آن اجرا می‌شود.
- مسئله Signalها که ذاتاً Per Processاند نه Per Thread: دو رشته برای دو منظور مختلف منتظر DEL باشند.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- تلفیق دو روش در قالب Scheduler Activation با استفاده از مزایای هردو
- شبیه‌سازی عملکردهای Kernel Thread با انعطاف‌پذیری‌های User Threads
- تخصیص CPUهای مجازی به هر پروسس یا CPUهای واقعی اگر وجود دارند.
- عملکردهای سیستمی توسط سیستم و عملکردهای وقت گیر در سطح User.
- بلوکه کردن در سطح User با سمافور محلی، انجام Activationبرای سیستم، مکانیزم upcall یا احضار ran time systتوسط kernrl
- CPUمجازی یکی به هر Process داده می‌شود بعد می‌تواند با درخواست بیشتر شود، پس گرفته شود و به دیگران داه شود.
- Karnel :Up Callبا دانستن بلوکه شدن یک پروسس به RTS خبر می‌دهد تا آنرا متوقف و دیگری را شروع کندUp Call) ، احضار rantime syst توسط هسته(.
- امکان بن‌بست وجود دارد . Up Callمنطقی نیست زیرا مسیر عکس طی می‌کند.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- رشته‌ها و RPC
- در یک سیستم توزیع شده هم RPC خوب است هم رشته‌ها.
- روشی برای احضار رشته‌ای در یک پروسس توسط رشته‌ای در پروسس دیگر روی همان ماشین با کارائی بالاتر.
- زیرا حتی در یک سیستم دارای RPC تعداد زیادی احضارها روی همان ماشین است.
- رشتهٔ S از serverبعد از شروع، با گفتن به kernel رابطهای خود (روال‌های قابل احضار و ....) را صادر می‌کند.
- رشتهٔ C از clientبعداز شروع، رابطهای مورد نیاز خود را از Kernelوارد می‌کند.
- حالا Kernel می‌داند C در آینده S رااحضار خواهد کرد مقدمات لازم را فراهم می‌کند.
- با استفاده از یک رشتهٔ پارامترهای مشترک نیازی به کپی‌کردن یا بسته‌بندی پارامترها نیست) تغییر Memory Map)وRPCها به شکل محلی سریع عمل می‌شود.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- تکنیک دوم: ناپدید شدن رشته در server وقتی کارش تمام شد
  - رشتهٔ درخواست کننده از بین می‌رود بدون نیاز به ذخیره کردن چیزی
  - با وصول درخواست جدید در server رشته جدید بوجود می‌آید (Implicit Receive).
  - با تنظیم اشاره گرها، پیغام واصله در فضای آدرس رشتهٔ جدید قرار می گیرد.
  - به این رشته‌ها POP-UP گویند، بلوکه نمی‌شوند، Ssve کردن نمی‌خواهند ش 4-9 ص.185
  - مزایا :بلوکه نشدن رشته برای کار جدید، عدم ذخیرهٔ داده‌های قبلی، سهولت ایجاد رشته جدید نسبت به زنده کردن قبلی، عدم کپی پیغام
  - نتیجه صرفه‌جوئی در زمان) در حال تحقیق روی رشته‌ها.(

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

۴-۲ مدلهای مختلف سیستم توزیع شده

- بررسی دو مدل **Processor Pool, Work Station**.
- مدل **Work Station**: تعدادی کامپیوتر (شخصی) در یک محوطه جغرافیائی مثل دانشگاه یک **LAN** تشکل می‌دهند.
- شکل ۴-۱۰ ص ۱۸۶: بعضی استفاده کننده‌های خاص (اساتید) بعضی استفاده کننده‌های عمومی (دانشجویان) در طول روز.
- نوع **Diskless** با **Fileserver** یا **Disk full** یا یا بدون **Server**.
- **Diskless** معمول‌تر چون ارزان‌تر ، نگهداری ساده‌تر، نصب نرم‌افزارها، **Back up** ، جابجاپذیری و قابلیت دسترسی.
- **Disk full** : یکی از روشهای ص ۱۸۷.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- خلاصه همه در شکل ۱۱-۴ ص ۱۸۹.
- مدل W.S. امکانات فراوان: قدرت محاسباتی و منابع خصوصی، امکانات گرافیکی سریع، امکانات بیشتر با افزودن دیسک.
- دو اشکالات: ارزانی .ا تا ..ا CPU به یک نفر بدهیم بین آنها خفه می‌شود.
- دوم: .W.Sهای بیکار زیاد در حالیکه عده‌ای مشتاقانه نیاز به منابع دارند، نتیجه کارایی پایین.
- حل مشکل اول باPC های چند پردازنده‌ای، هر پنجره یکCPU ، ارزان، کارایی پایین ولی کم اهمیت.
- در حل مشکل دوم :تحقیق زیاد شده، در دانشگاهها 30% ماشینها در شلوغ‌ترین ساعت خالی‌اند.
- دستور فرمان اسم ماشین :rsh نیاز به دانستن ماشینهای بیکار، اجرا در محیط متفاوت، برخورد درخواستها.
- تحقیق روی سه مورد ص :190 یافتن ماشین بیکار، اجرای شفاف دستورات، برگشتن صاحب ماشین.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- معنی W.S. بیکار: کسی روی آن کار نکند! همیشه Clock Server, Mail Server و مشابه آن هستند، login کند ولی کار نکند، برای چند دقیقه صفحه کلید را لمس نکنند و ماوس را، پروسس استفاده کننده‌ای هم بکار نیفتد.
- معنی Load روی ماشین بیکار: وصول Mail خیلی زیاد بدون لمس صفحه کلید.
- یک راه: وارد کردن نام و مشخصات Server بیکار توسط خودش در یک فایل، یک Registry یا چند Registry
- استفاده از فرمان، دستور remote، جستجوی Registry و اجرای دستور روی یک ماشین بیکار.
- Broadcast کردن بیکاری برای همه بدون نیاز به فایل عمومی، سهولت یافتن Server بیکار.
- ایجاد یک Registry در هر ایستگاه، وقوع شرایط :Race راه حل شکل 4-12 ص.191
- راه دیگر انجام بطریقه Client Driven بجای.Server Driven
- اعلام نیازها توسطClient ، وصول جواب و انتخاب.Server

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- سپس انتقال کد و ایجاد محیط محلی در آنجا، مسئله نیاز به شبیه‌سازی دقیق محیط Client شامل سیستم فایل، دایرکتوری، متغیرهای سیستم، . . .
- مسئله نیازها بعضی باید در محل اجرا حل شود) اندازه حافظه، اسم ماشین و(... ، بعضی در Server)فایل‌ها(، بعضی در مبدا) صفحه کلید و صفحه نمایش، ماوس.(
- مسئله وابستگی به زمان و تفاوت زمانی بین دو ماشین) عدم همگامی :(ارسال موارد وابسته بزمان به مبدا تاخیر دارد که خود یک مشکل است.
- خلاصه روش اجرا در راه دور با رعایت شفافیت پیچیدگی فراوان دارد.
- اگر کار براصلی برگشت؟ قطع برنامه‌های!Remote ، دادن مهلت و بعد قطع اگر خودش نکرد!
- راه دیگر انتقال پروسس به ماشین اصلی یا ماشین خالی دیگر.
- مسئله جمع‌آوری ملحقات پروسس از اطراف Kernel و انتقال به ماشین دیگر) فایل‌های باز،تایمرها، پیغام‌های منتظر، بافرها.(
- اینکار ساده‌ای از نظر عملی نیستRPC) های انجام شده، فرزندان ایجاد شده، برگرداندن ماشین به وضع قبلی.(

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مدل process Pool
- مخزنی از CPUها که بصورت دینامیک می‌توان به همه داد (بجای PCهای چند پردازنده‌ای) شکل ۱۳-۴ ص ۱۹۳.
- هر کاربر یک ترمینال مدرن دارد. بیشتر شبیه اشتراک زمان‌های قدیمی است تا مجموعه PC ولی با تکنولوژی جدید.
- شبیه ایده جمع‌آوری فایل‌ها در یک جا برای جمع‌آوری CPUها هم در یک جا با همه مزایای آن.
- استفاده از این مخزن با مدل صف‌بندی است که قبلاً داشته‌ایم شکل ۱۴-۴ ص ۱۹۵.
- متوسط زمان بین درخواست و پاسخ با n ماشین $T = \frac{1}{nM - n\lambda} = T/n$
- یعنی زمان پاسخ به اندازه n برابر کم می‌شود. این باعث می‌شود سنگین‌تر شدن وزنه سیستم متمرکز می‌شود. $T = \frac{1}{M - \lambda}$

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- یک ماشین 1000MIPS زمان پاسخی 100 برابر سریعتر از 100 ماشین خصوصی 10MIPS دارد.
- ولی در کنار زمان پاسخ مسائل چون شیی متمرکز، گرانی قیمت، تحمل خرابی هم هست که در توزیع شده خیلی بهتر است.
- با وجود عدم امکان شکستن هر کاری به n بخش در عوض به K<n بخش باز هم سرعت بالاتر است و مسائل جستجو بدنبال ماشین خالی را ندارد و قایم موشک بازی!، ماشین‌ها صاحب ندارند و پیاده سازی هم ساده‌تر است.
- اگر هم کارهای ساده edit و Mail انجام می‌دهد W.S. اگر کارهای خیلی CPU Boadاست: مخزن پروسس.
- مدل مخلوط: داشتن W.S. برای کارهای سریع و محل و سپردن کارهای محاسباتی سنگین به پروسسورها از مخزن.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

۴-۳ تخصیص پردازنده

- هر روشی که در یک سیستم توزیعی به کار رود نیاز به تخصیص پردازنده‌ها دارد.
- تخصص ماشین به پروسس در حالت .W.S، تخصص پروسسور در روش Pool.
- فرضیات: همه ماشینها یکسانند، تفاوت فقط در سرعت است، همه پروسسورها بهم وصلند.
- ایجاد پروسس جدید به روش Fork است یا به روش تولید فرزند.
- مفسر فرمان از طریق گرفتن یک فرمان عمل Fork را انجام می‌دهد.
- ردهٔ اول الگوریتم های جابجا ناپذیر (Non Migratory).
- پروسس جایابی می‌شود و تا زمان خاتمه همانجا می‌ماند صرف‌نظر از وضعیت ماشین مثل Load یا خالی بودن سایر ماشین‌ها، پیاده‌سازی ساده است.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- ردهٔ دوم الگوریتم‌های جابجا پذیر (Migratory).
- پروسس حتی در زمانیکه در حال اجراست بین پروسس‌ها جابجا می‌شود، بالانس کردن Load بهتر است ولی پیاده‌سازی پیچیده‌تر است.
- تخصیص پردازنده بدون هدف خیلی ساده و بطور ردیفی یا تصادفی می تواند باشد.
- پس یک چیزی بعنوان هدف باید باشد که بهینه شود.
- یکی بالا بردن بکارگیری CPUهاست، جلوگیری از بیکاری CPU ها.
- دومی حداقل کردن متوسط زمان پاسخ.
- مثال شکل 4-15 اگر A را به B,1 را به 2 زمان پاسخ (8+10)/2، برعکس آن می‌شود(6+30)/2
- ملاک دیگر نسبت پاسخ یا response ration است=زمان پاسخ ماشین نوعی /زمان پاسخ واقعی
- P_1 که یک ثانیه کار دارد با 5 ثانیه زمان پاسخ، P_2 که یک دقیقه طول می‌کشد با 70 ثانیه زمان پاسخ.
- زمان پاسخ P1 کمتر است ولی نسبت پاسخ دومی خیل کمتر است 70/60«5/1

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مفاهیم طراحی در الگوریتم‌های تخصیص: ۵ موضوع وجود دارد ص ۱۹۹
- الگوریتم‌های قابل پیش‌بینی در برابر غیر قابل پیش‌بینی.
- الگوریتم‌های متمرکز در برابر توزیعی.
- الگوریتم‌های بهینه در برابر نیمه‌بهینه.
- الگوریتم‌های محلی در برابر سراسری.
- الگوریتم‌های وابسته به ارسال کننده در برابر وابسته به دریافت کننده.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- گر چه کمتر سیستمی قابل پیش‌بینی است ولی تقریب‌هائی براساس آمار گذشته می‌توان بدست آورد.
- وضعیت رزرو جا در روز خاص در فصل خاص بین دو شهر خاص، تعداد مشتریان بانک و نوع کار آنها.
- خیلی وقتها اینکار اصلاً امکان‌پذیر نیست زیرا معلوم نیست چه کسی چه کاری انجام می‌دهد.
- در مورد دوم: وجود عناصر متمرکز همیشه مشکل‌زاست ولی بعضی موارد چاره‌ای جز این نیست.
- در مورد سوم: رسیدن به نتیجه بهینه گر چه امکان‌پذیر است ولی پیچیده است لذا:
  - در عمل الگوریتم‌های غیرقابل پیش‌بینی، توزیع شدهٔ نیمه بهینه استفاده می‌شود.
- مورد چهارم: آیا پروسس جدید روی ماشین خودش اجرا شود یا بجای دیگر ارسال گردد.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- سیاست انتقال:
- دید اول : ماشین محلی خلوت (مقدار کارزیر یک آستانه)؛ اجرا در محل و گر نه به جای دیگر برود.

Trans For Policy
سیاست انتقال

- پیاده‌سازی ساده ولی الگوریتم خیلی خوش بینانه و ناپخته (Crude).
- دیده دوم: جمع‌آوری اطلاعات سراسری (Global) سپس تصمیم گیری.
- دید دوم نتیجه بهتر و بهینه‌تر می‌دهد ولی پیاده‌سازی مشکل‌تری دارد.

- مورد آخر: سیاست جایابی Location Policy نمی‌تواند براساس اطلاعات محلی باشد.
- می‌توانند پروسسورهای محلی (ارسال کننده) بدنبال پروسس بیکار بگردند شکل ۱۶-۴ ص ۲.۱.
- می‌توانند پروسسورهای دیگر اطلاع دهند که بیکارند و دنبال کار بگردند شکل b.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مفاهیم پیاده‌سازی الگوریتم‌های تخصیص پروسسور
- اندازه گیری بار هر پروسس محلی: شمارش تعداد پروسس‌های موجود که زیاد جالب نیست زیرا:
- در یک پروسسور بیکار می‌تواند تعدادی پروسس **Mail** و **News** و **Clock** و .... باشد.
- روش بهتر شمارش تعداد پروسس‌های در حال اجرا (زنده) یا آماده اجرا، همان بالائی‌ها هم می‌توانند در این گروه باشند.
- روش دیگر محاسبهٔ درصد کار کردن **CPU** است با وقفه‌داد دوره‌ای و چک کردن وضعیت.
- در این روش وقتی وقفه‌ها ماسک شوند کار خراب می‌شود و درصد دارای تقریب زیادی است.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- چگونگی در نظر گرفتن هزینه‌های بالاسری است (مثال پژوهشکده) دو برابر کردن CPUها لزوماً بازدهی دو برابر نمی‌شود.
- در نظر گرفتن این هزینه کار ساده‌ای نیست و در موارد معدودی انجام می گیرد.
- فاکتور دیگر پیچیدگی الگوریتم است که کمتر کسی به این موضوع پرداخته است، مقایسه ۳ الگوریتم در ص ۲.۳.
- ۱) برداشتن تصادفی یک ماشین و ارسال کار، آن هم می‌تواندهمینکار را بکند.
- همگام بودن از نظر بهنگام بودن اطلاعاتی درباره Loadها چون ماشین‌ها مستقل از هم عمل می کنند.
- Prob (2 کردن ماشینها تا یافتن ماشین کم کار یا رسیدن به ماکزیمم.
- Prob کردن n ماشین و انتخاب کم کارترین

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مثال‌هائی از الگوریتم ها
- الگوریتم قابل پیش‌بینی برمبنای تئوری گراف‌ها.
- پروسس‌ها معین، n تعداد پروسس k > تعداد پروسسور.
- نیاز به زمانبندی داریم بطوریکه ترافیک شبکه را حداقل کند.
- ساختن گراف: پروسس‌ها گره‌اند و رابطها میزان پیغام بین هر دو پروسس (وزن).
- هدف تقسیم گراف به چند زیر گراف که مقدار حافظه و CPU مورد نیاز زیر یک حد می‌نیمم باشد در هر زیر گراف.
- در اینجا ارتباطهای داخلی هر زیر گراف مهم نیست ولی درجه هر زیر گراف مهم است.
- مثال شکل ۱۷-۴ ص ۲.۴ دو مدل تقسیم‌بندی برای تخصیص ۹ پروسس به ۳ پروسسور.
- پیدا کردن بهترین جواب یک مسئله تئوری گراف است.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- الگوریتم متمرکز

- الگوریتم قبلی نیاز به اطلاعات کافی در شروع دارد.
- روش زیر (Up Down) این را نیاز ندارد.
- یک هماهنگ کننده جدول به کارگیری یک سطر برای هر W.S. دارد می‌سازد و اول کار صفر است.
- هر اتفاق مرتبط با تخصیص بیفتد به هماهنگ‌کننده گفته می‌شود تا جدول را بهنگام کند.
- تصمیم‌های تخصیص برپایه این جدول گرفته می‌شود :درخواست پروسسور، آزاد شدن آن، تیک ساعت، ....
- تخصیص بخش مناسبی از قدرت پردازش به هر کاربر بجای تضمین درصد بکارگیری بالا.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- وقتی یک پروسسور، پروسسی دارد با نیاز به اجرا در بیرون، از هماهنگ کننده درخواست می کند.
- هماهنگ کننده اگر پروسسور بیکار دارد می‌دهد و گرنه درخواست را ثبت می کند.
- پروسسورهای دارای کار روی سایر پروسسورها نمره منفی در هر ثانیه می گیرند که در سطرش توی جدول جمع می‌شود.
- وقتی درخواستی دارد که قابل اجرا نیست امتیازهای منفی از عنصر داخل جدول آن کم می‌شود.
- پس امتیازها بالا و پائین می‌روند (اسم روش) (+: درحال استفاده از منابع، - نیازمند منبع، عادی).
- پروسس آزاد شده به کسی که پائین‌ترین امتیاز را دارد داده می‌شود این خصوصیت الگوریتم است.
- در شرایط مختلف نتایج خوب داده.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- الگوریتم سلسله مراتبی

- وجود نقطه متمرکز (روش قبلی) همیشه مسئله‌ساز است بخصوص در سیستم‌های شلوغ.
- ساختن یک سلسله مراتب از ماشین‌ها مثل یک چارت سازمانی شکل ۱۹-۴ ص ۷-۲.
- بدلیل محدودیت ارتباطها، مدیریت ساده است، اجتناب از یک مدیریت در صدر و بریدن کلهٔ درخت. شکل ۱۹-۴.
- با خرابی یک عنصر در سطح **k** مدیران سطح **k-1** عنصری جایگزین آن می کنند از خودشان.
- کار غیرقابل انجام به بالا منتقل می‌شود و تا زمانیکه قابل اجرا شود آنگاه تقسیم‌بندی شده بطرف پائین تا برگ‌ها می‌رود.
- مشکلات این روش درخواست‌های همزمان، شرایط **Dead Lock, Race** خواهد بود.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- الگوریتم توزیعی وابسته به درخواست کننده
- ماشین حاوی پروسس بقیه را تا N تا **Probe** می کند تا یک ماشین مناسب پیدا کند
- و گرنه روی خودش اجرا می‌شود.
- در شرایط مختلف تست شده و نتیجه خوب داده،
- در یک سیستم شلوغ همه در حال ارسال **Probe** هستند!

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- الگوریتم توزیعی وابسته به گیرنده

- ماشین آزاد شروع به درخواست کار می‌کند تا N تا، اگر پیدا شد انجام می‌دهد.
- و گرنه به کارهای خودش می‌پردازد یا بیکار می‌ماند و مدتی بعد اینکار را تکرار می‌کند.
- اشکال قبلی را ندارد زیرا با **Probe** در زمان آزاد بودن همه ماشینها بالا می‌رود.
- تلفیق این دو الگوریتم هم امکان دارد با نگهداری سوابق گیرنده‌ها و استفاده از آنها در زمان نیاز.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- الگوریتم مزایده‌ای: عرضهٔ تقاضا در بازار

- خریداران پروسس‌ها هستند و فروشندگاه CPUها که سیکل‌هایشان را به بالاترین نرخ اعلام شده می‌دهند.
- پروسسورها قیمت خود را روی یک فایل عمومی می گذارند با آخرین نرخ استفاده شده و امکانات خود.
- پروسس‌ها قیمت‌ها را می گیرند و آنچه برایشان مناسب است انتخاب می کنند و قیمت می‌دهند.
- پروسسورها از بین قیمت‌های اعلام شده بالاترین را جواب **OK** می‌دهند و بقیه **Not OK**.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

۴-۴ زمانبندی در سیستم‌های توزیع شده

- معمولاً هر پروسسور، پروسس‌های خود را زمانبندی می‌کند بدون توجه به وضعیت بقیه پروسسورها.
- در حالت پروسسورهای خیلی وابسته روی پروسسورهای متقاوت روش کارآئی نیست.
- مثال شکل ۴-۲. a ص .۲۱ که D, A نیاز به ارتباط دارند ولی در سیکل‌های زوج و فرد اجرا می‌شوند.
- راه حل اجرا پروسس‌های مرتبط در یک سیکل ولی کار ساده‌ای نیست.
- پروسس‌های مرتبط معمولاً با هم شروع می‌شوند این می‌تواند یک راهنمائی باشد.
- روش (Coscheduling) همزمانبندی ماتریس شکل ۴-۲. ص .۲۱.
- پروسس‌های مرتبط در یک Slot زمانی روی پروسسورهای مختلف اجرا می‌شوند(بطور همزمان).
- امکان ادغام این روش با روش سلسله مراتبی و تشکیل ماتریس توسط هر مدیر برای کار گرانش.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

۴-۵ تحمل خرابی‌ها

- خرابی در سیتسم‌های Salty Critical بویژه خیلی مهم است و باید پیشگیری شوند.
- خرابی اجزاير ماشین: می‌تواند به دلایل بسیار زیادی باشد ص ۲۱۲ و همیشه به توقف سیستم منجر نمی‌شود.
- خرابی گذرا، متناوب و همیشگی طبقه‌بندی می‌شود.
- نوع گذرا: با تکرار عمل حل می‌شود (شبکه‌ها و گم شدن بیت).
- نوع متناوب: از همه بدتر است وقتی دکتر بیاید عیب می‌رود، ماشین و مکانیک.
- نوع همیشگی: تا تعمیر جزء معیوب می‌ماند.
- خرابی در همه سطوح از بیت گرفته تا پروسسور و نرم‌افزارها امکان دارد.
- اگر احتمال وقوع خط در یک ثانیه p باشد، متوسط زمان بین دو خطا است $\frac{1}{p}$ ص ۲۱۳.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- خرابی سیستم
- قابلیت اعتماد در سیستم‌های توزیع‌شده خیلی مهم است. زیادتر بودن تعداد اجزاء احتمال خرابی را هم بالا می‌برد.
- در نوع خطای **Fail-Silent** که سیستم متوقف می‌شود و نه ورودی قبول می کند نه پاسخ می کند.
- نوع دوم .... **By** که سیستم کار می کند ولی جواب درست نمی‌دهد و یافتن آن بسیار مشکل‌تر است.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- سیستم‌های همگام و غیرهمگام
- در سیستمی که پیغام ارسالی حتماً در زمان T پاسخ داده می‌شود یا در همین مدت n بار تکرارش Timeout می‌شود آنرا همگام گویند. ماشین خراب زود شناسائی می‌شود (با مفاهیم عادی Sync و..... قاطی نشود).
- در سیستمی که اینطور نباشد غیرهمگام است و کار با آن خیلی مشکل‌تر است برای یافتن خرابی.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- کاربر و افزونگی (Redundancy)
- افزونگی می‌تواند در سه مورد افزونگی اطلاعات، افزونگی زمان و افزونگی فیزیکی استفاده شود.
- اطلاعات: اضافه کردن بیت‌های اضافی برای تصحیح اشتباهات ش Hamming Code.
- زمان: تکرار عمل‌ها که جواب نداده است که در مورد اشکالات گذرا یا متفاوت بکار می‌رود.
- لازم است تکرار عمل اشکالی ایجاد نکند (عمل Idempotent).
- فیزیکی: اضافه کردن اجزاء شبیه بهم برای جایگزینی در زمان خرابی، تعدد پروسسور و دستگاههای جانبی.
- این مورد می‌توان تکرار عملی Active Replication (همه Serverها با هم کار کنند).
- یا می‌تواند Primrg buck up باشد که در صورت توقف جایگزین می‌شود.
- در چه تکرار لازم، متوسط و بدترین حالت عملکرد وقتی خطائی وجود ندارد، متوسط و بدترین عملکرد در هنگام بروز خطا ص ۲۱۵.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- کاربرد **Active Replication**
- تکنیک خیلی معمول در دنیای واقع هم وجوددارد، اعضاء بدن، چهار موتور هواپیما، چند تا داور در بازی .....
- به روش ماشینهای با حالات مشخص محدود **State Machine FSM** هم معروف است، **Mull Pl Server** این جوری‌اند.
- درمدارهای الکترونیک شکل ۴-۲۱ a که خوب نیست و b که خیلی تحمل‌پذیر خطاست (TMR) ص ۲۱۵.
- چه یک عنصر و چه یک **Voter** خراب شود اشکالی ایجاد نمی‌شود، در ورودی درست جواب درست می‌دهد.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مقدار تکرار؟ سیستم K Fcalt Tolerant یعنی k عضو خراب شود و باز سیستم کار می کند.
- در خطای Fcal Silently کافی است K+1 عنصر داشته باشیم تا K تحمل‌پذیر خطا باشد.
- K تا توقف کند هنوز یکی هست که کار کند.
- در حالت خرابی Byzantine باید 2K+1 عضو داشته باشیم که اگر K تا جواب غلط بدهند و K+1 جواب درست این جواب قبول شود.
- برای داشتن، FSM باید درخواست‌ها به همان ترتیب به همه Serverها برسد که حالات آنها یکسان باشد و جوابها یکسان باشد.
- استفاده از شماره ردیف سراسری یا ساعت سراسری و برچسب زمانی روی درخواست‌ها و اجزا به ترتیب برچسب زمانی.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- تحمل خطا با استفاده از **Primary – Backup**
- در هر لحظه یک **Server** اصلی است و دیگری **Backup**، در صورت توقف اصلی، پشتیبان بکار می‌افتد.
- مثال فایل شبکه تلفن‌ها.
- در زندگی روزمره هم استفاده می‌شود، معاونین، کمک خلبان، ژنراتور برق و .....
- مزایا نسبت به روش قبلی: اجراشدن یک کپی درهر حالت و ارسال پیام‌ها به یک مقصد بجای مقصد.
- پراسسور کمتری هم نیاز دارد.
- ولی در خطاهای **Byzan time** کار خراب می‌شود زیرا **Primany** کار می‌کند ولی نتایج غلط می‌دهد که معلوم نیست .

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مثال شکل ۴-۲۲ ص ۲۱۸، توقف در مراحل مختلف نتایج مختلف می‌دهد، خرابی اصلی قبل از ۲، و قبل از ۳ (دو باره کاری).
- یک موضوع هم جایگزین کردن Backup در چه زمانی؟ امکان تبادل اطلاعات با Server متوقف نیست.
- جایگزینی باید سخت‌افزاری شود، نوشتن هر دو پروسس روی یک دیسکت، تکرار دیسکت بصورت سخت‌افزاری.
- نگهداری Log برای ترمیم لازم است شاید هم تکرار Log.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- توافق روی سیستم خراب
- نیاز به توافق روی اجراء در حال کار در موارد مختلف، مثل Commit کردن، تقسیم کار و ....
- سه سئوال ص ۲۱۹ سطرح است، توافق هنگامی مشکل است که بعضی اجزاء متوقفند یا دست کار نمی‌کند.
- اول: فرض کنید پروسسورها کاملند ولی ارتباط اشکال دارد و پیامها گم می‌شود (دو جزء مهم پروسسورها و ارتباطات).
- مسئله پیداکردن دو پراسسور که در مورد دولتی یک لیست توافق کنند. Two-Army Problew
- داستان دو لشکر یکی وسط دره و دو تا روی تپه‌های دو طرف و تلاش برای ارتباط برای حمله به ارتش پائین دره.
- نیازه به یک تعداد دفعات می‌نیمم دارد که آخرین پیام خیلی مهم است.
- پس حتی دو پراسسور هم در شرایط خط ارتباطی ضعیف نمی‌توانند با هم توافق کنند.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- دوم: فرض کنیم خطوط ارتباطی درست است ولی پرواسسورها خرابند.
- باز هم با مسئله ارتش تطبیق می کند **Byzantion Genoral Problem**.
- ارتش داخلی دره بوسیله چندین گروه محاصره شده و ارتباط دوبدو کامل است ولی عده‌ای ژنرال‌ها نمی‌خواهند ژنرال‌های صادق کارشان را پیش ببرند.
- بجای توافق برای حمله بخواهند قدرت دفاعی خود را به دیگران اطلاع دهند (یک بردار nتائی پیش هر کس).
- شکل ۴-۲۳ ص ۲۲۱ اول ارسال نیرو از هر کس به هر کس که ۳ به همه دروغ می گوید.
- بعد ارسال بردار حجم شده از هر کس به هر کس باز ۳ به همه دروغ می گوید.
- بردارها مقایسه‌شده و اکثریت قبول می‌شود هر کدام اکثریت ندانست **Auknown** می‌شود و دروغگو مشخص می‌شود.
- در مثال شکل شکل ۴-۲۴ ص ۲۲۲ الگوریتم عمل نمی کند (بیش از $\frac{2}{3}$ پروسسورها باید درست کار کند).

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

۴-۶- سیستم‌های توزیع‌شده Real Time: خانوادهٔ دیگر سیستم‌های توزیع‌شده

- تعریف وابستگی به زمان در یک سیستم .R.T، عمل جذر روی دو سیستم، عمل پردازش سیگنال قلب روی همان دو سیستم.
- Fly-by-Wire، Drive-by-Wire، Shoot-by-Wire، مثال‌ها ص ۲۲۳، پخش کننده CD صوتی.
- محرک زمانی به کامپیوتر داده می‌شود: پریودیک هر Δt، Aperiodit فواصل نامعین ورود هواپیما به فرودگاه، Sporadic غیرمنتظره.
- مثال شکل ۴-۲۵ ص ۲۲۴ سه رخداد C, B, A پریودیک و X غیرمنتظره.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

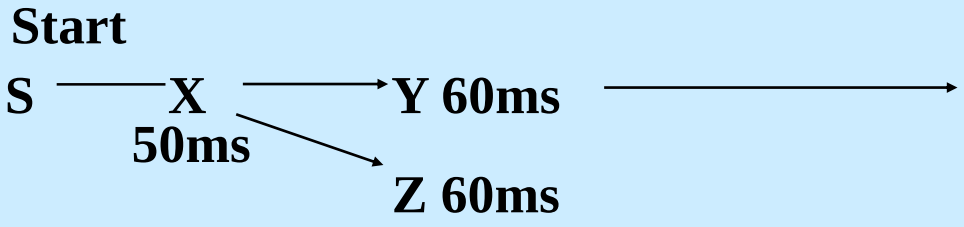
- حق ندارد رخدادی را از دست بدهد، می‌توان پراسسور خاص برای مدیریت پیام‌ها گذاشت.
- مثال شکل ۲۶-۴ ص ۲۲۵.
- در مدل Soft RT امکان از دست دادن کسری از پیامها هست سیستم تلفن روتینگ، ۱ در ۱.۵ تا طوری نیست.
- در مدل Hard RT ابداً نباید چیزی ازدست برود (کنترل موشک) باعث فاجعه می‌شود.
- تلفیق ایندو هم هست.
- موهومات RT: نوشتن Diviu Driver به زبان اسمبلی، محاسبات بسیار سریع.
- کامپوترهای خیلی سریع باعث مترو که شدن RT می‌شود.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مفاهیم طراحی
- همگام کردن ماشینهای سیستم RT با داشتن Clockهای مختلف: در فصل ۳ گفته شد.
- سیستم برپایه رخداد در مقابل سیستم برمبنای زمان.
- برپایه رخداد پیاده‌سازی ساده‌ای دارد برای S.RT مناسب است ولی در شرایط بار کاری زیاد شکست می‌خورد (وقفه رانشی)
- بوجود آمدن شرایط Euent Shower.
- برمبنای زمان.
- هر Δt اهمیت دارد کم باشد تعداد وقفه زیاد می‌شود، زیاد باشد رخدادها از دست می‌روند.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مثال آسانسور در دو حالت بالا، انتخاب نوع مدل به نوع کار بستگی دارد.
- طبقه ۹۹ است اول طبقه ۱ بعد طبقه ..۱ را فشار می‌دهد.
- برمبنای رخداد **Faster Response at Low Load**.
- برمبنای زمان سیستم قابل پیش‌بینی است، برعکس بالا.
- قابل پیش‌بینی بودن - سیستم حتی در پربارترین حالت باید **Dead Line**ها را بپذیرد.
- وابستگی رخدادها بهم موضوع دیگری است که در صورت قابل پیش‌بینی بودن قابل برنامه‌ریزی است بدترین حالت را باید بدانیم.



فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- تحمل خطا: در سیستم RT خیلی اهمیت دارد
- روش Active Replication خیلی استفاده می‌شود به شرطی که اجراء سیستم بتواند در موارد لازم با هم توافق کنند.
- Primany Backup بدلیل احتمال از دست دادن رخدادها در زمان تحویل سیستم، کمتر استفاده می‌شود.
- تلفیق ایندو: یک ماشین تصمیم گیرنده بقیه اجرا کننده بکار می‌رود.
- تحمل خطا باید در شرایط کاملاً استثنائی هم کار کند، همهٔ اجزاء بتوانند ازکار بیفتند و بار کاری هم حداکثر باشد.
- داستان قطع برق و از کار افتادن تلفن.
- Fail Safe بودن در زمانیکه احتمال فاجعه است، سکوی نفتی – سیستم قطار زیرزمینی فاصله کافی برای ترمز اضطراری.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- پشتیبانی زبان
- امکان استفاده از زبانهای عادی نظیر C که معمول می‌شود.
- بهتر بودن وجود زبان خاص RT برای تعریف رشته‌های کوچک و کوتاه.
- امکان محاسبهٔ حداکثر زمان اجرا در زمان کامپایل.
- نمی‌توان حلقه While یا For با مقادیر متغیر داشت، همه چیز باید ثابت باشد و بازگشت هم ممکن نیست.
- یک نکته تیک ساعت است که طول آن و تعداد بیت آن مهمند شکل ۴-۲۷ ص ۲۲۹، ۶۴ بیت یک ns.
- حداقل و حداکثر تأخیرها هم باید قابل اعمال باشد ولی اعمال حداکثر مشکل یا غیرممکن است.
- اگر اتفاقی که منتظر آن هستیم رخ ندهد هم باید راهی برای آن گفته شود.
- Timeout شدن یا بلوکه شدن وقتی به مدت کافی متنظر شد.
- یک دستور شبیه ص .۲۳- Euory…sec do هم لازم است برای رخداد پریود یک.
- محاسبه زمان لازم برای اینگونه سرویس‌ها و تعداد ماشین لازم برای کل کار.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- ارتباطات در DRT
- کلیدموفقیت: قابل پیش‌بینی بودن و معین بودن، قابل پیش‌بینی بودن ارتباط بین پراسسورها.
- پاکت‌ها ممکن است با هم برخورد کنند، ماشین‌ها همه صبر کنند و دوباره پاکت‌ها را بفرستند (در Ethernet) و هیچ حدی روی سیستم نیست که بدترین حالت را نشان دهد.
- حال در Tokenring هر نور صبر می‌کند تا Tocen برسد و پیامش را به آن بچسباند.
- حداکثر n بایت در هر Token برای K ماشین هر پاکت اورژانسی بعداز زمان Kn بایت می‌رسد.
- روی Tokenها می‌توان اولویت هم گذاشت، پاکت را که می‌گذارد برای خودش یا اولویت بالاتر رزروش می‌کند که در دور بعدی به خودش یا اولویت بالاتر برسد.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- روش Time Division Multipl Access یا TDMA
- تقسیم ترافیک به تعدادی Frame با اندازه‌سازی هر پراسسور دارای یک بخش از فریم که می‌تواند استفاده کند شکل ۴-۲۸ ص ۲۳۱.
- در WAN هم همهٔ موارد بالا باید رعایت شود در حالیکه ارتباط هم Connection Orieuted است.
- قابلیت‌های شبکه هنگام پیاده‌سازی توافق می‌شود مثل حداکثر تأخیر، حداقل عرض خط و .....
- امکان رزرو بافر و فضای جدول، زمان CPU و ..... هم برای تضمین پاسخ هست.
- استفاده از این منابع مسلماً دارای هزینه است.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مسئله مهم در WARTDS گم شدن پاکت‌هاست.
- به اندازه Δt بعد از ارسال اگر ack نیامد ارسال مجدد می‌شود.
- این انتظار بدون حد و مرز در سیستم RT پذیرفته نیست.
- یک راه ارسال موازی ۲ یا چند پاکت شبیه بهم برای تضمین رسیدن.
- احتمال عدم ارسال 10^{-5} یا یک در 10^{5} گم‌شدگی با در خط یک در 10^{10}.
- اگر یک پاکت یک Msec بخواهد، در هر ۴ ماه یک گم‌شدگی داریم و با سر خط در هر ۰/۰۰۰۳ سال.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- پروتکل با محرک زمانی یا Time Trigyered Prutocal TTP پیاده شده در MARS
- یک نود حاوی حداقل یک CPU و معمولاً چند تا بعنوان نور متحمل خطا و Fail Silent.
- ارتباط با دو خط TDMA که پاکت‌ها همزمان روی آنها ارسال می‌شود و زمان گم شدن یک پاکت هر ۳۰ میلیون سال است.
- سیستم با یک ساعت تحریک می‌شود و هر µsec یک تیک دارد، دقت مضاوب 10ms است.
- همه نودها از برنامه‌های یکدیگر باخبرند و می‌دانند کی یک پاکت از کی نود ارسال می‌شود که Detect کنندش.
- عزم وجود یک پاکت یعنی فرستنده‌اش متوقف شده با توجه به احتمال گم‌شدن خیل کم.
- مثلاً نود ۶ قرار است پس از مقداری محاسبات جوابی را در طول ۲ ثانیه در Slot شماره ۱۵ قرار دهد (TDMA).
- اگر پیام یافت نشد بقیه نودهای فرض می کنند نور ۶ خوابیده است و باید چاره‌ای بیندیشند.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- اطلاعات همه باید مثل هم باشند اگر نبود خطای فاحشی اتفاق افتاده.
- وضعیت سراسری که هر نور آنرا دارد شامل حالت فعلی، ساعت سراسری و حالت عضویت سیستم است.
- **Mode** (حالت) مثل شمارش معکوس، شلیک، پرداز و فرود در کنترل موشک.
- هر **Mode** نیازبه جزئیات مثل لیست نودهای همکار، آرایش **TDMA**، اسم و فرمت پیام و .... می‌باشد.
- دومی ساعت سراسری است که واحد شمارش بستگی به کاربرد دارد.
- شکل ۴-۲۹ یک نمونه پاکت که شامل فقط یک **Lager** است از **end** تا **end**.
- سومی نشان می‌دهد کدام نود خوابیده کدام کار می‌کند.
- هر چند دقت یکبار یک پیغام **Intialize** می‌آید که نودهای از دور خارج شده می‌توانند وارد دور شوند.
- بدلیل وجود زمان وصول در پاکت و دانستن آن توسط گیرنده ساعت‌ها می‌توانند **Sgnc** شوند.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- زمانبندی RT
- متشکل از مجموعه‌ای از پروسس‌ها یا رشته‌های کوتاه با کار مشخص و حد زمانی مشخص.
- تصمیم گیری برای اجرای چه پروسسی روی چه پروسسوری.
- انواع زمانبندی HRT در مقابل SRT (اولی حتماً در زمان تعریف شده تمام شود).
- انواع زمانبندی انحصاری در مقابل غیرانحصاری (Dreemptive).
- انواع دینامیک در مقابل استاتیک (اولی تصمیم گیری در زمان اجرا، دومی تصمیم گیری قبل از اجرا).
- انواع دینامیک متمرکز در مقابل غیرمتمرکز (تصمیم گیری‌ها روی یک ماشین دومی غیرمتمرکز).

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- سئوال چگونه همهٔ شرایط را داشته باشیم.
- مثلاً ۶۰ تا دقیقه در ثانیه هر وقفه ۵۰ میلی ثانیه حال چه کنیم!
- M تا کار و n تا پروسسور ci زمان CPU و pi پریود ان بین دو وقفه.

برای امکان‌پذیری باید داشته باشیم $\mu = \sum_{i=1}^{m} \frac{Ci}{Pi} \leq N$ فاکتور بکارگیری

- هر ۲۰ میلی ثانیه یک کار ۱۰ ثانیه‌ای نیاز به CPU_2^1 دارد – قابل برنابندی بودن؟
- هزینه بالا سری فراموش شده. مثال سیستم رادار.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- زمانبندی دینامیک: الگوریتم **Rate Monotonic** غیر انحصاری بدون شرایط ترتیب و در ..... ناسازگاری یک پردازنده
- اولویت مناسب است با فرکانس اجرا، تکلیف هر ... اولویت .۵ دارد و تکلیف هر **100mS** اولیوت .I دارد.
- انتخاب بالاترین اولویت و جایگزینی آن با تکلیف جاری.
- ثابت شده که کارها قابل زمانبندی هستند اگر $\mu = \sum_{i=1}^{m} \frac{Ci}{Pi} \leq N$
- سمت راست با $m \to \infty$ می‌رود بسمت **ln 2=0.643** (همان **SPT** است).

این عدد ..... است و تا عدد **0.88** قابل زمانبندی است(تجربی)

- دو الگوریتم دیگر هم هست خودشان بخوانند.
- زودترین ..... اول .....

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- زمانبندی استاتیک: زمانبندی قبل از شروع اجرا (اولین فصل تدوین شده)
- جستجو برای یافتن زمانبندی بهینه دقت زیادی می‌خواهد **Exponoution**، از کلک استفاده می‌شود.
- شکل ۴-۳۰ ص ۲۳۸ یک کار که با یک رخداد بیرونی شروع می‌شود کارهای محلی یا از راه دور دیگر را شروع می کند.
- بعد از ۱ می‌تواند۲ یا ۳ را شروع کند، بعد از ۲و ۳ یا ۴ را.
- ترتیب‌ها را روی **A** و **B** هر دو وجود دارند.
- شکل ۴-۳۱ در نظر گرفتن ارتباطات هم، همه چیز قابل پیش‌بینی است اگر اشکال غیرمنتظره پیش نیاید.
- توقف‌ها هم می‌تواند با تکرار **CPU** (دو تا در هر طرف) که همزمان کار کنند حل شود و همینطور خطوط موازی.
- نتیجه: در سیستم **RT** هیچوقت از حداکثر امکانات استفاده نمی‌شود.

فصل چهارم: پردازش‌ها و پردازنده‌ها در سیستم‌های توزیع‌شده

- مقایسه زمانبندی دینامیک و استاتیک
- استاتیک برای طراحی با محرک زمانی و دینامیک برای طراحی با محرک بصورت رخداد.
- در استاتیک مقدار کار قبل از شروع اجرا زیاد است ولی در دینامیک همه چیز بعد از اجراست.
- در دینامیک استفاده بهینه‌تر از منابع می‌شود زیرا برنامه‌ریزی استاتیک باید بدترین حالت‌ها را در نظر بگیرد.
- ولی کلاً در HRT منابع باید تا حدی آزاد باشد که بتوان Dend Lineها را رسید.
- در استاتیک قبل از اجرا وقت کافی هست که حالت بهینه پیدا شود (مثال راکتور اتمی).