Capitolo 1

Sotto la cresta dell'onda

Tutti i giochi di parole e i doppi sensi che sono stati profusi a piene mani nello scorso decennio a partire dai nomi in codice e poi di battesimo delle diverse versioni di OS X vanno archiviati. La serie dei grandi felini si è infatti chiusa e Mavericks, OS X 10.9, corrisponde al toponimo di una spiaggia ben nota agli appassionati californiani di surf. Partiamo dunque con un *calembour* in tema surfistico che vuole al tempo stesso sottolineare il carattere "pro" di questo libro: molto più spesso di quanto facciamo nella *Guida Completa* a OS X indagheremo sotto la superficie del sistema operativo Mac.

Nulla potrebbe essere più in profondità dell'argomento di questo capitolo, ossia il kernel, il nucleo di OS X attorno al quale sono posizionati gli strati di software che portano all'esperienza utente che tutti siamo abituati a conoscere.

Il nucleo di OS X si chiama Mach, ed è stato creato nella sua forma iniziale presso l'università Carnegie Mellon, per essere ovviamente ulteriormente sviluppato e arricchito da Apple in tutti questi anni.

Mach presenta varie caratteristiche interessanti, la cui presenza rende Mavericks più brillante in vari casi specifici. Alcune di queste caratteristiche sono state introdotte negli anni passati; altre sono una novità. Per alcune figure professionali queste nozioni possono essere importanti e persino suggerire un aggiornamento immediato alla versione di OS X più recente possibile.

In questo capitolo

- **La moltiplicazione dei processori**
- **La distribuzione dell'elaborazione**
- **Grand Central Dispatch**
- **Più che mai sessantaquattro**
- **Il distacco del Retina**
- **GPU e CPU**
- **OpenGL 4.1**
- **Lucchetti più piccoli**

La moltiplicazione dei processori

Quasi tutti i Mac venduti sino al 2005 possiedono un solo processore e quel processore sa eseguire una sola istruzione per volta. Questo modo di lavorare dei processori recenti ma non recentissimi viene qualche volta chiamato *seriale* o *single core*. Tutti i Mac in produzione oggi e la quasi totalità di quelli venduti dall'inizio del 2006 in poi lavorano invece con più processori e/o con processori *dual core*, i quali sono in grado di eseguire più istruzioni per volta. È abbastanza intuitivo il fatto che un processore dotato di parallelismo interno, cioè capace di eseguire più istruzioni contemporaneamente, sia migliore di uno che non ne è capace, ma le prestazioni di un computer non dipendono certo solo da questa prerogativa. Il sistema operativo Mac OS fino alla versione 9, di serie sui Mac fino a fine XX secolo, non era in grado di lavorare veramente bene con più di un processore per volta. Anche se esistevano già alcuni modelli estremamente costosi di Mac dotati di più di un processore, l'uso in parallelo di queste due unità di calcolo incontrava severe limitazioni, al punto che venivano utilizzati solo per alcuni compiti estremamente specifici. Con l'arrivo di OS X, invece, è stato il sistema operativo stesso a occuparsi della distribuzione del carico di lavoro su tutti i processori disponibili, con alcune limitazioni. Il computer, in ogni momento, esegue una gran quantità di programmi più o meno contemporaneamente. Una visita al programma Monitoraggio Attività, che si trova nella cartella *Utility* della cartella *Applicazioni*, mostra numerosi programmi aperti contemporaneamente sul computer. Tutti i programmi elencati sono in questo momento presenti nella memoria centrale e competono per l'attenzione del microprocessore. È evidente, dunque, che il computer ha sempre molto lavoro da svolgere. È perfettamente possibile immaginare un Mac che lavora con un centinaio di processori simultaneamente presenti e attivi: basterebbe dedicare un processore a ciascuno dei processi che vediamo attivi in quel programma. Tuttavia non sarebbe un uso particolarmente efficace della potenza di questo ipotetico Mac da cento processori: infatti, la maggior parte dei processi visibili nel monitor delle attività richiede una quantità minima di potenza di elaborazione per funzionare.

Per esempio, il programma LoginWindow, che è perennemente attivo sul Mac, serve quasi esclusivamente a consentire la connessione al computer digitando nome e password. Il programma deve restare sempre aperto, in modo da essere a disposizione quando un utente corrente smette di lavorare perché l'utente successivo possa prendere il possesso del Mac (oppure l'utente attuale possa riprenderne il possesso dopo un periodo di pausa). È chiaro che destinare un intero processore al lavoro di un task semplice come LoginWindow sarebbe un colossale spreco di potenza di calcolo (Figura 1.1).

Molto più interessante è la possibilità di dedicare più processori contemporaneamente a quei programmi applicativi che eseguono calcoli estremamente complicati, come per esempio Photoshop, con i suoi filtri per l'elaborazione delle immagini in qualità fotografica, oppure iMovie, con la sua necessità di elaborare tutti i fotogrammi di un intero film (Figura 1.2). Qui però la flessibilità del sistema operativo OS X non è più sufficiente da sola. Anche se il sistema operativo è abbastanza intelligente da distribuire i programmi aperti su tutti i processori disponibili, la sua autonomia non è sufficiente per consentire a un'applicazione di lavorare contemporaneamente su più processori: infatti un programma, normalmente, non è nient'altro se non una sequenza di operazioni che vengono eseguite dall'inizio alla fine. Un programma come iMovie può lavorare nello stesso tempo con più processori soltanto quando i suoi programmatori intuiscono la possibilità di sfruttarli

tutti in parallelo; ovvero, il programmatore che si trova a dover elaborare uno stesso effetto su più fotogrammi destina ciascuno di quei fotogrammi a un differente processore, in modo che, per esempio, l'elaborazione di quattro fotogrammi avvenga simultaneamente su altrettanti processori. In questo modo l'elaborazione dei quattro fotogrammi termina in un quarto del tempo che sarebbe necessario a un Mac dotato di un unico processore operante alla medesima frequenza di elaborazione in GHz. Quindi soltanto i programmi che sono stati creati dai loro autori per riconoscere e sfruttare la presenza contemporanea di più processori (o più nuclei all'interno del medesimo processore, il che è pressappoco la stessa cosa) ricorrono a questa caratteristica dei Mac moderni.

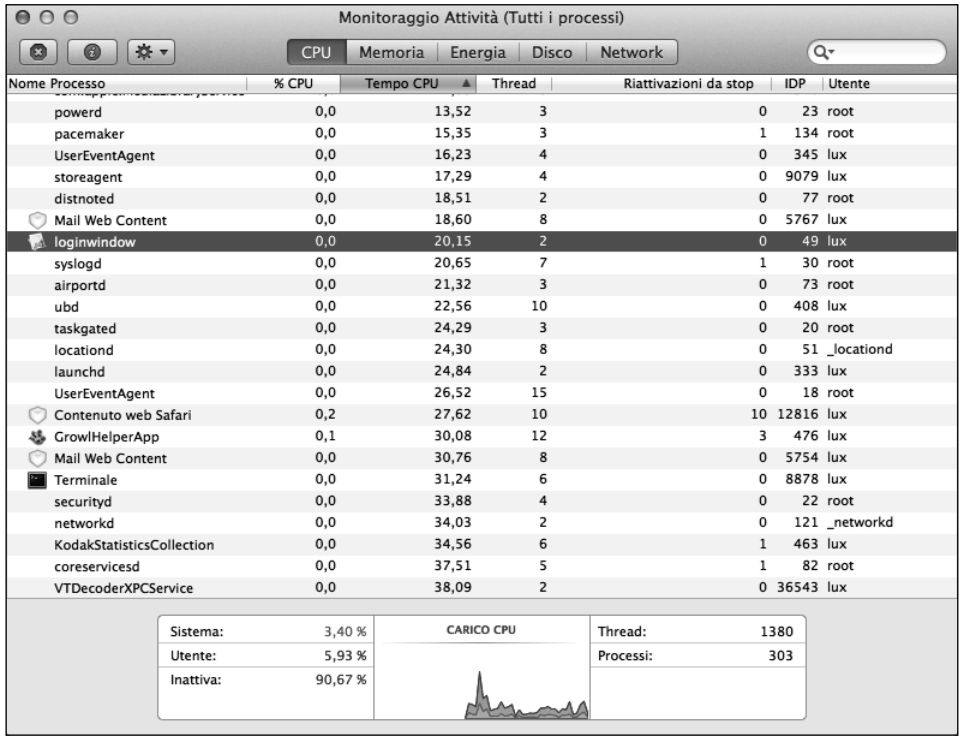

Figura 1.1 Nell'interfaccia grafica i programmi attivi sono molti meno delle decine di processi che il computer mantiene sempre in funzione, tra i quali LoginWindow, un programma che rimane attivo tutto il tempo in attesa che qualche utente si colleghi al sistema.

Non tutti i programmatori compiono lo sforzo necessario a rendere i loro programmi così intelligenti: dunque non tutte le applicazioni sono così intelligenti da dividere il loro carico di lavoro su tutti i processori disponibili. Soltanto i programmi che richiedono una gran quantità di potenza di calcolo sono internamente parallelizzati in modo da sfruttare tutta la potenza dei moderni processori dual core e quad core. Questo ha senso, poiché nessuno vuol pagare il doppio un applicativo che, nella pratica, si mostra soltanto dell'uno percento più veloce rispetto a quello sviluppato nella metà del tempo e quindi che costa la metà.

Figura 1.2 Se solo programmi come iMovie fossero capaci di ridistribuire il loro lavoro su più processori! In realtà è possibile, se tutti i programmatori operano nel modo giusto.

È importante notare che un programma moderno e ben scritto, per esempio Adobe Photoshop CS6, potrebbe dimostrarsi meno prestante di quanto dovrebbe sui Mac moderni dotati dell'ultima versione di OS X. Per esempio, se inserite in Photoshop ultima versione un filtro (o plug-in) vecchiotto, allora è certamente possibile che l'esecuzione di questo filtro rallenti Photoshop, perché l'autore di quel filtro non aveva pensato a parallelizzarne l'esecuzione. Sarà sufficiente aggiornare il filtro all'ultima versione, o in alternativa sostituirlo con un filtro equivalente ma sviluppato in tempi più moderni o da un programmatore più abile, per vedere aumentare a dismisura le prestazioni di Photoshop quando si fa uso di quel filtro. Qualcuno potrebbe a questo punto chiedersi: ma se questa è la realtà dei fatti, perché gli ingegneri che progettano i moderni microprocessori non li realizzano con un solo nucleo ma a una velocità di esecuzione della singola istruzione migliorata su quell'unico nucleo?

In effetti, questo è quello che si è fatto sino al 2005 circa, dopodiché, però, gli ingegneri hanno incontrato un limite non più valicabile: la velocità della luce. I circuiti non possono trasmettere informazioni a una velocità superiore, e ogni tentativo di velocizzare i processori, ovvero l'esecuzione di ogni singola istruzione all'interno del processore, andava a scontrarsi contro quel limite. I processori stavano diventando sempre più ingordi di corrente elettrica, il loro raffreddamento sempre più problematico e a causa di questi limiti fisici erano sempre meno utilizzabili nei computer portatili. È stato dunque deciso di concentrare le forze degli sviluppatori sulla possibilità di introdurre più unità di elaborazione all'interno di un unico microprocessore: è così che sono nati i processori *dual core* e poi *quad core* che trovate nei moderni Mac.

Il lettore del nostro libro a questo punto potrebbe domandarsi: non è possibile che io abbia investito moltissimi soldi per comprare un computer moderno e in realtà questa

macchina, alla prova dei fatti, nel mio lavoro quotidiano, si dimostri veloce tanto quanto il mio computer precedente? La risposta è affermativa: chi usa programmi non adattati per la realtà dei moderni sistemi dual core, come per esempio Microsoft Word 2011, potrebbe trovarsi di fronte a un programma che non beneficia affatto della presenza di un processore dual core o quad core. Resta il fatto che in un Mac moderno difficilmente in un dato momento è in esecuzione un solo programma: se siete abituati a tenere aperte più applicazioni contemporaneamente, magari per scaricare un documento da Internet o per ricalcolare l'indice del disco rigido con Spotlight mentre state scrivendo una lettera usando Microsoft Word, sicuramente il vostro computer sarà più efficiente usando un sistema dual core rispetto a quanto lo sarebbe stato con un singolo processore, ma la differenza non sarà marcatamente sensibile. Chi invece ha acquistato un sistema a due o più processori perché sa di avere necessità di sfruttare tutta la potenza del calcolo parallelo, potrebbe chiedersi se il suo sistema sia davvero efficacemente utilizzato, cioè se sia l'hardware, sia il sistema operativo sia il software in uso siano combinati al meglio per dare la massima velocità di elaborazione.
Per ottenere le prestazioni migliori da un Mac moderno occorre pertanto badare a tre fattori distinti: bisogna possedere la versione più recente del sistema operativo, naturalmente un Mac dotato di un numero quanto più grande possibile di nuclei di elaborazione (per esempio un Mac Pro) e una versione sufficientemente recente e ben scritta del programma sul quale state lavorando.
Come si capisce facilmente, è proprio quest'ultimo l'aspetto più problematico. Come capire se l'applicativo sfrutta al meglio tutti i core, i nuclei a nostra disposizione? Un sistema piuttosto semplice sfrutta un software gratuito compreso in Xcode di Apple: Time Profiler.

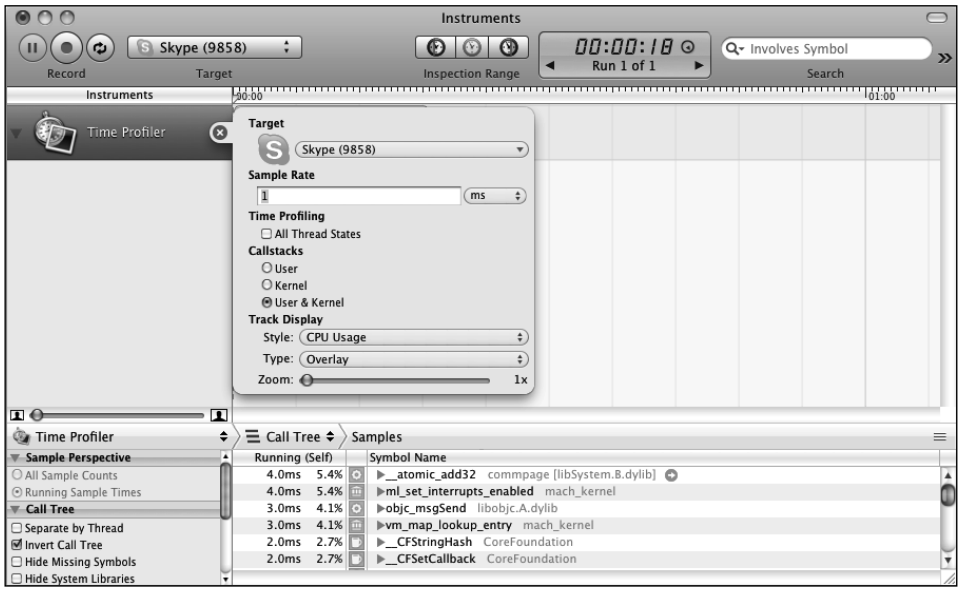

Figura 1.3 Time Profiler, uno dei programmi presenti in Instruments, consente il monitoraggio di tutte le funzioni di OS X. Qui dà un'occhiata a quello che combina Skype.

Time Profiler permette di effettuare una fotografia dell'andamento del sistema ogni millesimo di secondo (o anche più spesso, se avete una macchina potentissima e un programma che deve spaccare il bit in quattro) e di mostrare in un grafico le scoperte. Potete innanzitutto effettuare un'elaborazione cronometrando i tempi e quindi la rapidità di esecuzione; dopodiché potrete usare il sistema per capire quale porzione del programma occupa il processore per la maggior parte del tempo e rifletterci in modo da ottimizzarne al massimo il codice e l'efficienza.
Se il programma è stato scritto correttamente vedrete che la funzione (o le funzioni) più critica è in grado di sfruttare tutti i nuclei che macinano in parallelo. Non dovete aspettarvi che un sistema dual core sia effettivamente due volte più veloce del medesimo sistema che lavora su un nucleo solo: una velocità migliorata del 70% circa è la norma quando si lavora su un sistema dual core ben programmato. Parte della potenza di calcolo viene usata per far girare il sistema operativo stesso e garantire la sincronia dei processi.

La distribuzione dell'elaborazione

Come abbiamo visto, il vostro computer possiede almeno due unità di elaborazione, cioè è dotato di circuiteria che può lavorare su due (o più, sino a sedici) applicazioni contemporaneamente. OS X approfitta di questo stato di cose e distribuisce automaticamente il carico di lavoro sui processori disponibili.
Sino a pochi anni fa per un programmatore era difficilissimo scrivere un'applicazione che riuscisse a sfruttare tutta la potenza di calcolo divisa sui vari processori. In OS X le cose diventano più semplici se il programmatore adotta l'*Open Computing Language* (OpenCL). Questo linguaggio – che non va confuso con il popolare sistema OpenGL, usato per lavorare in grafica e di cui parliamo nel seguito del capitolo – è un'estensione del popolarissimo linguaggio C, di gran lunga il più usato dai programmatori professionisti. Chi scrive un'applicazione in OpenCL può sostanzialmente dire al sistema operativo "ho bisogno di fare queste sedici cose, arrangiati un po' tu". Su un Mac Pro da otto nuclei di elaborazione, OS X si può preoccupare di assegnare due compiti a ogni nucleo. Quando la nostra immaginaria applicazione con sedici elaborazioni in corso gira invece su un portatile MacBook con due nuclei, ciascuna unità di elaborazione potrebbe vedersene assegnati otto. In realtà le cose vanno anche meglio di così, perché OpenCL considera e sfrutta anche la scheda grafica. Le moderne schede video che equipaggiano i nostri Mac sono dotate di grandi capacità di elaborazione. Non sono flessibili e generaliste come il processore centrale; si tratta di unità specializzate nel macinare numeri, elaborare tabelle e calcolare vettori, cioè quelle operazioni che permettono di mostrare velocemente punti colorati a video. Ma non è poco, e i programmi che devono elaborare audio, video e foto trarrebbero un gran vantaggio dalla capacità di caricare la scheda video con parte del lavoro numerico da sbrigare. Qualche dato: un moderno MacBook Pro monta "soltanto" quattro nuclei Intel sul processore centrale Core i7, ma è dotato di ben 384 coprocessori grafici sulla scheda grafica GeForce GT 650M (`http://xrl.us/bpu34r`). In altre parole, un programma originariamente scritto per operare sulla sola unità centrale che lavora su una traccia audio usando un singolo nucleo di elaborazione può teoricamente diventare 387 volte più veloce se lo sviluppatore lo ritocca per sfruttare OpenCL. Nella pratica, si ottiene una velocizzazione superiore di circa quattro volte rispetto al passato (Figura 1.4).

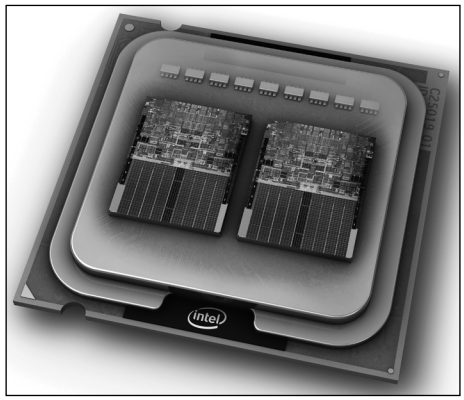

Figura 1.4 Lo schema interno di un moderno processore Intel quad-core, a quattro nuclei di elaborazione. (Fonte: Intel.)

OpenCL è basato su LLVM, un ottimizzatore di codice binario che prende un programma compilato e ne risistema le istruzioni in modo da facilitarne l'esecuzione al processore. Gli ingegneri di Apple l'hanno creato cercando di renderlo simile per sintassi e modo d'uso alla summenzionata libreria OpenGL per renderne immediato e comprensibile l'utilizzo agli sviluppatori di videogiochi, un genere di programmi che sembra non aver mai sufficiente potenza di elaborazione a disposizione.

10.9

Mavericks è equipaggiato con la versione 1.2 di OpenCL e introduce tra l'altro il supporto per le schede grafiche su chip HD 4000 Intel, che su MacBook Pro affiancano la scheda grafica a grandi prestazioni.

Grand Central Dispatch

L'architettura che permette a OpenCL di funzionare in modo efficace è stata chiamata da Apple *Grand Central Dispatch* e introdotta con OS X 10.6 Snow Leopard. L'idea di base è semplice. Un'applicazione creata per le versioni di OS X da 10.0 a 10.5 è una sequenza di istruzioni che vengono eseguite dalla prima sino all'ultima, linearmente. Immaginate per esempio iMovie impegnato nell'applicare una nuova colonna sonora, o un effetto visuale, a un filmato. La medesima applicazione Mac ottimizzata per Snow Leopard, Lion o Mountain Lion, come abbiamo detto, verrà frammentata dal suo sviluppatore in più compiti che possono venire eseguiti simultaneamente. Tornando al nostro esempio, applicare una dissolvenza in iMovie è un lavoro parcellizzabile, perché ogni fotogramma può venire elaborato come lavoro individuale.

Il programmatore usa istruzioni OpenCL per confezionare ogni compito in un pacchetto. I pacchetti da elaborare viaggiano in Grand Central Dispatch e raggiungono uno smistatore che inoltra ciascuno di essi al nucleo di elaborazione più adatto a completarlo. Su una macchina potente ogni compito viene accoppiato al nucleo di elaborazione più adatto; su una macchina economica o su un portatile che sta risparmiando batteria e dunque tiene spenta parte della sua elettronica, i compiti da svolgere vengono accodati "davanti" ai nuclei di elaborazione disponibili. Le code di solito vengono smistate nell'ordine di arrivo – un po' come succede dal salumiere, dove ogni cliente prende un

bigliettino quando entra nel negozio – ma in alcuni casi il sistema blocca l'esecuzione di un pacchetto. Per esempio, se abbiamo ordinato a iMovie di abbassare il volume del brano e di schiarire il video, è possibile che il lavoro sull'audio finisca prima dell'elaborazione dei fotogrammi e in questo caso il pacchetto di lavoro "sincronizza audio e video" aspetterà fuori dalla coda.
Se necessario, Grand Central Dispatch ricompila il compito che trova nel pacchetto usando un compilatore *just in time*, una tecnologia resa popolare quindici anni fa dal linguaggio Java. Per esempio, Grand Central Dispatch potrebbe decidere che il compito "aumenta il volume della canzone del 10%" è ottimale per il processore della scheda grafica, mentre conviene affidare il lavoro "impagina i sottotitoli" a un processore Intel. Se questo è il caso, il pacchetto verrà predisposto in modo adeguato perché l'elettronica della scheda grafica lo elabori al meglio. In un Mac con molta memoria a disposizione, il codice che aumenta il volume della canzone potrebbe a un certo punto trovarsi nella memoria centrale in due (o più) versioni: quella ottimizzata per il processore centrale Intel e quella perfetta per il coprocessore della scheda grafica. In questo caso, ogni volta che c'è bisogno di lavorare su un audio il sistema giudicherà quale sia la coda più veloce e assegnerà quello specifico audio al nucleo di elaborazione più scarico, sapendo di avere già pronta la sequenza di istruzioni eseguibili che gli è più congegnale.
Gli ingegneri di Apple hanno anche immaginato (e sviluppato e brevettato) un complesso sistema che consente a ciascun compito di condividere memoria, inviare flussi di dati e scambiare messaggi con gli altri.
A chi non è un esperto di informatica girerà un po' la testa. Per noi, che ne mastichiamo, l'argomento è davvero affascinante, ma ci fermiamo qui senza entrare in tutti i dettagli tecnici; chi è interessato li trova nella documentazione Apple per gli sviluppatori, scaricabile dal sito `https://developer.apple.com/`. Per un'infarinatura, cominciate consultando il documento PDF `http://xrl.us/bo8voo`, descrittivo in modo sintetico delle novità di Mavericks a livello di tecnologie fondamentali.
L'unica cosa davvero importante e indispensabile da comprendere per tutti gli utilizzatori di un Mac è che OS X, a differenza di tutti gli altri sistemi operativi esistenti, trova tutti i componenti del vostro calcolatore che possono venire utilizzati per elaborare dati e li mette tutti quanti al lavoro contemporaneamente. Questo ha delle conseguenze importanti ma non evidenti, se ci fermiamo a riflettere.
Prima riflessione. Immaginate di possedere un Mac Pro e di decidere per l'acquisto di una nuova scheda video. Vi verrebbe naturale togliere la vecchia scheda e rimpiazzarla con quella nuova, che collegherete al monitor. Sbagliato! Se lasciate entrambe le schede dentro al sistema, il vostro calcolatore andrà più veloce, perché Grand Central Dispatch potrà usare tutte e due le schede video come motori di elaborazione dati, anche se una delle due non è collegata ad alcun monitor.
Seconda riflessione. Aggiornare i propri programmi a una versione recente è sempre una buona idea: si aggiungono funzionalità, si rimediano difetti. Quando però il nostro sistema operativo è Snow Leopard o Lion abbiamo un eccellente motivo in più per prestare la massima attenzione alle nuove versioni, specialmente quando si parla di programmi ingordi di potenza di calcolo (cioè quelli che lavorano su video, suono e grafica ad alta risoluzione, compresi i videogiochi). Se gli autori dell'applicazione, infatti, prendono il programma che avevano scritto per OS X 10.5 e lo aggiornano in modo da passare all'uso di Grand Central Dispatch, ecco che noi possessori di una copia di OS X 10.6 con l'aggiornamento ci troviamo un programma tre volte più veloce nell'eseguire i lavori

che gli affidiamo. Un esempio concreto? I filtri di Adobe CS5.5 e CS6, che sfruttando le possibilità di Grand Central Dispatch sono percettibilmente più veloci di quelli di cui era dotata la CS4. Come si riconosce un programma ottimizzato per Grand Central Dispatch? Gli autori dovrebbero preoccuparsi di dichiarare a chiare lettere i vantaggi per chi possiede una versione recente di OS X, ma c'è un indizio facile da riconoscere per sapere se il programma a cui siamo interessati fa al caso nostro. Se sulla confezione (o sul sito web) c'è scritto che richiede OS X versione 10.6 o superiore, certamente il parallelismo di cui è capace il nuovo sistema operativo viene sfruttato a fondo.

Più che mai sessantaquattro

OS X 10.7 è stato il primo sistema operativo Apple che per funzionare richiede una moderna CPU, in grado di elaborare le informazioni in blocchi di 64 bit. I suoi predecessori invece erano in grado di accontentarsi anche delle più modeste architetture a 32 bit.
Il supporto dei 64 bit in OS X si è gradualmente evoluto nel tempo. Alla sua prima apparizione in un Mac, quel tipo di processore veniva costretto dal nostro sistema operativo a comportarsi come un 32 bit più veloce. Ma tutti i processori a 32 bit sono limitati a montare non più di 4 GB di memoria, perché possono assegnare a ciascuna locazione in memoria un indirizzo lungo al massimo 32 bit e con questi si possono rappresentare al massimo 4.294.967.296 numeri diversi. Un processore a 64 bit infrange questo limite e nel 2004 i Power Mac G5 sono stati i primi calcolatori Apple capaci di utilizzare quantitativi di RAM superiori, iniziando così il percorso verso un completo sfruttamento delle caratteristiche superiori delle più moderne architetture. Il nucleo di OS X viene dunque riprogettato per scatenare tutta la potenza dei processori contemporanei. La versione 10.4 di OS X può gestire sino a cento applicativi aperti simultaneamente, che diventano 256 con la 10.5. Sempre grazie a OS X 10.5 Leopard, gli sviluppatori possono avanzare una serie di altre richieste "pienamente a 64 bit" al sistema operativo, in maniera da sfruttare al massimo il processore: non è un'esigenza sentita da tutti e quindi non è un'occasione colta spesso, prevedibilmente. Per esempio, un programma per il trattamento testi passa oltre il 99% del suo tempo ad attendere che premiate un tasto, anche se siete dattilografi professionisti e digitate a tutta velocità. Ma per altre applicazioni, quelle per le quali la velocità di elaborazione non è mai troppa, il passaggio da un'architettura a 32 bit a una a 64 bit significa un vantaggio prestazionale del 15 percento circa.

CORE INGRATO

Se avete un dubbio sulla compatibilità con Mavericks del vostro calcolatore sempre valido ma non più nuovissimo, conducete un primissimo test dando nel Terminale il comando `uname -a`: se il risultato non termina con `x86_64`, il processore non è in grado di funzionare a 64 bit e certamente l'installazione di Mavericks è esclusa.

In Mavericks, come detto, tutto è sempre e solo a 64 bit e i vantaggi sono notevoli. Prima (in realtà prima di Lion), un Mac finiva per caricare due copie del software di base Cocoa (la parte del sistema operativo che sa come aprire e spostare le finestre sullo schermo, gestire i clic del mouse e così via): una copia a 64 bit per le applicazioni riviste per sfruttare quella tecnologia e una seconda copia a 32 bit per tutte le altre. Da Lion in avanti, invece, il sistema è interamente a 64 bit, quindi c'è una sola copia in memoria, per cui si usa meno memoria per Cocoa e ne resta libera di più per i nostri dati.

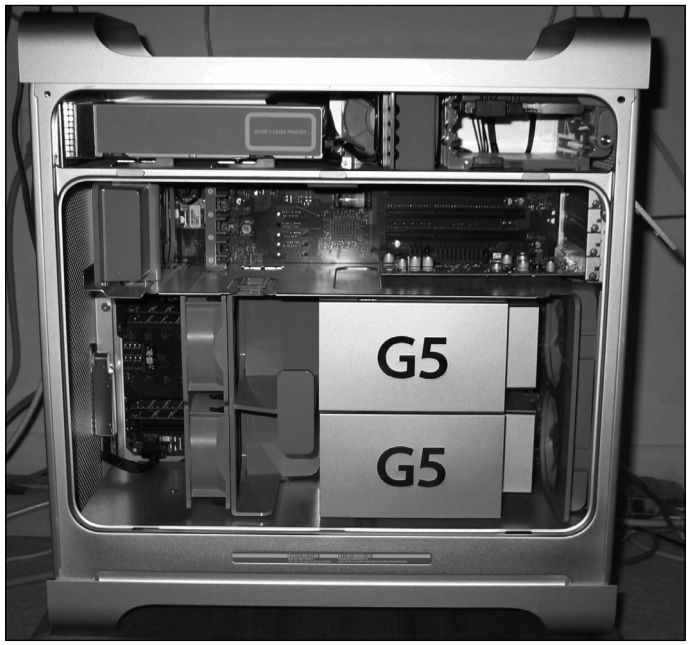

Figura 1.5 Prima ancora del passaggio ai processori Intel, Apple aveva già introdotto Mac con CPU capaci di elaborazione a 64 bit, come il Power Mac G5.

Il distacco del Retina

Nel 1984, i primi Mac apparivano sul mercato e incorporavano piccoli monitor in bianco e nero con una densità di 72 punti per pollice. Questo significa che quando sul monitor venivano accesi in bianco 72 pixel consecutivi ne risultava una linea lunga appunto un pollice, ovvero 2,54 centimetri. Una barra di scorrimento posizionata sul bordo destro della finestra di un programma di videoscrittura era, a quei tempi, larga 16 pixel, ovvero 5,6 millimetri.

Sono passati quasi trent'anni da allora e nell'informatica è cambiato quasi tutto. I moderni monitor sono ultrapiatti, hanno la capacità di mostrare milioni di colori e i loro pixel sono divenuti più piccoli. Il monitor di un MacBook Air, per esempio, ha 135 punti per pollice. Però una cosa è rimasta identica al 1984: in OS X 10.7 una barra di scorrimento è larga 16 pixel; esattamente come in Mac OS 1.0. Nel frattempo, però, quei 16 pixel sono diventati equivalenti a soli 3 millimetri.

Se, come gli autori di questo libro, voi nel 1984 c'eravate già, vi sarà probabilmente capitato di invecchiare nel frattempo. Di conseguenza, magari la vostra vista non sarà migliorata. Con 135 punti per pollice se avete più di quarant'anni sul groppone la barra dei menu diviene così piccola che le singole voci risultano quasi illeggibili a occhio nudo.

La situazione sta raggiungendo proporzioni tali da richiedere un intervento radicale, e nessuno lo sa meglio di Apple. La versione 3 di iPhone, introdotta sul mercato nel 2009, offriva uno schermo incorporato da 163 punti per pollice; la versione 4, quella del 2010, passò a 326. Steve Jobs, durante la presentazione, lo battezzò *display Retina*, perché i punti sono diventati così piccoli che l'occhio umano non è più in grado di discernere

la discontinuità tra un punto e i suoi vicini. Se i punti del monitor divenissero ancora più piccoli non si avrebbe alcun vantaggio percettivo (e dunque la cosa non accadrà). Viceversa, se collegassimo un monitor con la risoluzione dello schermo di iPhone 4 a un Mac senza cambiare nulla nel software, ci troveremmo con una barra di scorrimento larga 1,2 millimetri e dovremmo allevare un rapace per aiutarci a leggere le scritte sulla barra dei menu.
Apple ha dunque messo in campo una soluzione semplice ma efficace, la stessa già rodata su iPhone e affermata con la terza generazione di iPad: raddoppiare linearmente, ovvero quadruplicare come area, le dimensioni degli elementi dell'interfaccia utente come pulsanti, menu e icone (Figura 1.6).
In Mavericks l'indipendenza dalla risoluzione è una funzione a disposizione dell'utente con controlli e preferenze sui Mac equipaggiati con un monitor ad altissima risoluzione (tecnicamente detto anche *monitor HiDPI*). L'inizio di questa nuova epoca è stato segnato dal MacBook Pro Retina di giugno 2012, dotato di 2880×1800 pixel fisici. Quel sistema consente di variare la risoluzione effettiva dello schermo tra cinque valori, da 1024×640 a 1920×1200 punti (che vengono resi utilizzando i 2880×1800 di cui sopra, quindi con finissimo dettaglio, al punto che diventa difficile o impossibile distinguere i singoli pixel effettivi).
Quando l'utente imposta lo schermo del MacBook Pro Retina sulla risoluzione di 1920×1200, il calcolatore disegna nella memoria della scheda grafica interna una scrivania virtuale a 3840×2400 punti (oltre nove milioni di pixel!) per poi trasferirla nei "soli" pixel fisici di cui è dotato lo schermo. È uno sforzo massiccio per la pur potentissima scheda grafica di cui è dotato l'apparecchio. Quando si scorreva una pagina di Facebook, ricchissima di elementi grafici, in Safari sotto Lion, il MacBook Pro Retina riusciva a generare circa venti fotogrammi al secondo, contro i cinquanta di cui sono capaci i fratelli minori che non sono appesantiti da tutti quei milioni di pixel. Ma Lion non era ottimizzato per il display Retina. In Mavericks, la medesima esperienza di scorrimento fa misurare anche più di trenta fotogrammi al secondo, un miglioramento del cinquanta percento.
Molte applicazioni Apple fornite con Mavericks sono ottimizzate per il display Retina, ma naturalmente al mondo non c'è solo Apple. Bisognerà dunque attendere una nuova generazione di applicazioni di terze parti in grado di comportarsi come si deve in un ambiente ad alta risoluzione, incorporando tra l'altro immagini realizzate in qualità elevata dagli illustratori originali. L'onere di quest'ultima realizzazione spetta agli sviluppatori indipendenti.

GPU e CPU

Da quando Apple ha compiuto la transizione ai processori Intel per tutti i Mac in vendita, gli utenti Mac hanno largamente smesso di interessarsi delle problematiche prestazionali dei microprocessori. È un peccato, perché nel frattempo sono successe alcune cose molto interessanti: parlarne non rientra nell'ambito di questo libro, tranne che per un aspetto. Questo aspetto riguarda la grafica, che da sempre è molto importante nel mondo Mac. Tutti sanno che il microprocessore (o CPU, *Central Processing Unit*) è il componente cruciale di un personal computer, al punto che spesso la sua indubbia importanza viene sopravvalutata.

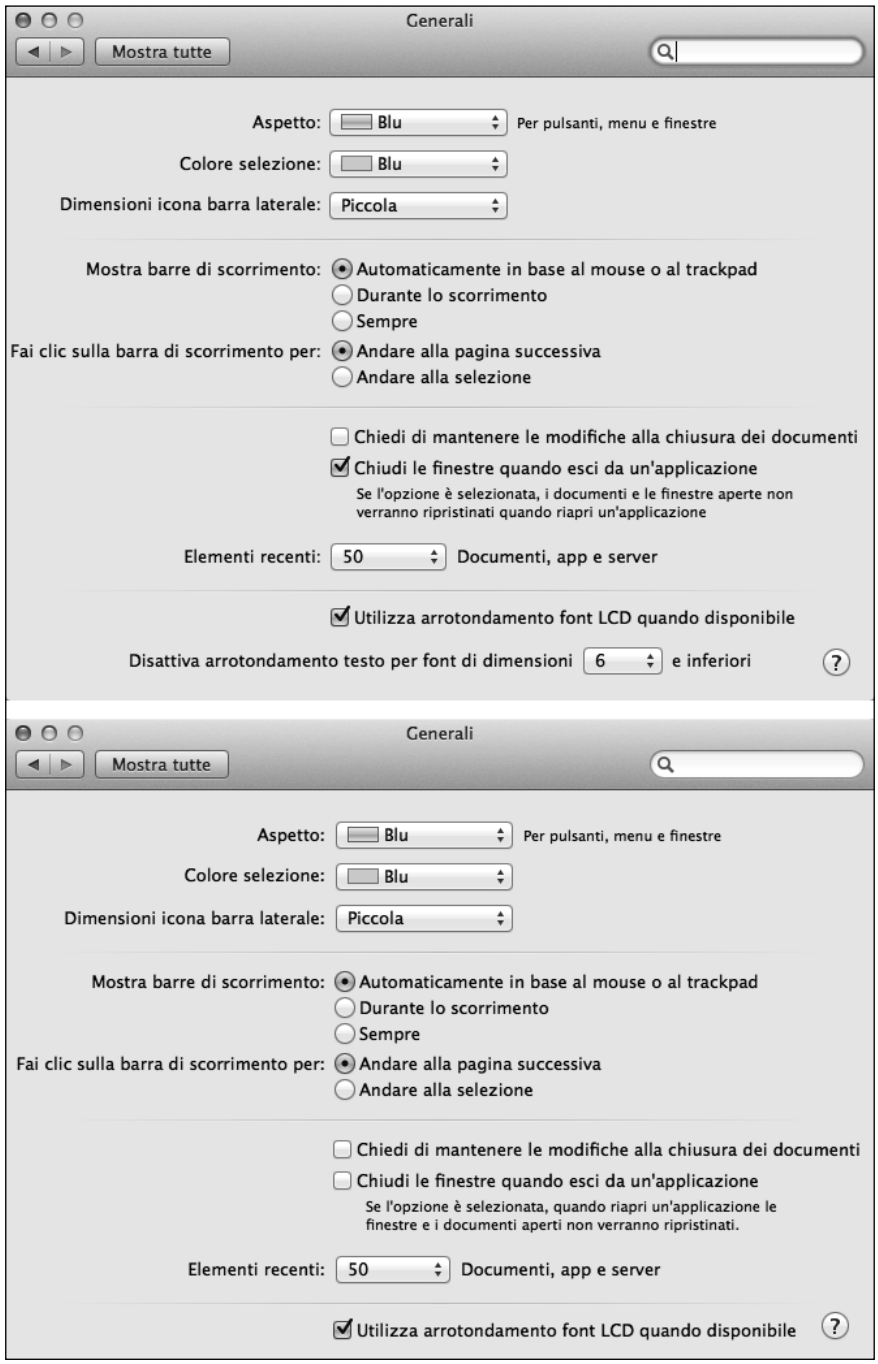

Figura 1.6 L'evoluzione di OS X procede anche attraverso il distacco tra misure dell'interfaccia e disponibilità dei pixel sullo schermo. Ne è prova indiziaria l'evoluzione del pannello Generali delle Preferenze di Sistema da Mountain Lion a Mavericks: manca una funzione di arrotondamento dei caratteri legata alle caratteristiche dello schermo.

Tutti i computer moderni, però, sono anche dotati di una GPU (*Graphical Processing Unit*), una specie di processore specializzato responsabile della gestione della scheda video, ovvero della trasmissione dell'immagine al monitor. Negli ultimi anni le GPU sono diventate sempre più importanti: i modelli più recenti hanno prestazioni imponenti, una massiccia quantità di memoria dedicata (oltre un gigabyte, nei casi migliori) e la capacità di eseguire elaborazioni in modo indipendente rispetto al processore principale. I motivi per cui si è arrivati a questi estremi sono molti, ma due sono fondamentali. Primo, tutti i moderni personal computer possono venire usati per guardare video digitali (dischi DVD o filmati come quelli offerti in QuickTime), e c'è bisogno di molta potenza per decomprimere ogni singolo fotogramma dal formato minimizzato in cui si trova sul disco. Secondo, i moderni videogiochi sono così sofisticati che c'è bisogno di potenza dedicata per elaborare gli aspetti grafici della rappresentazione (il posizionamento degli oggetti in uno scenario tridimensionale, le ombreggiature e così via). Le moderne GPU, in sostanza, hanno prestazioni comparabili a quelle delle moderne CPU. Per alcuni aspetti limitati e specializzati sono anche superiori, per esempio nelle elaborazioni matematiche e anche per il parallelismo: una GPU è progettata per gestire schermi composti da milioni di punti luminosi, elaborandoli tutti contemporaneamente.

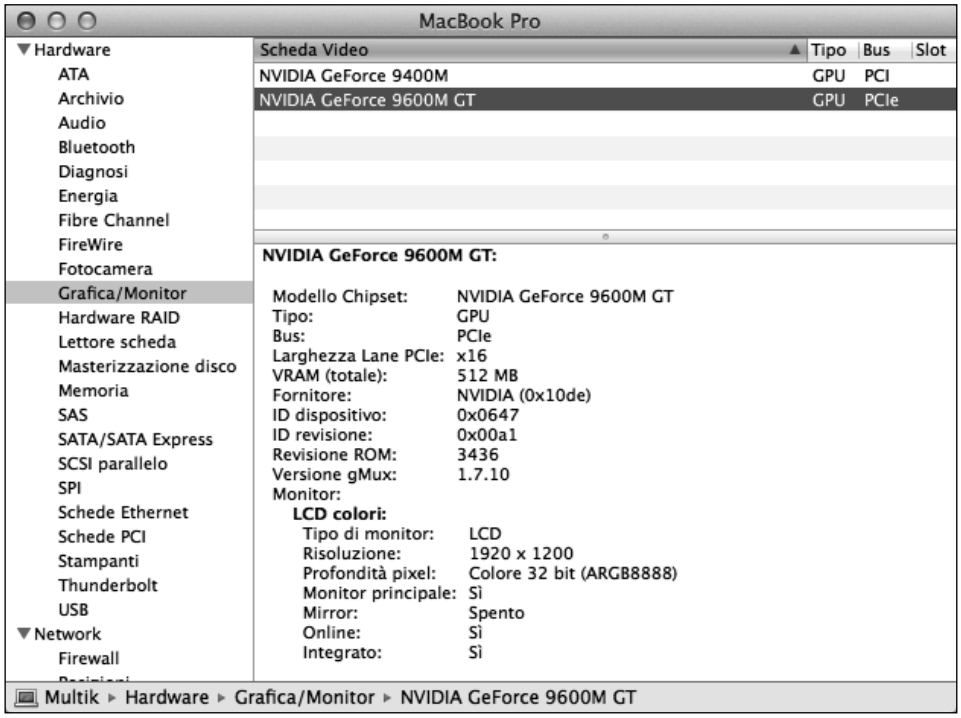

Figura 1.7 Le schede video diventano sempre più potenti, tanto che OS X assegna loro compiti di gestione della grafica assai onerosi allo scopo di liberare il processore per altre attività prioritarie.

A partire da OS X 10.4, Apple si è chiesta se non fosse il caso di demandare alla GPU una parte del carico di lavoro che tradizionalmente era affidato alla CPU. E ha trovato

una serie di compiti in cui i vantaggi sono cospicui. Così, OS X contiene una libreria con una sessantina di "effetti speciali" che possono venire spostati sulla GPU. Se il sistema operativo si trova a lavorare su un computer vecchiotto, o comunque dotato di una scheda video limitata in cui le prestazioni della GPU non eccellono, allora queste elaborazioni vengono eseguite dalla CPU.

OpenGL 4.1

OpenGL (*Open Graphics Library*) è un metodo standard ideato all'inizio degli anni Novanta con il quale i programmatori possono inviare ordini alla scheda video di un calcolatore, demandandole i dettagli realizzativi di funzioni come la rotazione di un oggetto grafico. In altre parole e per esempio, se un videogioco ha bisogno di rappresentare una piramide di legno e una piramide d'acciaio che, ruotando, volano l'una verso l'altra e collidono, allora il programmatore dovrà impegnarsi a calcolare le coordinate dei vertici della piramide e fornire l'immagine della superficie di legno e metallo. Il nostro sviluppatore non si preoccuperà troppo dell'opportunità di disegnare per prima la piramide a destra o quella a sinistra, anche se è possibile che su un determinato modello di scheda grafica ci siano notevoli vantaggi prestazionali in uno dei due casi. Il programmatore darà invece i suoi ordini usando il metodo standard OpenGL e lascerà che uno strato software creato dal produttore della scheda grafica si occupi dei dettagli come quello citato (Figura 1.8).

Figura 1.8 OpenGL è molto usato nell'ambiente dei programmatori indipendenti e dai produttori di videogiochi per la sua versatilità. Su Mac sono disponibili innumerevoli salvaschermo che utilizzano OpenGL per produrre effetti speciali durante l'inattività del computer. Provate ad aprire il sito MacUpdate (https://www.macupdate.com/) oppure App Store per Mac e cercate "screen saver".

10.9

OpenGL viene frequentemente rinnovato. OS X Lion ha offerto per la prima volta su Mac tutte le funzionalità che la caratterizzano, e anzi è arrivata a supportare completamente la sottoversione 3.2. Mavericks arriva al supporto di OpenGL 4.1. La versione corrente di OpenGL mentre questo libro viene scritto è la 4.4. Un certo ritardo nell'aggiornamento delle componenti base del sistema è una tradizione di Apple, anche comprensibile, in

quanto all'adozione di una versione X di un software deve corrispondere una ingente batteria di test di affidabilità, compatibilità e così via. Semmai c'è da essere contenti che il divario sia relativamente minimo. Si noti che la copertura di OpenGL su Mac è equivalente a quella utilizzata su iOS, GLKit. Questo facilita il lavoro dei programmatori che intendono offrire i loro programmi sia in versione iOS sia OS X, per esempio l'ambiente di scrittura *Textastic* o il gioco *Hero Academy*.

Lucchetti più piccoli

Un ulteriore cambiamento avviene ancora più in profondità dentro al nucleo del sistema operativo, anche in questo caso con la conseguenza di migliorare le sue prestazioni. Chi passava da Mac OS 9 a OS X notava immediatamente che il nuovo sistema operativo era molto più efficace e scattante quando doveva lavorare con i file sul disco. Meno evidente, ma perfettamente misurabile, era la perdita di prestazioni nel sistema di rete. Per rendere OS X compatibile con le macchine multiprocessore, sin dalla versione 10.0, Apple ha inserito nel nucleo del sistema operativo due *kernel funnel*: uno sul sistema di rete e uno sul resto del nucleo. Il *funnel* è un collo di bottiglia che obbliga i programmi a "mettersi in fila indiana" quando chiedono qualcosa al sistema. Nell'uso normale il funnel non si nota, ma per alcuni utilizzi comporta un calo significativo delle prestazioni. Per esempio, una macchina attiva su Internet deve tradurre ogni nome (come *www.apple.com*) in un indirizzo IP (come *17.254.0.91*) prima di accedervi; in alcuni casi si dimostra molto efficiente la scelta di cercare di risolvere più nomi in parallelo anziché scandire sequenzialmente l'elenco.
Un computer che agisce da server, per esempio, è inefficiente come sistema di posta elettronica ad alte prestazioni: quando deve spedire diecimila e-mail si dimostra francamente lento. Glen Anderson, il papà dello storico programma di posta elettronica Eudora (`http://eudora.com/`, ora `https://wiki.mozilla.org/Penelope`), documenta il problema nelle note tecniche del suo programma. Il suo apprezzato programma Eudora Internet Mail Server (`http://eudora.co.nz/`) in versione definitiva per OS X ha subìto un ritardo di quasi due anni perché Anderson ha deciso di riscrivere di sana pianta gran parte del codice di rete di OS X per aggirare il problema (Figura 1.9). Analogamente, anni fa cercare di impiantare un sito web molto frequentato su una macchina OS X era sconsigliabile se, come spesso accade, occorreva tenere traccia di tutti i visitatori deducendo (a fini statistici) dalle loro richieste la nazione e il provider da cui arrivavano. Di conseguenza, gli specialisti che nel 2001 decisero di mettere alla prova – cronometro alla mano – un sistema OS X Server si scontrarono spesso con risultati deludenti. Normalmente si trattava di esperti Unix che ignoravano la questione del funnel e pertanto non si capacitavano dei risultati, come accadde a Moshe Bar, giornalista di *Byte*, che documentò la superiorità di un'installazione Linux su OS X pur usando lo stesso hardware Mac.
Non a caso Steve Jobs, che amava mettere a confronto i Mac e i PC durante le sue presentazioni, era sempre concentrato sulle prestazioni dei filtri e delle azioni di Photoshop e mai lo si è visto vantarsi delle prestazioni in rete.
Nel 2005 finalmente Apple ha deciso di mettere mano al problema in occasione del rilascio di OS X 10.4. Il kernel funnel non è stato interamente eliminato, ma la sua incidenza è oggi drasticamente diminuita. Il blocco in Tiger avviene infatti a livello di singoli sottosistemi. Immaginate che in un supermercato dove esiste una sola cassa per

i pagamenti si passi a un nuovo sistema in cui ogni banco (verdura, macelleria, pesce, panetteria e così via) dispone di una cassa propria: è evidente che la maggior parte dei consumatori trarrebbe vantaggi dalla modifica. È proprio questo il passo avanti offerto da OS X 10.4, al cui interno non esistono più le due code relative alla rete e alla memoria, ma decine di code più snelle e rapide. Da questo punto di vista Mavericks eredita la struttura del funneling di Tiger; il problema sussiste nonostante siano passate altre cinque versioni del sistema operativo (Figura 1.10).

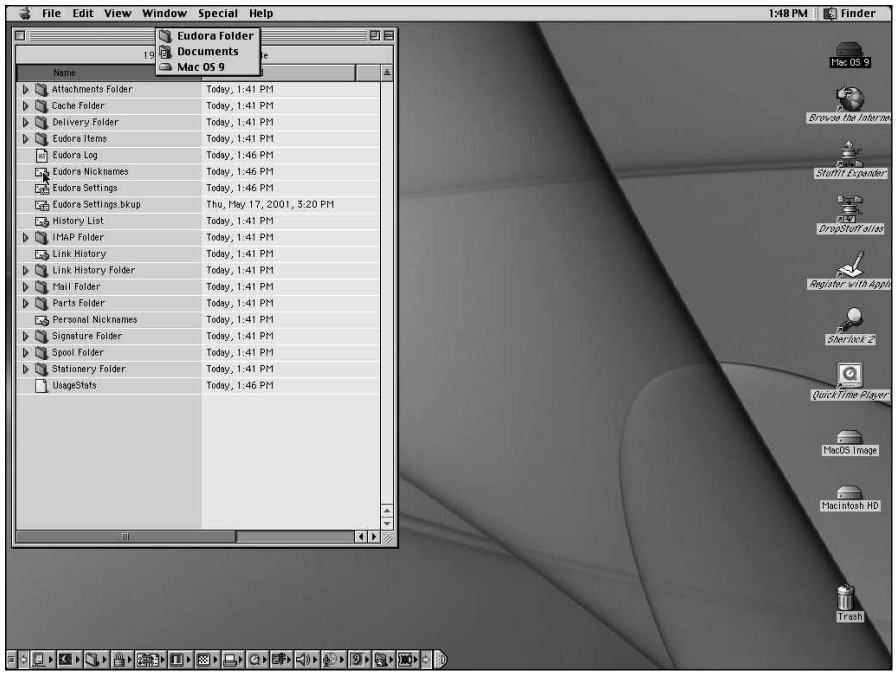

Figura 1.9 Eudora, qui mostrato nello splendore archeologico di Mac OS 9, è uno dei programmi più longevi nella storia di Mac OS. Forse proprio per questo il suo autore è stato il primo a documentare il problema del kernel funnel nelle prime versioni di OS X.

Figura 1.10 OS X 10.4 dista oltre un lustro dai nostri giorni, ma il problema del kernel funnel sussiste ancora.