چه تغییراتی را در هسته Linux مشاهده خواهیم کرد

کنفرانس هسته Linux

حمید یوسف زاده
yousefz@ccwmagazine.com

روزهای ۲۸ و ۲۹ مارس ۲۰۰۱ شهر سن خوزه در ایالات متحده میزبان برنامه نویسان هسته لینوکس بود که در قالب نشست هسته لینوکس گردهم آمده بودند تا در یک ملاقات رودرو با یکدیگر، راهکارهای آتی را در مورد هسته لینوکس و سمت وسوی آن ترسیم کنند. در دو قسمت قبلی این گزارش، مروری داشتیم بر هشت سخنرانی از ۱۳ سخنرانی این نشست. در بخش پایانی، به مرور پنج سخنرانی دیگر این کنفرانس می پردازیم.

سخنرانی شماره ۸: مدیریت حافظه در هسته لینوکس

سخنران، Rik van Riel بود که در مورد آینده مدیریت حافظه مجازی در لینوکس صحبت کرد. در این سخنرانی، برخی جنبه های مدیریت حافظه که امکان ایجاد بهبود در آنها وجود دارد، مورد بررسی قرار گرفت. در واقع در پیاده سازی فعلی مدیریت حافظه، مکانیزم انتخاب صفحه ای از حافظه که باید جایگزین شود، به این صورت است که جدول صفحات حافظه پویش می شود و صفحاتی که امکان جایگزین شدن دارند، از حافظه اصلی خارج می شوند. این روش به خوبی کار می کند ولی این روش بهترین روش نیست و نیاز به بهبود دارد.

در این بحث، روش نگاشت معکوس (reverse mapping) مطرح شد. اکنون فهمیدن این که چه پردازه هایی دارای یک نگاشت مجازی برای یک صفحه فیزیکی هستند مشکل است. اما اگر یک نگاشت معکوس وجود داشته باشد، انتخاب صفحه ها برای جایگزینی می تواند با پویش حافظه فیزیکی انجام گیرد که باعث بهبودهایی می شود. اما این کار، هزینه سنگینی دارد که آن دوبرابر شدن اندازه جدول صفحات است.

برای حل شدن این مشکل نیز بزرگ کردن اندازه صفحات پیشنهاد شد که آن نیز باعث هدررفتن حافظه در بسیاری از کاربردها می شود. در ادامه، استفاده ترکیبی از صفحات کوچک و بزرگ مورد بحث حضار قرار گرفت. سپس به اشتراک گذاشتن جداول صفحات پیشنهاد شد. در پیاده سازی فعلی، اگر دو پردازه نگاشتی به یک ناحیه از حافظه مجازی ایجاد کنند، دو جدول صفحه جداگانه برای آنها ایجاد می شود که این باعث هدر رفتن حافظه به میزان زیاد می شود. با استفاده از روش اشتراک، این مشکل حل می شود.

در ادامه، یک صحبت کلی در مورد مکانیزم های Readahead انجام شد. سپس نیاز لینوکس به افزایش کارایی Thrashing مطرح شد. سخنران، مکانیزمی را پیشنهاد کرد که با استفاده از آن هنگامی که سیستم آغاز به thrash می کند، پردازه ها را متوقف می کند. درپایان، لزوم بهبود کد فعلی برای سهولت نگهداری آن بیان شد و این که باید مستندات بیشتری در این کد قرار داده شود.

سخنرانی شماره ۱۰: ورودی/خروجی ناهمگام

این سخنرانی توسط BenLaHaise ایراد شد که در مورد پیاده سازی خود از ورودی/خروجی ناهمگام (asynchronous) صحبت کرد. ورودی/خروجی ناهمگام خوب در هسته لینوکس برای مدت ها یک آرزو بوده است که به نظر می رسد این آرزو بالاخره در نگارش ۲/۵ از هسته به واقعیت خواهد رسید.

سخنران بیان کرد که تصمیم گرفته است روش Poslx را مستقیماً پیاده سازی نکند، بلکه روشی انعطاف پذیرتر و عملی تر انتخاب کند. در پیاده سازی وی برخی فراخوانی های سیستمی (system call) جدید فراهم می شود:

-فراخوانی سیستمی ()submit_ios-- که به یک

پردازنده اجازه می دهد یک عمل ناهمگام را تقاضا کند.

-فراخوانی سیستمی io_cancel() -- برای لغو اعمال تقاضا شده قبلی

-فراخوانی سیستمی io_getevents() -- برای اخذ اطلاعاتی در مورد اعمال کامل شده

-فراخوانی سیستمی io_wait() -که به یک پردازنده اجازه می دهد منتظر تکمیل یک عمل خاص بماند.

این پیاده سازی همچنین یک device جدید به نام dev/aio/ ایجاد می کند که برای نگاشت بخشی از حافظه به کار می رود که برای اطلاعات تکمیل عملیات ورودی/خروجی ناهمگام استفاده می شود.

یکی از مشکلات موجود، این است که چگونه تکمیل عملیات در هسته را باید به پردازه ها اطلاع داد. معمولاً این کار توسط صف های انتظار انجام می شود. اما در حالت ناهمگام، پردازه در این صف قرار نمی گیرد. برای حل این مشکل سخنران صف انتظار را گسترش داده است تا به نحوی به پردازه ها اطلاع داده شود.

البته کارهای دیگری نیز باید برای ورودی /خروجی ناهمگام انجام شود. از جمله مستند کردن کد، درنظر گرفتن محدودیت های منابع، گسترش انواع devece که ورودی/خروجی ناهمگام را پشتیبانی می کنند، و بسیاری کارهای دیگر.

سخنرانی شماره ۱۱: مدیریت نیرو

در این سخنرانی Andy Grover از اینتل کار خود زمینه مدیریت نیرو (Power management) در لینوکس را ارائه کرد. کد قبلی که مبتنی بر APM بود، منقرض شده است و اکنون ACPI به عنوان روش استاندارد مطرح است.

پیاده سازی اینتل از ACPI توسط FreeBSD استفاده شده است و اینتل منتظر هسته ۲/۵ است تا آن را برای لینوکس ارائه کند. برای ایجاد درک بهتر در مورد میزان پیچیدگی ACPI کافی است بدانیم پیاده سازی آن ۵ الی ۷ نفر سال کار برده است و هنوز هم پشتیبانی انتقال سیستم به حال خواب (Sleep) را ندارد.

خواب بردن سیستم یک کار پیچیده است. کد ACPI باید بتواند تمامی devece های سیستم را پیدا کند، به آنها بگوید که حالت خود را ذخیره کنند، و سپس آن را خاموش کند. این کار باید در ترتیب درست انجام شود، مثلاً خاموش کردن کنترلر دیسک در حالی که دیسکی در آن در حال گردش است، اشتباه غیرقابل قبولی است.

سخنران برای انجام این کار روشی را پیشنهاد کرد که البته بیشتری آن هم انجام شده است و برای هسته ۲/۵ آماده خواهد بود.

سخنرانی شماره ۲۱: Bitkeeper

سخنرانی توسط LaryMcvoy از LMBench ایراد شد. Bitkeeper یک مکانیزم کنترل سورس است که به نظر خیلی خوب می رسد. این محصول برخی قابلیت ها را دارد که آن را برنزاز سیستم CVS می کند. اما بیشتر مزایای آن را می توان در واسط کاربر آن دانست. کار با این سیستم لذت بخش و آسان است و ابزارهای مختلفی دراختیار کاربران قرار داده می شود. مثلاً یک ابزار مناسب برای مشاهده سابقه (history) سیستم وجود دارد که یافتن آخرین تغییردهنده یک کد را بسیار آسان می کند. Bitkeeper را می توان یک پیشرفت قابل ملاحظه در عرصه تولید تیمی نرم افزار دانست.

سخنران سعی داشت ایجاد کنندگان هسته را متقاعد کند که برای مدتی از Bitkeeper استفاده کنند. در زمان برگزاری کنفرانس هیچ موافقتی با این موضوع صورت نگرفت ولی استفاده از آن در تولید هسته ۲/۵ دور از انتظار نیست.

سخنرانی شماره ۱۲: مباحثه

آخرین بخش از کنفرانس، یک میزگرد بود. این بحث توسط Ted Tso رهبری می شد. او پیشنهاد کرد که برنامه نویسان هسته (و مخصوصاً لینوس توروالدز) یک تاریخ قطعی برای اتمام کارهایی که روی هسته ۲/۵ انجام می شود، تعیین کنند. به این ترتیب هر کسی می داند چه زمان باید که خود را آماده کند.

همچنین استفاده از یک روش رسمی و برنامه ریزی شده برای پروسه کنترل باگ ها مورد تأیید قرار گرفت. بحثی در مورد Bugzilla و همچنین GNATS انجام شد که هیچ کدام مورد تأیید حضار قرار نگرفتند.