Crema, 29 Novembre 2003
LINUX DAY 2003

LINUX KERNEL 2.6
Le nuove feature

di Gian Paolo Ghilardi

Dove eravamo rimasti?

- 22 Novembre 2001: con il rilascio del kernel 2.4.15 il creatore di Linux, Linus Torvalds, ha deciso di aprire il nuovo branch di sviluppo del kernel (2.5.x), che condurrà alla futura serie 2.6
- lo spirito con cui nasce la nuova serie di sviluppo è quello di ripulire il codice e sostituire quello che non va senza però dimenticare il supporto alla "latest cool feature of the day", tanto cara all'autore di Linux
- lo scopo finale rimane immutato: la "world domination last", teorizzata dallo stesso Torvalds, compie un nuovo passo in avanti. ☺

Linux 2.6

- In due anni di sviluppo, il Linux 2.6 ha subito radicali cambiamenti e miglioramenti sotto ogni punto di vista e per ogni tipologia di utenza (guru, programmatore esperto, utente neofita,...)
- La quantità di nuove feature e cambiamenti è veramente impressionante: elencarle tutte richiede parecchi minuti.
- Il kernel 2.6 è così "avanti" che l'ormai "vecchio" 2.4 vive di backport della serie nuova: solo quello che funziona sul 2.6 può essere aggiunto nel 2.4!
- Scopo di questa presentazione è mostrare le feature più innovative della nuova serie e spiegarne l'impatto reale nella vita di tutti i giorni, in termini chiari ed immediati

Nuove feature: riepilogo 1/2

Le feature del 2.6 sono davvero troppe!
Presenteremo le seguenti:

- fully-preemptive Kernel
- Process Scheduler O(1)
- Anticipatory I/O Scheduler
- supporto ai thread migliorato (NGPT e NPTL)
- pseudo-filesystem Sysfs

Nuove feature: riepilogo 2/2

- nuovo filesystem Reiser4
- innalzamento della frequenza di clock del kernel
- modifica on-the-fly di clock e di voltaggio della CPU
- supporto all'ACPI migliorato
- software suspend su disco ed in RAM
- driver ALSA e lm_sensors integrati

Fully-preemptive Kernel

- un kernel è fully-preemptive se è in grado, in qualunque momento e con un ritardo minimo, di sospendere l'esecuzione di un processo per eseguirne un altro (per esempio un processo che ha una qualche valenza critica per l'intero sistema come una richiesta IRQ)
- in user-space il "vecchio" kernel 2.4 era già preemptive, mentre in kernel-space questa capacità era limitata: spesso si doveva "emularla" ricorrendo a "trucchi sporchi" come sleep, yield, busy waiting anche grossolani (Big Kernel Lock),...
- questi stratagemmi penalizzavano le prestazioni perchè venivano inutilmente persi molti cicli di CPU
- Robert Love si è interessato al problema: ha creato infrastruttura e API per gestire efficacemente la full kernel preemption.
  Con Ingo Molnar, ha inoltre eliminato parecchi Big Kernel Lock, ostacoli alla full premption
- la nuova feature in poche parole: i tempi di risposta ad un evento IRQ sono bassissimi, i processi sono gestiti meglio e le prestazioni aumentano in maniera chiaramente avvertibile

O(1) Process Scheduler

- lo scheduler dei processi è il sottosistema che ne decide l'ordine di esecuzione: decide chi deve essere eseguito e quando
- nel 2.4 lo scheduler verifica le priorità di tutti i processi in coda prima di decidere chi deve essere eseguito. Quindi impiega un tempo lineare: O(N)
- il 2.6 ha un nuovo scheduler, in grado di decidere che processo eseguire con un tempo costante: O(1)
  - ora vi sono due code per ciascuna CPU: una per i processi in attesa del quanto di CPU ed una per quelli che l'hanno già ricevuto
  - la scelta del processo da eseguire in base alla priorità di fatto non sussiste perchè il kernel sa già quale sarà il prossimo processo da eseguire: è il primo della coda dei processi in attesa
    Ecco perchè il tempo di esecuzione è O(1)!
  - i processi "real-time" condividono una sola coda, così da essere serviti velocemente
- la nuova feature in poche parole: le prestazioni dell'intero sistema migliorano notevolmente perchè vengono ridotti i costi di servizio ("overhead") dovuti alla gestione stessa dei processi
- nelle prime implementazioni vi erano problemi di starvation e excessive cpu-affinity: ora sono state risolte senza penalizzare l'eccellente lavoro svolto

Anticipatory I/O Scheduler 1/2

- ogni sistema operativo a fianco del più noto scheduler di processi ne ha un'altro, dedicato alla gestione dell'I/O. Scopo di questo sottosistema è quello di gestire le richieste di scrittura e di lettura da e verso dispositivi di memorizzazione di massa (es. hard disk)
- problema: vi è mai capitato di notare dei vistosi rallentamenti dell'intero sistema mentre state trasferendo dei dati da un lettore cd sul vostro harddisk IDE? Avete mai osservato un sistema apparentemente bloccato fino al termine di un lungo trasferimento di file fra due dischi o via rete?
- kernel 2.4:
  - se contemporaneamente ad una prolungata scrittura di dati da disco si verifica una serie di richieste di lettura, lo scheduler "standard" di Linux offre prestazioni misere perchè è basato su un modello a "deadline" piuttosto rigoroso
  - ogni nuova richiesta (non importa se sia di lettura o di scrittura) è semplicemente posta in coda per essere processata sequenzialmente!
  - se ci sono troppe richieste di scrittura in sequenza, le richieste di lettura soccombono ed il risultato è che l'apertura e la responsività di programmi che accedono in lettura a dei dati è seriamente rallentata o addirittura "bloccata".
  - ecco perchè può capitare che XMMS smetta di funzionare correttamente (ed inizi a emettere suoni a sprazzi) mentre copiate il contenuto di un cd sul vostro harddisk!

Anticipatory I/O Scheduler 2/2

- il problema è semplice: come ottimizzare la coda di letture e scritture in maniera che le letture non siano penalizzate? L'utente avverte questo tipo di penalizzazione come lentezza del sistema! (es. apertura di file vistosamente rallentata)
- kernel 2.6
  - Nick Piggin, uno studente universitario australiano, ha lavorato su un nuovo tipo di scheduler I/O per Linux
  - il nuovo scheduler "anticipatorio" risolve in maniera intelligente il problema citato poc'anzi: dopo ogni lettura attende pochi millisecondi, cercando di "anticipare" una eventuale nuova richiesta di lettura (che ovviamente avrà priorità su eventuali richieste di scrittura). Così tale richiesta di lettura non è penalizzata ed il sistema appare più reattivo!
  - l'Anticipatory è divenuto lo scheduler I/O di default nel kernel 2.6 (ce ne sono altri, meno "famosi" ma altrettanto validi benchè sfruttino approcci al problema radicalmente differenti)
  - all'occorrenza si può forzare perchè emuli il "vecchio" deadline scheduler
  - la nuova feature poche parole: se state copiando il contenuto di un cd sul vostro harddisk IDE, sarete liberi di aprire e leggere contemporaneamente dei file senza notare vistosi rallentamenti dovuti alla scrittura su disco

Supporto ai thread migliorato 1/2

- la gestione dei thread usata fino alla serie 2.4 (i famosi LinuxThreads, definiti “penosi” dallo stesso Linus) è stata sostituita da due nuove implementazioni

- Next Generation Posix Threads (NGPT)
  - progetto sviluppato da IBM
  - creati per sostituire i LinuxThreads in maniera rapida ed indolore
  - supporto allo standard POSIX migliorato
  - aumento di prestazioni su sistemi SMP
  - notevole miglioramento in user-space
  - richiede alcune modifiche in kernel-space per essere sfruttato al meglio

Supporto ai thread migliorato 2/2

- Native Posix Thread Library (NPTL)
  - sviluppato da Ingo Molnar e Ulrich Drepper (RedHat)
  - sfrutta un modello 1:1, totalmente in kernel space (mentre la NGTP usa un modello M:N dove M sono i thread in user-space ed N quelli in kernel-space)
  - integrato (e quindi già testato) già nelle distribuzioni di RedHat a partire dalla 8.0
- ovviamente sono stati fatti vari benchmark fra le due implementazioni e pare che l'NPTL sia il vincitore (benché c'è chi sostenga che lo scontro sia impari dal momento che si confrontano due modelli di thread diversi e nati per scopi diversi)
- la nuova feature in poche parole:
  - basta ricompilare le applicazioni multithread con il supporto per una di queste due nuove implementazioni per ottenere incrementi prestazionali notevoli
  - il supporto migliorato allo standard POSIX aiuta i programmatori che scrivono applicazioni multipiattaforma (ossia rende i programmi più facilmente "portabili" verso altri SO)
  - finalmente si fa un po' di chiarezza sia a livello di API, sia a livello di documentazione!

Sysfs Pseudo-filesystem

- il 2.6 include un nuovo pseudo filesystem che rappresenta il "nuovo posto giusto" per le informazioni sui dispositivi
- il nuovo filesystem contiene 6 directory (block, bus, class, devices, firmware, net) ognuna delle quali rappresenta un diverso raggruppamento logico di periferiche (es: per bus di connessione, per classe di appartenenza,...)
- la nuova feature in poche parole:
  finalmente si fa un po' di ordine!
  - /proc conterrà solo le informazioni sui processi
  - /sys conterrà le informazioni sui dispositivi

Reiser4 Filesystem

- è in corso la fase di test di un nuovo filesystem per il kernel 2.6: Reiserfs v4.0 (“reiser4”), progettato da Hans Reiser di Namesys
- è un filesystem di nuova concezione ed è stato scritto ex-novo
- feature supportate:
  - journal playback: se la vostra macchina va in tilt, al riavvio non dovete attendere i soliti lunghi controlli di coerenza sul filesystem!
  - interfacciamento basato su plugin: “file” e “directory” non sono altro che due dei possibili modi con cui gestire i dati: tramite i plugin si possono creare nuove attributi di sicurezza, magari creare motori SQL per ricercare file su HDD,... Esistono ben 7 tipologie distinte di plugin: file, directory, hash, security, item, key assignment. Il filesystem delega totalmente ai plugin il “come gestire” i dati!
  - atomicità delle operazioni: ogni operazione sul filesystem deve essere eseguita “in blocco”, senza interruzioni di sorta: quindi per Reiser4 un’operazione o è stata “completata nella sua interezza” o semplicemente “non è mai avvenuta”!
  - progettato per un alto livello di sicurezza militare: la DARPA sponsorizza questo progetto perchè possa essere certificato per usi militari, in ambienti protetti
  - è uno dei progetti meglio documentati per Linux
- su precisa domanda posta da un giornalista, Hans Reiser ha definito questo progetto come "l’unica alternativa concreta e competitiva al futuro WinFS di Microsoft“
- le prestazioni sono già eccellenti (per non dire sorprendenti, a detta di chi lo ha testato) benché gli sviluppatori abbiamo dato la priorità alla stabilità (altrimenti Linus ne ritarderà l’inserimento nella serie stabile ufficiale ☺)

Responsività e Tuning

- gli sviluppatori hanno dato molta importanza al concetto di "responsività" ossia la capacità di un sistema di rispondere più velocemente possibile ad eventi sincroni ed asincroni
- gli utenti desktop percepiscono un sistema più reattivo anche in situazione di carico notevole usando il 2.6
- per aumentare la responsività sono state apportate dei cambiamenti ed introdotte nuove feature:
  - la frequenza interna del clock ora è impostata a 1000 Hz (i context switch sono maggiori e quindi anche i processi utente hanno maggior probabilità di essere eseguiti)
  - un miglior controllo delle fasi di lock ha permesso l'eliminazione di parecchi Big Kernel Lock che, bloccando temporaneamente l'intero sistema a seguito di richieste particolari, rendevano quest'ultimo meno responsivo
  - Con Kolivas, un altro kernel hacker, ha sviluppato diverse patch per migliorare il modo con cui lo Scheduler O(1) "tratta" i programmi multimediali (es. Xmms)

Modifica on-the-fly di clock e di voltaggio della CPU

- si può cambiare il clock ed il voltaggio della CPU in tempo reale sui sistemi che supportano questa feature
- notevole risparmio di batteria sui laptop!
- molte tecnologie supportate: Intel SpeedStep, Transmeta Crusoe, Intel Xeon, AMD PowerNow K6, ARM, AMD Elan, VIA Cyrix Longhaul
- per agire sul clock della cpu basta scrivere nei file presenti nelle directory /sys/devices/system/cpu/cpu[ID]/cpufreq/ (si noti il razionale utilizzo del filesystem Sysfs per questo scopo)
- la nuova feature in poche parole:
  - a differenza del "vecchio" 2.4 ora vi sono molti più file di configurazione, ognuno dei quali rappresenta uno dei differenti parametri di funzionamento della CPU: la flessibilità di utilizzo è aumentanta!
  - ora vi è una sola interfaccia comune a più tecnologie e, soprattutto, è ben definita: presto saranno disponibili piccoli programmi grafici per gestire questa funzionalità in maniera semplice e rapida (ossia si assumeranno il compito di gestire in maniera trasparente tutti i file di configurazione al posto vostro)

Integrazione di patch “conosciute”

- Nel 2.6 sono state incluse due patch lungamente attese
  - driver audio ALSA (Advanced Linux Sound Architecture): garantiscono una migliore qualità sonora ad un numero più ampio di schede rispetto ai vecchi driver OSS (Open Sound System)
  - driver lm_sensors: permettono di monitorare temperature, velocità delle ventole e voltaggi sull’hardware dotato degli appositi sensori (praticamente tutte le schede madri recenti, alcune schede video,...)

Software suspend su disco e su RAM

- è possibile salvare lo stato completo del sistema:
  - su disco: in fase di reboot è possibile effettuare il restore
  - su RAM: la macchina rimane accesa ed entra in stand-by software
- in entrambi i casi non è richiesto il supporto all'Advanced Power Managament (APM) abilitato visto che si tratta di una feature squisitamente software
- la nuova feature in poche parole:
  - salvando su disco: se attivate la funzionalità, poi spegnete e riaccendete il PC, potrete continuare a lavorare esattamente da dove eravate prima dello spegnimento
  - salvando in RAM: potrete mettere "in pausa" la vostra macchina senza doverla spegnere (la batteria del vostro laptop ringrazierà)

That’s all, folks!

Grazie per l’attenzione... ☺