سیستم‌عامل پیشرفته

محمد داورپناه جزی

گروه مهندسی کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی فولاد

ترم دوم ۹۴-۹۳

فصل سوم: همگامی در سیستم‌های توزیع شده

- در سیستم توزیع شده همهٔ داستان ارتباط‌ها نیستند بلکه همگامی و نحوهٔ آن هم اهمیت دارد.
- نواحی بحرانی، تخصیص منابع، دو به دو ناسازگاری، ارتباط و همکاری بین پروسس‌ها (IPC).
- روشهای مربوط به سیستم پردازنده مثل .... 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 تصمیم گیری در مورد آنها نیست (الگوریتم‌ها توزیعی است).
- خصوصیات الگوریتم‌های توزیعی ص ۱۱۹.

۱- پخش بودن اطلاعات بین ماشین‌ها ۲- تصمیم گیری پروسس‌ها برمبنای اطلاعات محلی

۳- عدم قبول حتی یک نقطهٔ شکست ۴- عدم و جود یک منبع زمانی عمومی

- سه تای اول ≡ 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 بودن تمرکز مورد نیاز است!
- زمان در سیستم متمرکز یکی است ولی در توزیعی خیر مثال Make زبان C در UNIX شکل ۳-۱ ص ۱۲۰.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- ساعت‌های منطقی
- وجود کریستال، Clock tick, Counferreg , Holding reg.
- تفاوت سرعت کریستال‌ها در CPUها ⇐تفاوت ساعت : Clock-Tick Clock Skew.
- آنچه هم است زمان مطلق نیست بلکه نسبت زمانها به یکدیگر است
- فقط پروسس‌های وابسته باید Snyc باشند.
- مهم توافق زمانی بین ماشین‌هایست 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 لم پورت
- $a \rightarrow b$ یعنی a قبل از b اتفاق افتاده (در یک پروسس یا ارسال و دریافت بین دو پروسس) ص ۱۲۲
- x, Y را همزمان گویند اگر $x \not\rightarrow y$ و $y \not\rightarrow x$ 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 کاهشی.
- شکل ۲-۳ ص a۱۲۳: پیغام C, D مشکل کلی دارد، بقیه در محاسبهٔ زمان ارسال مشکل دارند.
- شکل ۲-۳ ص b۱۲۳: مشکل را با الگوریتم لمپورت حل کرده. افزودن ساعت اگر $C(a) < C(b)$ به اندازه $C(a)+1$.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- برای زمانهای مساوی، شماره پروسس به راست زمان اضافه شود.
- با شرایط زیر می‌توان تخصیص زمان در سیستم توزیعی دارد:

1. $a \rightarrow b$ در یک پروسس پس $C(a) < C(b)$.
2. A ارسال و b 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)

- امکان فیزیکی
- زمانهای لمپورت با زمان واقعی تطبیق ندارد و در سیستم‌های بلادرنگ اشکال دارد.
- دو مشکل: تطبیق همه ساعت‌ها با ساعت حقیقی، تطبیق همه ساعت‌ها با یکدیگر.
- مفهوم transit of Thesun و روز خورشیدی (نجوم) شکل ۳-۳ ص ۱۲۵.
- ۳۰ میلیون سال پیش سال اندازه ۴۰ روز بوده! البته ۳۶۵ روز طولانی‌تر.
- متوسط ثانیهٔ خورشیدی.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- ساعت اتمی: حاوی ..... ۱۳۳: ۷۷۰/۶۳۱/۱۹۲/۹ ترانزیشن، در ۵۰ آز وجود دارد.
- BIH متوسط می گیرد، ساعت اتمی بین المللی TAI را می‌دهد.
- یک روز اتمی 3μSec کوتاه‌تر از متوسط روز خورشیدی است.
- در سال ۱۵۸۲ پاپ گریگوری سیزدهم دو روز از تقویم حذف کرد. ⇐ دعوا شد.
- شکل ۴-۳ نحوهٔ همگامی TAI و زمان خورشیدی و معرفی UTC بجای GMT.
- تا کنون ۳۰ تا ثانیه کبیر داشته‌ایم.
- چند رادیو و ماهواره‌ها UTC را اعلام می کنند ولی زمان انتقال پیام را هم باید در نظر گرفت.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم همگام‌سازی ساعت
- اگر یک ماشین WWW (گیرندهٔ UTC) داشته باشد بقیه می‌توانند همگامش شوند.
- اگر نه نیاز به الگوریتم داریم.
- در زمان t روی ماشین p زمان $C_p(t)$ است ایده‌آل: $C_p(t)=t \ \forall \ p, t$. یا $\frac{dc}{dt}=1$ با دقت 10^{-5} داریم

p=Maximum Drift Rate بجای ۲۱۵۹۹۸-۲۱۶۰۰۲ ۶۰×۳۶۰۰=۲۱۶۰۰

$1-p \le \frac{dc}{dt} \le 1+p$ شکل ۵-۳ ص ۱۲۸.

- دو ماشین با Drift مخالف پس از Δt دارای $2p\Delta t$ ۳/۲۵ اختلاف.
- برای اختلاف کمتر از δ باشد باید هر $\delta/2p$ با هم هنگام شود

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم کریستین
- سنکرونیزه کردن بقیه ماشین‌ها با یک ماشین که WWW دارد (Time Server).
- هر $\Delta t \leq \delta/zp$ پیغام به Time Server و دریافت زمان هر چه سریعتر(C_{UTC}) شکل ۶-۳ ص ۱۲۹.
- مشکل اصل ساعت فرستنده عقب است و محلی $C > C_{UTC}$ و کار خراب می‌شود.
- بجای اصلاح یکجا می‌توان در هر وقفه (..۱ تا در ثانیه) بجای 10ms یا 9ms یا 11ms اضافه کرد.
- مشکل فرعی زمان ارسال پاسخ است که یک تقریب افزون در $\frac{T_1 - T_0}{2}$ شکل ۶-۳ ص ۱۲۹ به زمان ارسالی.
- اگر مدت سرویس وقفه (I) را بدانیم تقریب بهتر (T_1-T_0-I) وصول گری بقیه، تعیین می‌نیمم بعنوان زمان واقعی.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم برکلی
- برای راحتی که WWW 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 ۱۳۰.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم معدل گیری
- الگوریتم‌های بالا متمرکز دارد.
- تقسیم زمان به فواصل $T_0+iR, T_0+(i+1)R$.
- Broad Cast فرمان در شروع هر فاصله از هر ماشین (همه در یک لحظه نیست).
- جمع‌آوری فرمانها در هر ماشین و معدل گیری از همه یا حذف بزرگترین و کوچکترین‌ها و معدل گیری.
- افزودن زمان انتقال هم خوب است.
- داشتن چند منبع زمانی خارجی (از WWV، GEOS، .....).

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- با توجه به شرایط شبکه باید برای UTC فاصله زمانی تعریف کرد بجای زمان مطلق.
- زمانهای دریافتی باید معدل گیری شود. هیچ برد از منوال پاسخ را فوری نمی گیرد.
- ماشین‌ها فاصلهٔ زمانی خود را Broad Cast 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 ۱۰ بحث می‌شود).

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- بکار گیری ساعت‌های همگام
- پیدایش مکانیزم‌های همگام‌سازی ساعت‌ها روی شبکه در سال‌های اخیر.
- الگوریتم حداکثر یکبار تحویل پیام.
- روش سنتی: شماره گذاری پیام‌ها و ذخیره شماره‌ها برای تشخیص پیام تکراری از پیام اصلی.
- روشن یا قطع برق اطلاعات پاک می‌شوند و چقدر وقت باید آنرا نگهداشت.
- هر پیغام یک مهر زمانی و شماره مسیر دارد و برای هر مسیر دیرترین مهر حفظ می‌شود.
- پیام‌ها با زمان کوچکتر صرف نظر می‌شود. چون تکراری‌اند.

حداکثر تفاوت سرعت – حداکثر پیغام – 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.
- بعداز هر Δt زمان جاری روی دیسکت ثبت می‌شود.
- بعداز هر Crash و سپس Reboot، G از روی دیسکت خوانده شود و Δt 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 .... At Most است.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- تطبیق Cache Cousisteuly برمبنای ساعت
- در سیستم فایل توزیعی عمل Caching برای تسریع استفاده می‌شود.
- مشکل عدم تطبیق Cacheها پیش می‌آید.
- یک راه تمیز دادن Cache برای خواندن و برای نوشتن است.
- مسئله وقتی است که نوشتنی بعد از یک خواندن انجام شود که Cache خواندن را باید Invali dale دارد و با روش همگامی ساعت‌ها برطرف می‌شود.
- روی هر فایل Cache شده یک Lause گذاشته می‌شود (مدتی که Valid است).

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- بعداز 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 بعداز تعدادی خواندن انجام شود Server از هم Clientهای خواننده درخواست Timeout دو درس روی Lease می‌دهد.
- اگر 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 ناحیه بحرانی استفاده می‌شود.
- در تک پروسسوری با یک سمافور یا **Monitor** مشکل حل است.
- حالت چند کامپیوتری یا پروسسوری اینجا بحث می‌شود.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- یک الگوریتم متمرکز
- داشتن یک ماشین هماهنگ کننده برای نواحی بحرانی شکل ۳-۸ ص ۱۳۴.
- روش عادلانه، دارای پیاده‌سازی ساده و حاوی سه پیغام (Release, Grant, Req) است.
- ولی هماهنگ کننده یک نقطه شکست است که مخالف قرار ما است، شلوغی آنهم ..... خواهد شد.
- اگر هماهنگ کننده مثل حالت b شکل جوابی ندهد Clieut 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 می فرستد.
- روش Releable، (با Aclc) باید و روش گروهی 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 برمی گرداند.
2. توی ناحیه است جواب نمی‌دهد و درخواست را Queue می کند.
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 درخواست کننده صبر می کندهمه OK بدهند و وارد می‌شود وقتی هم خارج شد به آنها هم که در صف هستند OK می‌دهد و صف را خالی می کند.
- شکل ۹-۳ ث ۱۳۶ حالت تصادف را نشان می‌دهد.
- در مقایسه با روش متمرکز اینجا (n-1) 2 پیام داریم آنجا ۳ تا ولی در اینجا نقطه شکست نداریم.
- ولی بجای یک نقطه شکست n نقطه شکست داریم (یعنی دو تا بدی دارد نسبت به متمرکز).
- توقف هر ماشین با انتظار تشخیص داده نمی‌شود-یک راه روش Timeout است.
- مشکل دیگر نیاز به ارتباط گروهی است که ممکن است وجود نداشته باشد.
- مشکل بعدی اینکه بجای شلوغی یک پروسس همهٔ پروسس‌ها شلوغ می‌شوند.
- یک راه تضمین یک تعداد حداکثر OK بجای همه است.
- چرا استفاده می کنیم برای انیکه نشان دهیم الگوریتم توزیع شده امکان‌پذیر است.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم **Token Ring**
- شکل ۳-۱۰ ص ۱۳۹ شبکه **BUS** شکل **a** پروسس‌ها ترتیبی ندارند.
- در نرم‌افزار حلقه منطقی درایم 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 رو می کند به بعدی.
- هر پروسس اگر بخواهد وارد ناحیه بحرانی بشود **Token** را نگه می‌دارد تا از ناحیه خارج شود آنگاه رها می کند.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- Starvatur پیش نمی‌آید، حداکثر انتظار n-1 تا مدت ورود و خروج به ناحیه 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 شکبه حذف شود البته توسط همه.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- مقایسه سه الگوریتم
- شکل ۱۱-۳ ص.۱۴ خلاصه 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: همهٔ پروسس‌ها شمارهٔ بقیه را می‌دانند.
- آنچه نمی‌دانند: کدام up‌اند 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 پروسس بفهمد Coord جواب نمی‌دهد، درخواست انتخاب Coord می‌دهد.

1. P پیغام انتخاب به همه پروسس‌ها با شماره بالا می‌دهد.
2. اگر کسی جواب ندهد P برنده می‌شود و Coord می‌شود.
3. اگر یکی با شماره بالا جواب بدهد او کار را بدست می گیرد و کار P تمام می‌شود.

- حال 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.
- او نتیجه را به همه اعلام می کند.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- الگوریتم رینگ
- هر پروسس می‌داند قبلی‌اش کیست.
- پروسسی که فهمید Coord خراب است شماره‌اش را به قبلی‌اش می‌دهد.
- آنها که متوقف‌اند Ship می‌شوند.
- هر پروسس شمارهٔ خودش را به کمیت اضافه می کند.
- نهایتاً پیغام به شروع کننده می‌رسد.
- او پروسس با لاترین نمره را بعنوان Coord اعلام می کند.
- شکل ۱۳-۲ ص ۱۴۳

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

۴- تراکنش‌های اتمی

- آنچه تاکنون در مورد همگام‌سازی دیدیم خیلی سطح پائین بود (سمافورها ، نواحی بحرانی، ....)
- برنامه‌نویس سطح بالا نیاز به تصادیف سطح بالا دارد. یک روش تراکنش‌ها یا اعمال اتمی است.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- مدل تراکنش
- سیستم= مجموعه‌ای از عمل‌های مستقل که هر کدام امکان شکست دارند.
- حافظه RAM، حافظه دیسکت، حافظه پایا شکل ۱۵-۳ ص ۱۴۶ حالت b خرابی از a کپی می‌شود.
- خرابی یک بلوک در اثر گرد و خاک حالت C، بلوک از کپی درست اصلاح می‌شود.
- محیط مناسب برای محیط متحمل خرابی.
- اجزایی اولیه تراکنش BT، ET، 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 ۱۶-۳ ص ۱۴۸.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- خصوصیت‌های تراکنش: 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 Appent ...ا بایت به آن کماکان ..ا بایت است و در یک لحظه CMT ..ll بایت می‌شود. بدون توجه به اینکه چقدر زمان و چقدر کاربرده تا ...ا بایت اضافه شود.
- مثال دیگر کپی کردن که اگر جا نباشد نصفهٔ کپی شده هم پاک می‌شود. در Dos اینطور نبود.
- سازگاری: هر عمل بانکی موازنه حسابها را به هم نمی‌زند مگر در لحظات انجام تراکنش که قابل رویت نیست.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- ایزوله بودن (ترتیب‌پذیر): اعمال همزمان را با ترتیب خاصی بتوان انجام داد که همان نتایج را بدهد.
- مثال شکل ۱۷-۳ ص ۱۴۹ که برنامه ۳ ترتیب پذیرفت.
- مسئولیت ترتیب‌پذیری با خود سیستم است.
- پایا بودن: بعد از **Commit** نتایج همیشگی گردد.

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 با نتایج CMT باید پاک شود (Ando).
- پس اصول بالا نقض می‌شود، این اصول باید برای تراکنش پدر صدق کند.
- هر پدر یک کپی خصوصی بردارد، فرزندن روی آن عمل کنند، فرزندان نتایج هم را می‌بینند.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- پیاده‌سازی
- انجام موارد ACID نیاز به مدیریت بسیار دقیق دارد.
- روش فضای کاری خصوصی: انجام همهٔ اعمال روی کپی خصوصی و جایگزینی در لحظه CMT.
- اشکال: هزینه کپی‌سازی همه Objectهای مورد نیاز.
- بهینه‌سازی:

1. ایجاد اشاره‌گر به WS پدر، برای بالاترین سطح ایجاد اشاره‌گر به نسخه اصلی برای خواندن و ایجاد یک کپی جدید برای عمل نوشتن.
2. کپی کردن ایندکس و رکوردهای 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 دومی کپی اصلی اعمال می‌شود

  شکل ۱۹-۳ ص۲ ۱۵.
- اگر خواستیم بر گردیم مقادیر قدیمی را از Log جایگزین می کنیم Roll Back.
- وقوع شکست قبل از آخرین جمله و بعد از نوشتن Log آن.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- روش تایید دو فاری TWO Phase Commit (CMT اتمی در سیستم توزیع شده)
  - عمل CMT لحظه‌ای است، در سیستم توزیع شده عمل‌ها توسط چند .... انجام می‌گیرد و نیاز به هماهنگی دارد.
  - یک پروسس نقش هاهنگ 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 گر نه RLBK می‌شود در Log هم نوشته می‌شود.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- شکل ۳-۲۰ ص ۱۵۳.
- در مقابل Crash مقاوم است، بعد از نوشتن Log اولیه کار را دنبال می کند.
- بعد از نوشتن نتیجه رأی Crash کند، بعد از up شدن نتیجه را به بقیه می گوید.
- اگر یکی از دیگران جواب ندهد ارسال تکرار می‌شود یا Timeout شود.
- اگر بعد از دادن جواب سقوط کند از همانجا بعد ادامه می‌دهد.
- مکانیزم‌های کنترل همزمانی (الگوریتم‌ها): مواظبت از اینکه پروسس‌ها سر راه یکدیگر قرار نگیرند.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- قفل کردن
- قدیمی‌ترین و پراستفاده ترین مکانیزم با استفاده از مدیر قفل مرکزی یا محلی.
- مدیر قفل با توجه به لیست خود، درخواستهای جدید برای فایل‌های قفل شده را رد می کند.
- قفل کردن و باز کردن توسط سیستم تراکنش انجام می‌شود بدون دخالت User.
- جداسازی قفل خواندن (Share) از قفل نوشتن (exclusive).
- محدودهٔ مورد قفل: فیلد، رکورد، فایل، جدول، DB، صفحه یا بلوک سخت‌افزاری (مسائل مربوط).
- فرض ما: قفل روی فایل (که خوب نیست) Locking Granularity.
- دانه‌بندی ریزتر= توازی بیشتر= محدوده قفل کوچکتر= گرانتر= تعداد قفل بیشتر= Dead Lock بیشتر.
- قفل دو فازی شکل ۲۱-۳ برای جلوگیری از Dead Lock.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- بهنگام‌سازی اصلی در مرحله تقلیل قفل، اگر قفلی سر راه بود صبر می کند تا آزاد شود.
- ثابت شده TPL ترتیب‌پذیری را تضمین می کند.
- در TPL قطعی فاز تقلیل بعد از CMT یا RLBK انجام می‌شود.
- دو مزیت: ۱) همهٔ فایل‌ها بعداز CMT یا RLBK شدن توسط دیگران استفاده می‌شوند.
  - (2اخذ و آزادی قفل‌ها توسط سیستم انجام می‌شود.
- گرفتن قفل در زمان نیاز به منبع و آزادی پس از خاتمه
- تراکنش‌های به هم گیر نمی‌کنند و Cascade Abort پیش نمی‌آید بدلیل استفاده از اطلاعاتی که نباید دیده شود.
- امکان Dead Lock هیچوقت از بین نمی‌روند حتی در. TPL
- مکانیزم‌های پیشگیری که قبلاً گفته شده، گراف use و مکانیزم زمانی) قفل بیش از T ثانیه = Dead Lock).

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- کنترل همزمانی خوشبینانه
- برو جلو هر کاری می‌خواهی بکن بی‌خیال بقیه، اگر مشکلی بود بعداً فکرش را می کنیم (کاربرد سیاسی).
- فرمان CMT چک می‌شود اگر فایلی در طول کار عوض شده RLBK شده و کار تکرار می‌شود.
- برای فضای کاری محلی خصوصی 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 می گیرد (الگوریتم لمپورت برای همگامی استفاده می‌شود).
- هر فایل دو مهر خواندن و نوشتن مربوط به آخرین تراکنش بکار برنده آن دارد.
- در شرایط عادی هر دسترسی بعداز مهر زمانی فایل است.
- شکل ۳-۲۲ ص ۱۵۷.
- در مقایسه با قفل دارای D.L. نیست که این مزیت اصلی است.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

۵- بن‌بست در سیستم‌های توزیعی

- D.L. در سیستم توزیعی به مراتب پیچیده‌تر است بدلیل پراکنده بودن اطلاعات مربوطه بین ماشین‌ها، اهمیت زیاد در DDB.
- وجود دو حالت زیر، در اینجا هر دو را یکی می‌گیریم .
- بن‌بست ارتباطی :پیغام‌ها به هم گیر کرده‌اند و بافر خالی هم نیست.
- برنامه الف‹-- پیغام‹--1برنامه ب ‹--پیغام ‹--2 برنامه پ ‹-- پیغام ‹--3 برنامه الف
- بن‌بست منبعی :رقابت در گرفتن منابع بصورت انحصاری.
- احتمال بروز بن‌بست ارتباطی خیلی خیلی ضعیف است به خصوص در مدل C/S.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- استراتژی‌های مهم (یادآوری OS1).

1. ۱) الگوریتم شترمرغ: فراموش کردن مسئله.
2. ۲) کشف: صبر کنم بن‌بست شود، کشف کنیم، ترمیم کنیم.
3. ۳) جلوگیری از وقوع: اکیداً پیش نیاید.
4. ۴) سعی در اجتناب: تخصیص با دقت لازم در جهت اجتناب از بن‌بست.

- مورد ۴ در سیستم متمرکز هم استفاده نمی‌شود چه رسد به توزیع شده چون خیلی مشکل است ۳.۲ بحث می‌شود.
- بخصوص مورد ۲ از همه آسان‌تر است که بعداز 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 می کند.
- با تراکنش اتمی می‌توان کار قربانی شده را تکرار کرد با آرزوی موفقیت در دور دوم یا دورهای بعدی!

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- کشف بن‌بست متمرکز
- یک پروسس هماهنگ کننده همهٔ گراف‌های بقیه ماشین‌ها را Union می کند.
- با کشف یک سیکل پروسس را Kill می کند.
- وجود سه روش: ۱) هر ماشین باید اطلاعات مورد نیاز را به پروسس هماهنگ کننده ارسال کند.
- ۲) حذف یا اضافه شدن هر arc ارسال شود، دوره‌ای arcهای اضافه یا حذف شده ارسال شود.
- ۳) هماهنگ کننده خودش هر وقت لازم بود درخواست اطلاعات کند.
- هر سه روش اشکال دارد. شکل ۳-۲۳ ص ۱۶۰.
- بوجود آمدن بن بست نادرست بدلیل عدم دریافت بموقع پیام‌ها

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- پیام ماشین صفر برای آزادسازی R بوسیله B بعد از پیام ماشین یک برای نیاز B به T به هماهنگ کننده می‌رسد گر چه زودتر ارسال شده.
- هماهنگ کننده قبل از ایجاد حلقه به همه اعلام کند پیامی دارند بفرستند و بعد آنها را اعمال کند قبل از ایجاد حلقه.
- این روش خوب 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 TPL سریع‌تر می‌شود.
- پروسس بجای چند بار انتظار برای چند منبع یکبار برای همه منتظر می‌ماند.
- شکل ۲۴-۳: اشکال اصلی در منابع بین ماشینی است.
- بردار (گیرنده، ارسال کننده، پروسس بلوکه شده) در اثر احضار الگوریتم (چندی میزرا حسس) ارسال می گردد.
- وقتی دوباره به اولیه ارسال کننده رسید .D.L کشف می‌شود مثال: بردار (0,8,0) آخر کار.

فصل سوم : همگامی در سیستم‌های توزیع شده (ادامه)

- راه اول: امر به خودکشی پروسس اول اشکال: احضار همزمان بیش ا ز یک پروسس و خودکشی اضافی.
- راه بهتر: افزودن شماره پروسسس‌ها به آخر پیام و انتخاب پروسس با شماره بزرگترین برای خودکشی.
- تئوری و عمل هماهنگ نیستند: چگونه پروسس بلوکه 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 ساختاری بن‌بست امکان‌پذیر نباشد.
- روشهائی 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 نزولی است که حلقه ایجاد نمی‌کند.

1. تکلیف: با جستجو در انتشارات در مورد قفل‌های مختلف موجود پروتکل‌های مربوطه برای قفل کردن و نحوهٔ پیاده‌سازی احتمالی استراتژی TPL یا روش مشابه روی آنها حداقل دو صفحه مطلب بنویسید. هر نفر برای یک محیط DB