

27 Novembre 2004
LINUX DAY 2004

Linux Kernel 2.6: stato di avanzamento e futuro

GIAN PAOLO GHILARDI

Una breve cronistoria

- il 17 Dicembre 2003 viene rilasciata ufficialmente la prima release della nuova serie del Kernel Linux, la 2.6.0.
- vediamo ora, a quasi un anno da quel momento, come sia cambiato il kernel e tentiamo di estrapolare (non essendovi nulla di definito) indicazioni sul suo sviluppo.

Il nuovo (?) modello di sviluppo

- dopo l'uscita della release 2.6.0 ufficiale, l'autore e il project leader di Linux, Linus Torvalds, comunica al mondo che Andrew Morton sarà il maintainer della nuova serie, mentre ritaglierà per sé stesso il ruolo di maintainer/curatore di un eventuale nuovo ramo (branch) di sviluppo che condurrà alla prossima serie di kernel.
- in questo momento, in attesa della release 10, il passaggio di incarico a Andrew Morton è, di fatto, fittizio:
 - Torvalds continua a “firmare” i rilasci ufficiali del kernel 2.6 mentre Morton ha assunto il compito di raccogliere, integrare e testare le numerose patch provenienti dagli sviluppatori.
 - la serie “-mm”, curata da Morton, rappresenta il primo passo prima dell'inclusione delle patch nel “tree” ufficiale del kernel (“vanilla”), gestito da Torvalds.

Pro del “nuovo” modello di sviluppo

- un significativo aumento della velocità di integrazione/test delle patch: in precedenza, essendovi solo Linus a curare il “merging” delle patch, poteva accadere che una di esse passasse “inosservata” a lungo.
- un secondo “tree” a fianco di quello ufficiale: ciò rappresenta una sorta di “controllo di qualità” sulle patch prima della loro inclusione nel kernel vanilla.

Contro del “nuovo” modello di sviluppo

- l'aumento del numero di patch è stato letteralmente vertiginoso.
- a titolo comparativo: nella serie 2.4.X, fra una pre-release e l'altra si avevano al massimo un centinaio di patch. Ora invece si parla di migliaia!!!
- garantire la stabilità è diventato molto più difficile.
- i detrattori di questo modello di sviluppo parlano della serie “-mm” come di un “tree discarica” dove tutti possono buttare dentro qualcosa, sperando che tutto funzioni correttamente (ed il risultato è che il tree non sempre compila). E' facilmente intuibile che serie ufficiale e serie -mm possano differire ampiamente e ciò comporta un ulteriore ostacolo ad un processo di stabilizzazione.

“Naming wars” 1/2

- l'aumento della velocità di integrazione delle patch ha portato ad un cambio nello schema di “naming” delle versioni del kernel.
- nella serie 2.4.X, prima di ogni rilascio, un kernel passava attraverso una fase di integrazione delle patch/feature (identificata dal prefisso “-pre”) ed una di correzione e stabilizzazione (“-rc”: release candidate).
- con l'avvento del kernel 2.6 e del tree “-mm”, Linus ha abbandonato la prassi di marciare con “-pre” le pre-release ed è passato a chiamarle direttamente “-rc”.

“Naming wars” 2/2

- questo nuovo modo di procedere ha provocato la reazione di numerosi sviluppatori.
 - “Concordo, le sigle -pre e -rc danno una forte indicazione che, per le -pre, si possono prendere in considerazione nuove feature mentre per le -rc è meglio fare test seri.” (autore: Bill Davidsen)
- tuttavia Linus, almeno per il momento, ha deciso di continuare così. :)
 - “Il fatto è che la convenzione usata per dare nomi al kernel ha sempre fatto pena. Quanto a me, io sono un tipi “artistico” quindi spesso tento di fare qualcosa di nuovo e invariabilmente stupido.”

Comunque, per qualche motivo (usare) quattro numeri (per identificare la release) mi pare visualmente antipatico; quindi, siccome la cosa non mi interessa molto, continuerò ad usare la sigla “-rc” e possiamo concordare che questa sigla sta per “Ridiculous Count” piuttosto che per “Release Candidate”.

Cosa molto più importante, forse potremmo tutti realizzare che questo non è certamente una problema così grande! ;)”

To fork or not to fork?

- molti sviluppatori si stanno chiedendo se, dato il vistoso aumento di bugfix/feature/..., abbia ancora senso definire la serie 2.6 come “stable” piuttosto di “development” (come lo è stata la serie 2.5.X).
- di conseguenza, diversi kernel hackers, hanno cominciato a richiedere a gran voce l’apertura di un nuovo branch di sviluppo, una nuova serie 2.7.X e porre la serie 2.6 sotto uno stretto processo di stabilizzazione.
- la risposta è nuovamente nelle mani di Linus (anche se al momento non pare orientato in tal senso).

La serie 2.6: il codice

- descritta l'attuale situazione “politica” dello sviluppo del kernel, passiamo ora all'aspetto più importante, ossia al prodotto dello sviluppo, il codice.
- lo sviluppo del codice è proseguito nell'ottica di un progressivo miglioramento di tutte le sue componenti.

Scheduler

- il process scheduler “ufficiale”, l’O(1) di Ingo Molnar, è stato oggetto di continui miglioramenti nel tentativo di supportare tutti i possibili carichi di lavoro (workload).
- allo scheduler si sono aggiunti nuovi scheduler ottimizzati, come quello desktop-oriented di Con Kolivas (sviluppatore che, tra l’altro, ha proposto di rendere gli scheduler intercambiabili attraverso un meccanismo “a plugin”).
- anche lo sviluppo degli I/O scheduler (anticipatory, CFQ, deadline...) introdotti col kernel 2.6 è stato portato avanti nel tentativo di ottimizzare le code di lettura e scrittura sui vari dispositivi a seconda dei diversi carichi di lavoro.
- a seguito di specifiche esigenze, sono apparse varianti ottimizzate (per esempio per il mercato “embedded”).

Supporto Real-Time 1/2

- recentemente sono state prodotte due diverse implementazioni del supporto a processi real-time (RT).
- la prima implementazione è di Ingo Molnar, l'autore dello scheduler O(1)
- nell'ottica di una riduzione progressiva delle latenze nel kernel ha introdotto diverse patch come quella che implementa la "kernel preemption volontaria". Una delle sue ultime patch ha introdotto una nuova modalità (PREEMPT_REALTIME) che rappresenta "una nuova implementazione di un modello di kernel totalmente preemptive nella quale gli spinlock e gli rlock usano semafori e supportano la modalità preemptive per default"
- a fianco della "fully preemption", tramite cui il kernel può in qualunque momento interrompere un processo per assegnare la CPU ad un altro, si è aggiunto il supporto (sperimentale!) a quei processi che richiedono una schedulazione immediata in caso di particolari eventi e, conseguentemente, latenze di attesa minime per ottenere la CPU.
- per ottenere questo risultato, Ingo ha lavorato sulle "primitive" del sistema (es: gli spinlock), rendendo la gestione dei processi e delle relative problematiche più deterministiche, facendo quindi un passo in direzione del supporto al "vero" Real Time.

Supporto Real-Time 2/2

- la seconda implementazione è opera invece di una azienda specializzata nel mercato embedded, MontaVista.
- l'implementazione proposta, definita “prototipale”, è costituita da una collezione di patch come quella per la preemption volontaria di Molnar, quella per la sostituzione dei Big Kernel Lock e degli spinlock attraverso mutex.
- a detta degli sviluppatori, la patch “è uno sforzo per ridurre ulteriormente la latenza di interrupt e per ridurre drammaticamente la latenza della preemption nei kernel della serie 2.6”.
- l'idea-chiave è quella di rendere il kernel 2.6 adatto per “applicazioni multimediali ad alte prestazioni e per quei programmi che richiedono rapidissime ed affidabili funzioni di controllo a livello di processo” (es. il dispositivo ABS delle automobili).

Software suspend

- recentemente la frattura relativa allo sviluppo della feature nota come software suspend si è ricomposta.
- questa feature permette di interrompere l'esecuzione di un sistema e salvarne lo stato su disco per poter riprenderne l'utilizzo successivamente.
- a seguito di un diverbio fra due sviluppatori vi erano ormai altrettanti diversi rami di sviluppo (identificati con i nomi pmdisk e swsusp).
- ricomposta questa frattura, ora gli sviluppatori hanno ripreso a lavorare insieme e sono in cantiere nuovi miglioramenti.

Compatibilità binaria 1/2

- uno dei problemi che affliggono il mondo di Linux ed OpenSource è rappresentato dalle incompatibilità binarie che si manifestano fra le distinte distribuzioni e talvolta anche fra una versione e l'altra di una stessa distribuzione .
 - esempio: moduli/driver che funzionano con una release di una distribuzione ma non funzionano su quella successiva
- tali incompatibilità talvolta sfociano in vere e proprie differenze a livello di API, posizione e formato dei file di configurazione e addirittura a livello di nome dei programmi di sistema.
- tutto ciò appare preoccupante e comporta una assurda frammentazione degli utenti Linux e OpenSource.

Compatibilità binaria 2/2

- per ovviare a questi problemi, si può contare sulla famosa organizzazione Linux Standard Base, votata alla creazione di specifiche di standard per “incrementare la compatibilità fra le distribuzioni e permettere ai programmi di essere eseguiti su ciascun sistema compatibile”)
- recentemente si è aggiunto un nuovo apposito Gruppo di Interesse in seno all’OpenSource Development Lab (OSDL), denominato “Binary Testing SIG” il cui scopo è risolvere appunto questi problemi garantendo a Linux la cosiddetta “forward binary compatibility” richiesta a gran voce da tutti gli utenti ed anche e soprattutto dalle aziende che fanno di Linux un business.
- speriamo bene! ;)

Stato di Reiser4 1/3

- il nuovo filesystem journaling prodotto da Hans Reiser non ha ancora trovato spazio nel kernel 2.6 ufficiale.
- è stato incluso nel tree “-mm” ed è tuttora considerato sperimentale (lo stesso autore ha più volte ammesso di essere al lavoro tanto in fase di ottimizzazione delle performance, quanto in quella di aggiunta delle feature previste/richieste)
- l'integrazione piena nel kernel ufficiale appare difficoltosa in virtù dei profondi cambiamenti richiesti da questo filesystem a livello di Virtual Filesystem Switch, lo strato software del kernel che fornisce interfacce standard ed unificate che permettono la coesistenza pacifica dei differenti filesystem in un sistema (es: una chiamata “open”, ad esempio, deve poter funzionare indipendentemente dal filesystem usato!)
 - Reiser4 è concepito in maniera molto differente dagli altri filesystem e adattarlo ad un'interfaccia come quella “imposta” dal VFS appare difficile (al punto che Hans Reiser ha provocatoriamente chiesto di usare il codice di Reiser4 come rimpiazzo del VFS, ritenendolo superiore a quest'ultimo! ;))

Stato di Reiser4 2/3

- perché Reiser4 è così “particolare”? i file si comportano in una maniera strana: a ciascuno di essi sono associati dei metadati (informazioni extra) ma non solo...
- esempio (*): supponiamo di avere un file di testo chiamato CREDITS
 - si possono visualizzare i metadati (definibili anche tramite plugin. es: Access Control List) attraverso il programma `tree` e...
 - ... si può leggerne il contenuto con `cat CREDITS` ad esempio...
 - ... ma è possibile ottenere analogo risultato con `cd CREDITS; cat .`

```
$ tree CREDITS/metastats
CREDITS/metastats
|-- bmap
|-- gid
|-- items
|-- key
|-- locality
|-- new
|-- nlink
|-- oid
|-- plugin
    |-- compression
    |-- crypto
    |-- digest
    |-- dir
    |-- dir_item
    |-- fibration
    |-- file
    |-- formatting
    |-- hash
    |-- perm
    `-- sd
|-- pseudo
|-- readdir
|-- rwx
|-- size
`-- uid
```

(*) tratto da lwn.net

Stato di Reiser4 3/3

- in Reiser4 il confine fra file e directory è labile e come è facilmente intuibile questa feature, per quanto interessante, comporta numerosi cambiamenti (e quindi critiche).
- Reiser4 rappresenta un modo nuovo di vedere e gestire file e directory e contiene altre funzionalità interessanti (la gestione plugin-based, l'atomicità di tutte le operazioni, ...).
- attendiamo fiduciosi la stabilizzazione e le ottimizzazioni di performance per poter iniziare ad usarlo in modo intensivo. ;)

Conclusioni

- in attesa dell'apertura di una nuova eventuale serie sperimentale (2.7.X?), lo sviluppo del Kernel serie 2.6.X continua senza sosta.
- in questa breve presentazione si è cercato di mostrare lo stato presente e qualche indicazione sul futuro di Linux.
- spero vi sia piaciuta e...
- ...that's all, folks! ;)