

# سیستم عامل پیشرفته

محمد داوریناه جزئی

ترم دوم ۹۴-۹۳

گروه مهندسی کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی فولاد

# فصل سوم: همگامی در سیستم‌های توزیع شده

- در سیستم توزیع شده همهٔ داستان ارتباطها نیستند بلکه همگامی و نحوهٔ آن هم اهمیت دارد.
- نوامی بحرانی، تخصیص منابع، دو به دو ناسازگاری، ارتباط و همکاری بین پروسس‌ها (IPC).
- روشهای مربوط به سیستم پردازنده مثل سمافور و مانیتور در سیستم توزیعی به درد نمی‌خورد. زیرا منبع حافظه مشترک است.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

## ۱- همگام‌سازی ساعت

- هدف جمع‌آوری اطلاعات در یک جا و تصمیم‌گیری در مورد آنها نیست (الگوریتم‌ها توزیعی است).
- فصوصیات الگوریتم‌های توزیعی ص ۱۱۹.
- ۱- پخش بودن اطلاعات بین ماشین‌ها      ۲- تصمیم‌گیری پروسس‌ها بر مبنای اطلاعات محلی
- ۳- عدم قبول حتی یک نقطه شکست      ۴- عدم وجود یک منبع زمانی عمومی
- سه تای اول  $\equiv$  عدم قبول وجود عناصر مشترک.
- سیستم توزیعی باید قابل اعتمادتر از تک سیستم باشد-توقف یک ماشین باید همان ماشین را متوقف کند.
- پس همگامی بدون تمرکز مورد نیاز است!
- زمان در سیستم متمرکز یکی است ولی در توزیعی فیر مثال Make زبان C در UNIX شکل ۳-۱ ص ۱۲.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- ساعت‌های منطقی
- وجود کریستال، Holding reg, Counterreg, Clock tick.
- تفاوت سرعت کریستال‌ها در CPUها  $\Leftarrow$  تفاوت ساعت : Clock Skew.
- آنچه مهم است زمان مطلق نیست بلکه نسبت زمانها به یکدیگر است
- فقط پروسس‌های وابسته باید Ssync باشند.
- مهم توافق زمانی بین ماشین‌هاست که لازم نیست با زمان واقعی مساوی باشد ، به این زمان منطقی گویند.
- ساعت منطقی در مقابل ساعت فیزیکی

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم لمپورت
- $a \rightarrow b$  یعنی  $a$  قبل از  $b$  اتفاق افتاده (در یک پروسس یا ارسال و دریافت بین دو پروسس) ص ۱۲۲
- $x$  و  $y$  را همزمان گویند اگر  $x \rightarrow y$  باشد یعنی دو حالت بالا را نداشته باشیم.
- $a$  قبل از  $b$  عمل واگیری است.
- نیاز به تابع زمان  $C(x)$  داریم که پس همه پروسس‌ها  $a \rightarrow b \Leftrightarrow C(a) < C(b)$ .
- زمان همیشه افزایشی است و اصلاح آنهم افزایشی است نه کاهش.
- شکل ۳-۲ ص ۱۲۳: پیغام  $C$ , مشکل کلی دارد، بقیه در محاسبه زمان ارسال مشکل دارند.
- شکل ۳-۲ ص ۱۲۳: مشکل را با الگوریتم لمپورت حل کرده. افزودن ساعت اگر  $C(b) < C(a)$  به اندازه  $C(a)+1$ .

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- برای زمانهای مساوی، شماره پروسس به راست مان اضافه شود.

- با شرایط زیر می‌توان تفصیص زمان در سیستم توزیعی دارد:

1.  $a \rightarrow b$  در یک پروسس پس  $C(a) < C(b)$ .

2. ارسال A و دریافت یک پیغام است پس  $C(a) < C(b)$ .

3. برای تمام رویدادهای  $a, b$  داریم  $C(a) \neq C(b)$ .

- امکان ترتیب زمانی فراهم شد.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- زمان فیزیکی
- زمانهای لمپورت با زمان واقعی تطبیق ندارد و در سیستم‌های بلادرنگ اشکال دارد.
- دو مشکل: تطبیق همه ساعت‌ها با ساعت فیزیکی، تطبیق همه ساعت‌ها با یکدیگر.
- مفهوم transit of the sun (ظهر) روز فورشیدی (فاصله دو ظهر در نجوم) شکل ۳-۳ ص ۱۲۵.
- ۳۰۰ میلیون سال پیش سال اندازه ۱۴۰۰ روز بوده! البته حالا ۳۶۵ روز طولانی‌تر شده.
- احتساب تعداد روز بیشتر و محاسبه متوسط ثانیه فورشیدی.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- ساعت اتمی: ماوی سزیم ۱۳۳.۱: ۹/۱۹۲/۶۳۱/۷۷ ترانزیشن، در ۵.۰ آز وجود دارد.
- BIH زمانها را می‌گیرد و سپس متوسط گیری می‌کند و ساعت اتمی بین المللی TAI را می‌دهد. تعداد تیک از ۱/۱/۱۹۵۸.۱. فیزیکدانها رئیس شدند!
- یک روز اتمی  $3\mu\text{Sec}$  کوتاه‌تر از متوسط روز فورشیدی است، ایجاد مشکل در طول زمان.
- در سال ۱۵۸۲ پاپ گریگوری سیزدهم ده روز از تقویم حذف کرد.  $\Leftarrow$  دعوا شد.
- شکل ۳-۴ نحوه همگامی TAI و زمان فورشیدی و معرفی UTC بجای GMT.
- تا کنون ۳ تا ثانیه کبیسه داشته‌ایم، زمان وجود ۸۰۰ میلی ثانیه افتلاف بین UTC و GMT.
- چند رادیو و ماهواره‌ها UTC را اعلام می‌کنند ولی زمان انتقال پیام را هم باید در نظر گرفت.



# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- نیاز به احتساب این موارد در سیستم عامل، الگوریتم همگام‌سازی ساعت
- اگر یک ماشین WWV (گیرنده UTC) داشته باشد بقیه می‌توانند همگامش شوند.
- اگر نه نیاز به الگوریتم همگام‌سازی هرچه دقیقتر داریم. فرض  $H$  وقفه در ثانیه و ساعت  $C$
- در زمان  $t$  روی ماشین  $p$  زمان  $C_p(t)$  است ایده‌آل:  $C_p(t) = t \forall p, t$  یا  $dC/dt = 1$
- در عمل دقیقاً  $H$  وقفه در تایمرها نداریم در نتیجه با دقت  $10^{-5}$  داریم  $10^{-5} - 10^{-6}$  بجای  $10^{-6} \dots = 10^{-6} \times 36 \dots$
- $1-p \leq dC/dt \leq 1+p$   $p = \text{Maximum Drift Rate}$  حد مجاز، شکل ۵-۳ ص ۱۲۸
- دو ماشین با Drift مخالف پس از  $\Delta t$  دارای  $2P\Delta t$  اختلاف. اختلاف کمتر از  $\delta$  همگامی هر  $\delta/2p$

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم کریستین
- سنکرونیزه کردن بقیه ماشین‌ها با یک ماشین که WWV دارد (Time Server).
- هر  $\delta/2p$  پیغام به TS دریافت زمان هر چه سریعتر ( $C_{UTC}$ ) ش ۶-۳ ص ۱۲۹
- مشکل اصل ساعت فرستنده عقب است و محلی  $C_{UTC} < C$  و کار فراب می‌شود.
- بجای اصلاح یکجا افزایش کمتر در هر وقفه (..۱ تا در ثانیه) بجای 10ms مثلا 9ms یا 11ms
- مشکل فرعی زمان ارسال پاسخ است، افزودن یک تقریب به زمان ارسالی ش ۶-۳ ص ۱۲۹.
- اگر مدت سرویس وقفه (I) را بدانیم تقریب ( $T_1 - T_0 - I$ ) زمان طی یک طرف
- راه دیگر تحلیل تعدادی ارسال و دریافت، حذف مقادیر عجیب و معدل گیری بقیه، تعیین می‌نیم بعنوان زمان واقعی.
- مسائل دیگر چون مسیرهای مختلف ارسال و دریافت، بار شبکه، ...

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم بر کلی
- برای حالتی که WWV نداریم، یک ماشین Time Deamon ساعتش دستی ست شود.
- بقیه ماشین‌ها را Poll کند، دستور تنظیم به عقب یا جلو بدهد شکل ۷-۳ ص ۱۳.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم معدل گیری
- الگوریتم‌های بالا عنصر متمرکز دارد. الگوریتم با حالت غیرمتمرکز هم وجود دارد.
- تقسیم زمان به فواصل  $T_0+iR$ ,  $T_0+(i+1)R$ .
- Broad Cast زمان در شروع هر فاصله از هر ماشین (همه در یک لحظه نیست).
- ساده ترین روش، جمع‌آوری زمانها در هر ماشین و معدل گیری از همه
- روش دقیقتر، حذف تعدادی از بزرگترین و کوچکترین‌ها (ذباله) و معدل گیری بقیه.
- افزودن زمان انتقال هم الگوریتم را دقیقتر می‌کند.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- داشتن چند منبع زمانی فاربی (از WWV، GEOS، .....).
- با توجه به شرایط شبکه باید برای UTC فاصله زمانی تعریف کرد بجای زمان مطلق.
- زمانهای دریافتی باید معدل گیری شود. هیچ پردازندهای پاسخ را فوری نمی گیرد.
- ماشین‌ها فاصله زمانی خود را Broadcast کنند در شروع هر دقیقه.
- زمان دریافت هم بین منبع و پروسسورهای مختلف فرق می کند همینطور شرایط شبکه.
- تصادف Packetها روی شبکه هم تأخیر ایجاد می کند.
- پروسسور هم می‌تواند مشغول کاری باشد و برای مدتی نتیجه را نگاه نکند (در فصل ۱ بحث می‌شود).

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- بکارگیری ساعت‌های همگام
- پیدایش مکانیزم‌های همگام‌سازی ساعت‌ها روی شبکه در سال‌های اخیر.
- الگوریتم‌ها کثرت یکبار تحویل پیام.
- روش سنتی: شماره‌گذاری پیام‌ها و ذخیره شماره‌ها برای تشخیص پیام تکراری از پیام اصلی.
- مشکل: با قطع برق اطلاعات پاک می‌شوند و مقدار وقت باید آنرا نگهداشت.
- روش بهتر: هر پیام یک مهر زمانی و شماره مسیر و حفظ دیرترین مهر برای هر مسیر
- پیام‌ها با زمان کوچکتر صرف نظر می‌شود. چون تکراری‌اند.

مدا کثرت تفاوت ساعت – مدا کثرت عمر پیام – زمان جاری =  $G$

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- کلیه پیغام‌های قبل از زمان  $G$  به راحتی حذف می‌شوند.
- پیغام‌های با شماره مسیر نامشخص هم به همین ترتیب عمل می‌شود.
- بعد از هر دلتا  $t$  زمان جاری  $G$  روی دیسک ثبت می‌شود.
- بعد از هر  $\text{Crash}$  و سپس  $\text{Reboot}$ ،  $G$  از روی دیسک فوایده شود و دلتا  $t$  زمان توقف به آن اضافه می‌شود.
- باز هم پیغام‌های کهنه‌تر از  $G$  بعنوان تکراری حذف می‌شوند.
- ممکن است بعضی پیام‌های جدید هم حذف شوند ولی سیاست  $\text{At Most Once}$  است.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- **تطبيق Cache Consistency بر مبنای ساعت**
- در سیستم فایل توزیعی عمل Caching محلی برای تسریع استفاده می‌شود.
- در انجام دو تغییر همزمان مشکل عدم تطبيق Cache ها پیش می‌آید.
- یک راه تمیز دادن Cache برای خواندن و برای نوشتن است.
- مسئله و قتی است که نوشتنی بعد از یک خواندن انجام شود که Cache خواندن را باید Invalidate کرد، زمان سر بار دارد ولی با روشن همگامی ساعتها بر طرف می‌شود.
- روی هر فایل Cache شده یک Lease گذاشته می‌شود (مدتی که Valid است).



# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- بعد از مدت فوق Client درخواست تجدید می‌دهد اگر هم نیاز نداشت که Cache را Timeout می‌کند.
- در آینده اگر نیاز داشت عدم تغییر آنرا از Server می‌پرسد (با مهر زمانی)، اگر نکرده یک Lease جدید رویش می‌گذارد.
- اگر نوشتن بعد از تعدادی فواندن انجام شود Server از همه Client های فواننده درخواست Timeout زودرس روی Lease ها می‌دهد.
- اگر مثلاً یکی هم Crash کرده Server صبر می‌کند تا Timeout گردد.
- برای اینکه نمی‌توان تفاوت آهسته بودن و Crash را فهمید در این روش Lease Timeout کار را حل کرده.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

## ۲- دو به دو ناسازگاری

- برای اطمینان از دسترسی انحصاری به داده‌های مشترک از نامیه بحرانی استفاده می‌شود.
- در تک پروسسوری با یک سمافور یا Monitor مشکل حل است.
- حالت چند کامپیوتری یا چندپروسسوری اینجا بحث می‌شود.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- یک الگوریتم متمرکز

- داشتن یک ماشین هماهنگ کننده برای نواحی بحرانی شکل ۸-۳ ص ۱۳۴.

- روش عادلانه، دارای پیاده‌سازی ساده و حاوی سه پیغام (Release, Grant, Req) است.

- هماهنگ کننده یک نقطه شکست است که مخالف قرار ما است، شلوغی آنها تنگنا فواید شد.

- اگر هماهنگ کننده مثل حالت b شکل جوابی ندهد Client نمی‌داند منتظر است یا Crash کرده.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- یک الگوریتم توزیع شده
- مبنای کار ترتیب زمانی درست بین هر جفت رویداد است مثل روش لمپورت در بخش قبلی.
- هر پروسس قبل از ورود به ناحیه بحرانی پیغامی حاوی نام ناحیه id خودش و زمان ورود به همه می‌فرستد.
- باید روش Reliable، (با Ack) باشد، روش پیام گروهی هم می‌تواند استفاده شود.
- پروسس گیرنده پیام با توجه به وضعیتش یکی از کارهای زیر را انجام می‌دهد (ص ۱۳۶).
  1. توی ناحیه نیست و نمی‌خواهد وارد شود: پیغام OK برمی‌گرداند.
  2. توی ناحیه است جواب نمی‌دهد و درخواست را Queue می‌کند.
  3. اگر قبلاً درخواست ورود داده ولی هنوز وارد نشده مهر زمانی آنکه زودتر است برنده می‌شود.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- هر پروسس درفواست کننده صبر می‌کند همه OK بدهند و وارد می‌شود وقتی هم فارم شد به آنها هم که در صف هستند OK می‌دهد و صف را فالی می‌کند.
- شکل ۹-۳ ص ۱۳۶ حالت تصادف را نشان می‌دهد.
- اینجا  $(n-1)$  پیام داریم با  $n$  پروسسور روشن متمرکز 3 تا، اینجا بدون نقطه شکست!
- ولی بجای یک نقطه شکست  $n$  نقطه شکست داریم) یعنی دو تا بدی دارد نسبت به متمرکز.
- توقف هر ماشین با انتظار تشفیص داده نمی‌شود-یک راه روشن Timeout است.
- مشکل دیگر نیاز به ارتباط گروهی است که ممکن است وجود نداشته باشد.
- مشکل بعدی اینکه بجای شلوغی یک پروسس همه پروسسها شلوغ می‌شوند.
- یک راه تضمین یک تعداد حداکثر OK بجای همه است.
- چرا استفاده می‌کنیم برای اینکه نشان دهیم الگوریتم توزیع شده امکان‌پذیر است.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم Token Ring
- شکل ۱-۳ ص ۱۳۹ شبکه BUS شکل a پروسس‌ها ترتیبی ندارند.
- در نرم‌افزار حلقه منطقی داریم که هر پروسس جای مشخص نسبت به بقیه دارد شکل b.
- هر پروسس بعدی خود را می‌شناسد.
- در شروع یک Token در حلقه رها می‌شود به پروسس صفر داده می‌شود.
- هر پروسس اگر وارد ناحیه بحرانی نشود Token را رد می‌کند به بعدی.
- هر پروسس اگر بخواهد وارد ناحیه بحرانی بشود Token را نگه می‌دارد تا از ناحیه فارغ شود آنگاه رها می‌کند.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- Starvation پیش نمی‌آید، حداکثر انتظار  $n-1$  تا مدت ورود و خروج به نامیه بحرانی است.
- دو تا اشکال: اولی گم شدن Token که پیدا کردنش سخت است بدلیل شرایط مختلف پروسس‌ها معلوم نیست هر پروسس چقدر Token را نگه می‌دارد.
- دوم شکست یک پروسس که باید توسط همسایه‌اش Detect شده و از شبکه حذف شود البته توسط همه.
- مقایسه سه الگوریتم
- شکل 3-11 ص 140 فاصله مقایسه‌ها

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

## ۳- الگوریتم‌های انتخاب

- موضوع انتخاب یک پروسس (هماهنگ کننده، ردیف کننده و ...) همیشه در سیستم توزیعی مطرح است.
- هر پروسس یک شماره شناسائی دارد (یگانه) معمولاً پروسس با بزرگترین شماره می‌شود هماهنگ کننده.
- فرض: همه پروسس‌ها شماره بقیه را می‌دانند.
- آنچه نمی‌دانند: کدام up و کدام Down.
- در یک الگوریتم: طوری عمل شود که همه روی انتخاب Coord توافق داشته باشند.



# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم بولی
- وقتی پروسسی بفهمد Coord جواب نمی‌دهد، درخواست انتخاب Coord می‌دهد.
- 1. P پیغام انتخاب به همه پروسس‌ها با شماره بالاتر از خودش می‌دهد.
- 2. اگر کسی جواب ندهد P برنده می‌شود و Coord می‌شود.
- 3. اگر یکی با شماره بالاتر جواب بدهد او کار را بدست می‌گیرد و کار P تمام می‌شود.
- در هر زمان هر پروسس می‌تواند از پروسس‌های زیر دست (با شماره کوچکتر) پیغام بگیرد.
- او جواب OK می‌دهد و کار تکرار می‌شود تا یکی برنده نهائی شود.
- آن یکی نتیجه را به همه اعلام می‌کند.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم رینگ
- هر پروسس می‌داند قبلی‌اش کیست.
- پروسسی که فهمید Coord فراب است شماره‌اش را به قبلی‌اش می‌دهد.
- آنها که متوقف‌اند skip می‌شوند.
- هر پروسس شمارهٔ خودش را به کمیت اضافه می‌کند.
- نهایتاً پیغام به شروع کننده می‌رسد.
- او پروسس با لاترین نمره را بعنوان Coord اعلام می‌کند.
- شکل ۱۳-۲ ص ۱۴۱

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

## ۴- تراکنش‌های اتمی

- آنچه تا کنون در مورد همگام‌سازی دیدیم فیلی سطح پائین بود (سمافور ، نوامی بحرانی، ...)
- برنامه‌نویس سطح بالا نیاز به تعاریف سطح بالا دارد. یک روش تراکنش‌ها یا اعمال اتمی است.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- مقدمه
- داستان عقد قرارداد که قبل از امضاء می‌توان بر گشت به جایی که هیچ اتفاقی نیافتاده.
- یک پروسس درخواست عملی توسط چند پروسس را اعلام می‌کند و منتظر می‌ماند با هم چونه بزنند.
- بعد شروع‌کننده درخواست فتم عملیات (Commit) می‌دهد اگر هم جواب OK بدهند.
- اینکار انجام می‌شود اگر نه (یکی جواب ندهد یا OK ندهد) همه برمی‌گردند به حالت قبل از OK و کلیه کارها پاک می‌شود (Roll Back).
- در DBMS به این استراتژی All or Nothing گویند که کار برنامه‌نویسان را راحت کرد.
- سیستم‌نوارهای قدیم حالت All or Nothing داشتند.
- در سیستم‌های مدرن مثال انتقال پول ۱- برداشت ۲- واریز (انجام هر دو در یک تراکنش (اتمی)).
- اگر اولی شد دومی نشد بتون بر گشت به قبل از ا.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- مدل تراکنش: سیستم = مجموعه‌ای از عمل‌های مستقل که هر کدام امکان شکست دارند.
- رده‌های حافظه: حافظه RAM، پاک شدن با قطع برق، دیسک، فطر crash.
- حافظه پایا، فقط فطر سیل و زلزله، ش ۱۵-۱۳ ص ۱۴۶ حالت b فرابی از a کی می‌شود.
- فرابی یک بلوک در اثر گرد و فاک حالت C، بلوک از کی درست اصلاح می‌شود.
- محیط مناسب برای محیط متحمل فرابی.
- اجزایی اولیه تراکنش: BT، ET، مرزها برای سیاست همه یا هیچ، AT، RD، WRT ص ۱۴۷.
- نوع دستورات اولیه به نوع کاربرد بستگی دارد مثال Send و Receive در سیستم Mail.
- مثال رزرو بلیط از WP به JFIC به NRB به MLD شکل 3-16 ص 148.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- خصوصیت‌های تراکنش ص ۱۴۸: اتمی، سازگار، ایزوله، پایا (ACID).
- هر تراکنشی اتمی است (All/Nothing) و در طول اجرا نتایج میانی آن قابل رویت نیست.
- یک فایل .. ا بایتی، در طول Append ... ا بایت به آن کماکان .. ا بایت است و در یک لحظه CMT .. ا بایت می‌شود. بدون توجه به اینکه چقدر زمان و چقدر کاربرده تا ... ا بایت اضافه شود.
- مثال دیگر کپی کردن که اگر با نباشد نصفه کپی شده هم پاک می‌شود. در Dos اینطور نبود.
- سازگاری: هر عمل بانکی موازنه مسابها را به هم نمی‌زند مگر در لحظات انجام تراکنش که قابل رویت نیست. مثال سیستم فایل و عملیات آن

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- ایزوله بودن (ترتیب‌پذیر): اعمال همزمان را با ترتیب فاصی بتوان انجام داد با همان نتایج
- امکان انجام شش ترتیب
- مثال شکل ۱۷-۳ ص ۱۴۹ که برنامه ۳ ترتیب‌پذیر نیست.
- مسئولیت ترتیب‌پذیری با فود سیستم است.
- پایا بودن: بعد از Commit نتایج همیشگی گردد.
- عدم تاثیر شکستهای بعدی روی نتایج

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- تراکنش‌های تو در تو
- تراکنش‌ها می‌توانند فرزند ایجاد کنند (Fork) و آنها هم بنوبه خود.
- یک فرزند Commit کند ولی پدر Roll Back کند، نتایج CMT باید پاک شود (undo).
- پس اصول بالا نقض می‌شود، این اصول باید برای تراکنش پدر صدق کند.
- نیاز به مدیریت فیلوی پیچیده
- هر پدر یک کپی فصوصی دارد، به هر فرزند هم یک کپی می‌دهد تا روی آن عمل کند
- در صورت CMT بایگزین کپی پدر، در صورت RLBK ناپدید
- فرزند بعدی نتایج قبلی را می‌بیند.



# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- پیاده‌سازی
  - انجام موارد ACID نیاز به مدیریت بسیار دقیق دارد.
  - روش فضای کاری فصوصی: انجام همهٔ اعمال روی کپی فصوصی و جایگزینی در لحظه CMT.
  - اشکال: هزینه کپی‌سازی همه Object‌های مورد نیاز.
  - بهینه‌سازی:
1. ایجاد اشاره گر به WS پدر، برای بالاترین سطح ا یجاد اشاره گر به نسخه اصلی برای فواندن و ایجاد یک کپی جدید برای عمل نوشتن. اشاره گره‌ای تو در تو
  2. کپی کردن ایندکس و ر کوردهای اصلاح شده (فقط) شکل ۱۸-۳ ص ۱۵۱.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- روش Write ahead Log (لیست قصدها)
- اول یک Log کامل درست می‌شود بعد تغییر روی کپی اصلی اعمال می‌شود  
شکل ۱۹-۳ ص ۲ ۱۵.
- اگر فواستیم بر گردیم مقادیر قدیمی را از Log بایگزین می‌کنیم Roll Back.
- وقوع شکست قبل از آخرین جمله و بعد از نوشتن Log آن قبل یا بعد از اعمال نهایی
- امکان رفتن به جلو و همچنین عقب بر مبنای log

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- روشن تایید دو فازی TWO Phase Commit یا CMT اتمی در سیستم توزیعی
- عمل CMT لحظه‌ای است، در سیستم توزیع شده عمل‌ها توسط چند ماشین انجام می‌گیرد و نیاز به هماهنگی دارد.
- یک پروسس نقش هماهنگ کننده دارد، قبل از شروع درخواست CMT یک Log می‌نویسد.
- بعد یک پیغام "آماده‌شو برای CMT" برای بقیه همکاران می‌فرستد.
- هر همکاری Log خود را می‌نویسد و پاسخ OK یا Not OK می‌دهد یا Timeout می‌شود.
- اگر همه OK دادند CMT و اگر نه RL BK می‌شود در Log هم نوشته می‌شود.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- شکل ۲-۳ ص ۱۵۳.
- با وجود log حافظه پایا، مقاوم در مقابل Crash.
- در شکست هماهنگ کننده بعد از نوشتن Log اولیه، پس از زنده شدن کار را دنبال می‌کند.
- بعد از نوشتن نتیجه رأی Crash کند، بعد از up شدن نتیجه را به بقیه می‌گوید.
- اگر یکی از دیگران جواب ندهد ارسال تکرار می‌شود یا Timeout شود.
- اگر بعد از دادن جواب سقوط کند از همانجا بعد ادامه می‌دهد.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- مکانیزم‌های کنترل همزمانی (الگوریتم‌ها): مواظبت از سر راه هم قرار گرفتن پروسس‌ها.
- قفل کردن: قدیمی‌ترین و پر استفاده ترین مکانیزم با استفاده از مدیر قفل مرکزی یا محلی.
- مدیر قفل با توجه به لیست فود، درخواستهای جدید برای فایل‌های قفل شده را رد می‌کند.
- قبل کردن و باز کردن توسط سیستم تراکنش انجام می‌شود بدون دفالت. User
- جداسازی قفل خواندن (Share) از قفل نوشتن. (exclusive)
- محدوده مورد قفل: رکورد، فایل، جدول، DB، صفحه یا بلوک سفت‌افزاری (مسائل مربوط).
- فرض ما: قفل روی فایل) که فوب نیست. Locking Granularity )
- دانه‌بندی ریزتر = توازی بیشتر = محدوده قفل کوچکتر = گرانتر = تعداد قفل بیشتر = Dead Lock = بیشتر.
- قفل دو فازی شکل 3-21 برای جلوگیری از Dead Lock.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- بهنگام‌سازی اصلی در مرحله تقلیل قفل انجام می‌شود، اگر قفلی سر راه بود صبر می‌کند تا آزاد شود.
- ثابت شده 2PL ترتیب‌پذیری را تضمین می‌کند.
- در 2PL قطعی فاز تقلیل بعد از CMT یا RLBK انجام می‌شود.
- دو مزیت:
  1. همهٔ فایل‌ها بعد از CMT یا RLBK شدن استفاده می‌شوند.
  2. افذ و آزادی قفل‌ها توسط سیستم انجام می‌شود.
- تراکنش‌های به هم گیر نمی‌کنند و Casca do Abort پیش نمی‌آید بدلیل استفاده از اطلاعاتی که نباید دیده شود.
- امکان Dead Lock هیچوقت از بین نمی‌روند حتی در 2PL.
- مکانیزم‌های پیشگیری که قبلاً گفته شده و مکانیزم زمانی (قفل بیش از T ثانیه = Dead Lock).

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- کنترل همزمانی فوشبینانه
- برو جلو هر کاری می‌فواهی بکن بی‌فیال بقیه، اگر مشکلی بود بعداً فکرش را می‌کنیم.
- فرمان CMT چک می‌شود اگر فایلی در طول کار عوض شده RLBK شده و کار تکرار می‌شود.
- برای فضای کاری محلی فصوصی مناسب است.
- توازی بسیار بالا، فرغ از Dead Lock، بدون انتظار برای قفل.
- عیب: تکرار کل کار، کندشدن در شرایط بار کاری شدید.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- مهر زمانی
- هر تراکنش در لحظه BT یک مهر زمانی می‌گیرد (الگوریتم لمپورت برای همگامی استفاده می‌شود).
- هر فایل دو مهر فواندن و نوشتن مربوط به آفرین تراکنش بکار برنده آن دارد.
- در شرایط عادی هر دسترسی بعد از مهر زمانی فایل است.
- شکل ۲۲-۳ ص ۱۵۷.
- در مقایسه با قفل دارای D.L. نیست که این مزیت اصلی است.

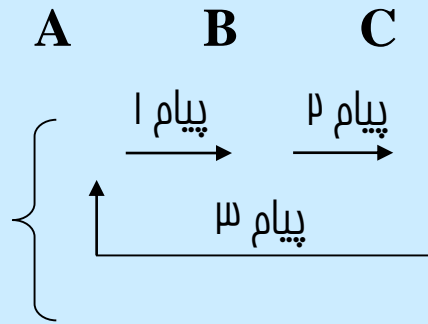


# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

## ۵- بن بست در سیستم‌های توزیعی

- D.L. در سیستم توزیعی به مراتب پیچیده‌تر است بدلیل پراکنده بودن اطلاعات مربوطه بین ماشین‌ها.
- بن بست ارتباطی: پیغام‌ها به هم گیر کرده‌اند و بافر فالی هم نیست.

در اینجا هر دو  
را یکی می‌گیریم.



- بن بست منبعی: رقابت در گرفتن منابع بصورت انحصاری.
- احتمال بروز بن بست ارتباطی خیلی خیلی ضعیف است.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- استراتژی‌های مهم (یادآوری DS1).
- 1. الگوریتم شترمرغ: فراموش کردن مسئله.
- 2. کشف: صبر کنم بن بست شود، کشف کنیم، ترمیم کنیم.
- 3. جلوگیری از وقوع: اکیداً پیش نیاید.
- 4. سعی در اجتناب: تفصیص با دقت لازم در جهت اجتناب از بن بست.
- مورد ۴ در سیستم متمرکز هم استفاده نمی‌شود چه رسد به توزیع شده چون فیلد مشکل است ۳.۲ بحث می‌شود.
- بخصوص مورد ۲ از همه آسان‌تر است که بعد از وقوع به فکر چاره باشیم.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- کشف بن‌بست توزیع شده
- بن‌بست کشف شده نیاز به یک قربانی برای حل دارد که User را ناراضی می‌کند.
- با تراکنش اتمی می‌توان کار قربانی شده را تکرار کرد با آرزوی موفقیت در دور دوم!

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- کشف بن‌بست متمرکز

- یک پروسس هماهنگ کننده همه گراف‌های بقیه ماشین‌ها را Union می‌کند.

- با کشف یک سیکل پروسس را Kill می‌کند.

- هر ماشین باید اطلاعات مورد نیاز را به پروسس هماهنگ کننده ارسال کند.

- حذف یا اضافه شدن هر arc ارسال شود، دوره‌های arc‌های اضافه یا حذف شده ارسال شود.

- هماهنگ کننده خودش هر وقت لازم بود درخواست اطلاعات کند.

- هر سه روش اشکال دارد. شکل ۳-۳-۱۶ ص ۱۶.

- بوجود آمدن بن بست نادرست بدلیل عدم دریافت بموقع پیام‌ها

سه  
روش

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- پیام آزادسازی R بوسیله B بعد از پیام نیاز B به T به هماهنگ کننده می‌رسد گر چه زودتر ارسال شده.
- هماهنگ کننده قبل از ایجاد حلقه به همه اعلام کند پیامی دادند بفرستند و بعد آنها را اعمال کند قبل از ایجاد حلقه.
- این روش فوب است ولی گران است.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم دیگر کشف بن‌بست توزیعی

- پروسس‌ها مجازند درخواست منابع چند گانه از جمله قفل کنند.

- فاز ترفیع قفل در 2PL سریع‌تر می‌شود.

- پروسس بجای چند بار انتظار برای چند منبع یکبار برای همه منتظر می‌ماند.

- شکل ۱۴-۳: اشکال اصلی در منابع بین ماشینی است.

- بردار (گیرنده، ارسال کننده، پروسس بلکه شده) در اثر احضار الگوریتم (پندی ..... ) ارسال می‌گردد.

- وقتی دوباره به اولیه ارسال کننده رسید D.L. کشف می‌شود مثال: بردار (0,8,0) آفر کار.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- راه اول: امر به فود کشی پروسس‌ها به آخر پیام و انتخاب پروسس با شماره بزرگترین برای فود کشی.
- توری و عمل هماهنگ نیستند: چگونه پروسس بلو که شده پیغام بفرستد!
- الگوریتم‌های زیادی ارائه و منتشر شده ولی بزودی دیگری غلط بودنش را اثبات کرده.
- این زمنیه‌ای است که هنوز جای کار زیاد دارد.

# فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- ممانعت از بن‌بست توزیع شده
  - عملکرد به نحوی که بصورت سافت‌تاری بین‌بست امکان‌پذیر نباشد.
  - روشهایی در OS1 بحث شده است رک به کتاب مربوطه.
1. فقط پروسسی مجاز به انتظار است که مهر زمانی‌اش کمتر است از پروسسی که برایش صبر می‌کند. اگر نه Abort می‌شود (الگوریتم Wait Die) شکل ۲۵-۳ ص ۱۶۴.
  2. فقط پروسسی مجاز به انتظار است که مهر زمانی‌اش بزرگ‌تر از پروسسی باشد که برایش صبر می‌کند.
- در هر صورت دنباله صعودی و یا نزولی‌ات که حلقه ایجاد نمی‌کند.
1. با جستجو در انتشارات در مورد قفل‌های مختلف موجود پروتکل‌های مربوطه برای قفل‌کردن و نحوه پیاده‌سازی استراتژی 2PL روی آنها حداقل دو صفحه مطلب بنویسید.
  2. قبل از ظهر از اطاق من بگیرید.