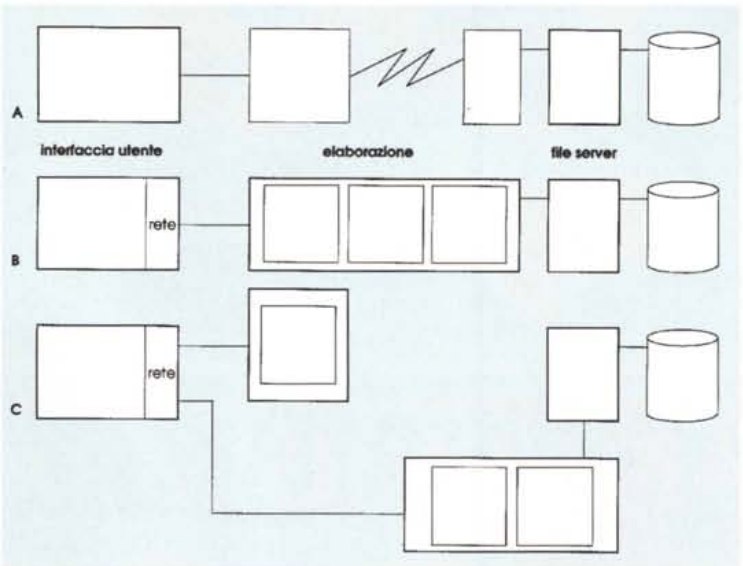
Il cliente ha sempre ragione

Client/server, rightsizing ed altri termini sono entrati a far parte del lessico quotidiano. Ma cosa vogliono dire esattamente? E da dove vengono? Abbiamo provato a spiegarvelo, facendo riferimento più alla storia, che offre confronti con cognizioni già note, che alle soluzioni progettuali o allo stato degli standard

di Leo Sorge

Per decenni l'informatica era gestita in un modo molto più semplice di quella odierna. Tutte le risorse hardware, ovvero l'unità centrale, le periferiche di uscita compresi i **terminali** e poi le reti, erano controllate da un software residente sulla sola unità centrale, che era l'unica ad elaborare. Stiamo parlando del **mainframe**, computer grandi come stanze e dalla potenza di calcolo pari a quella d'un 486 neanche DX2 ma dalle più spiccate attitudini di comunicazione, dato che erano stati progettati per gestire migliaia di terminali che, per quanto a caratteri e poco versatili, richiedevano dati contenuti in database di grandi dimensioni, fino all'ordine del gigabyte. I programmi che giravano su questi elaboratori erano enormi, perché risolvevano monoliticamente un problema complesso come l'amministrazione d'una azienda nella sua interezza, inglobando tutto: processo, terminali, periferiche d'uscita, rete. Inoltre la maggior parte del lavoro veniva svolto in **modalità differita**, per cui tra la richiesta dell'elaborazione e i dati finali passavano giorni, senza contare che tali risultati venivano stampati su enormi tabulati ed inviati per posta (non elettronica) ai destinatari, che ci facevano su le loro riflessioni con carta, penna e calamaio, ovvero a mano.



Dal cooperative computing al client/server distribuito

Quando ancora non era definito un vero modello, ci fu chi propose il cooperative computing, (A). In questo caso la logica del programma resta monolitica, ma alcune risorse in rete vengono chieste ed indirizzate direttamente.

Quando il collegamento in rete viene estratto dalla logica del programma, e questa viene suddivisa in moduli funzionalmente separati, allora abbiamo un client/server. In un primo caso (B) il client può avere la sola interfaccia utente e il server eseguire l'intera elaborazione. Se però il modello è distribuito (C), i vari moduli vengono eseguiti su macchine diverse, fermo restando la trasparenza dei meccanismi di rete, ignoti all'utente ma registrati dall'interfaccia (nella figura indicata con «rete»).

L'avvento dei **minicomputer**, grandi quanto un congelatore da ristorante ma che dovevano il prefisso al "mini" confronto con le dimensioni del mainframe, nell'ottica di allora sembrò una rivoluzione concettuale, mentre oggi possiamo affermare che si trattò solo d'un passo tecnologico, in pratica del primo **downsizing** significativo della storia. Ogni riduzione delle dimensioni degli apparecchi porta a rendere più distribuita una o più delle sue componenti, e il mini facilitò la localizzazione delle risorse sia di calcolo che di rete soprattutto grazie a Digital, che s'inserì nel primo dei tanti errori di IBM e dal nulla creò un impero.

L'avvento dei mini, pur distribuendo i dati ed eliminando i problemi dovuti alla grande distanza tra unità elaborativa e terminale, in realtà non faceva altro che introdurre un livello nella struttura di rete, in pratica burocratizzando su tre livelli un processo che ne aveva sempre avuti due. Inoltre aveva ampliato un problema fino ad allora minimo, ovvero la corrispondenza tra dati locali (sui mini) e dati remoti (sul mainframe). Il mini comunque sortì numerosi effetti positivi: oltre alla diffusione della rete e al downsizing dell'hardware arrivò il downsizing del software. Iniziò la progettazione di sistemi operativi moderni e modulari, dei quali il sopravvissuto di quei tempi è Unix, un progetto che tra mille contraddizioni ha rappresentato il contraltare di questi ultimi vent'anni. Possiamo ben dire che Unix su mini iniziò quel procedimento di suddivisione funzionale dei compiti del software, introducendo dapprima una struttura e poi una chiara suddivisione dei compiti. Se all'inizio questo processo riguardò il solo sistema operativo, poi si estese anche al software applicativo e quindi alle interfacce di rete, e per tutte queste componenti si applicò il concetto di **layering** o stratificazione. Il layering consiste nel suddividere il compito in livelli di diversa profondità concettuale ma allo stesso livello filosofico, ognuno dei quali rende disponibili un certo numero di servizi al livello superiore. Nell'hardware ad esempio si può parlare d'un primo livello CPU, che comprende il microprocessore e la memoria, poi d'un livello di bus esterno (per video, disco ed altre periferiche), poi della parte interna del sistema operativo e quindi delle applicazioni: oggi il software applicativo chiede risorse al solo sistema operativo senza aver bisogno di sapere qual è l'hardware. Un altro esempio è la rete, dove l'introduzione d'un gestore capace di indirizzare le singole specifiche (Ethernet, Token Ring, FDDI) permette di avere sopra di lui dei servizi comuni a tutti i sistemi. In questo modo si ottengono due risultati: da un lato un'assoluta pulizia ed omogeneità delle soluzioni software, dall'altro l'indipendenza dei

Microsoft, cinque strategie di rightsizing

La domanda non è se ci si debba muovere verso il client/server, poiché l'ovvia risposta è sì, bensì come realizzare questa transizione. Microsoft propone cinque strategie:
1) dare al mainframe un'interfaccia grafica;
2) scaricare parte dei dati su una workstation per prendere decisioni;
3) scaricare su un server dei dati da condividere su una rete di workstation;
4) spostare un'applicazione transazionale su un server locale;
5) spostare l'intero sistema transazionale su una rete di server e workstation.

La prima di queste strategie lascia tutto inalterato, ma fornisce all'utente un **front-end** a maggiore produttività. Le scelte 2 e 3, pur lasciando il sistema operativo sul mainframe, forniscono ai *decision-makers* i dati in un formato manipolabile da tool locali, mentre le ultime due possibilità sostituiscono integralmente sezioni dell'attività con le nuove tecnologie client/server.

È ovvio che le maggiori attenzioni vanno poste proprio sui punti a maggior complessità, perché sviluppare un database c/s di grandi dimensioni può essere costoso quanto uno su mainframe, ed inoltre tale operazione richiede un tempo abbastanza lungo durante il quale i due hardware devono convivere; infine tuttora i mainframe gestiscono moli maggiori di dati, per cui al livello d'impresa globale devono essere mantenuti in attività.

Un modello applicativo

Finora lo sviluppo delle applicazioni ha seguito il cosiddetto COBOL blob, che portava ad un unico monolitico programma di grandi dimensioni che seguiva un'intera applicazione, se non addirittura più d'una. Dovendo sviluppare secondo il paradigma c/s, che suddivide in moduli le varie funzioni d'una applicazione, Microsoft propone un modello di riferimento basato su **API** e standard ed articolato su 4 strati:
- elaborazione su server;
- conversione dall'attività al server;
- mappatura dei dati sull'attività;
- disponibilità dei dati grezzi.

Partendo dal fondo, una volta resi disponibili i dati questi vengono aggregati secondo le esigenze del business, e poi convertiti in un formato intelligibile dall'applicazione software che risiede sul server. Questo percorso è quello della prima fase per l'input dei dati, mentre il successivo e conclusivo passo deve fornire all'utente i risultati dell'elaborazione, calcolati dal server e portati fino all'utente attraversando i quattro livelli in senso inverso a prima.

livelli superiori dallo sviluppo della tecnologia. La nascita d'un nuovo processore o bus, così come quella d'un nuovo standard di rete, non procura altra necessità che la realizzazione d'un piccolo software che la gestisca, mentre prima con il software che vedeva tutto l'hardware ad ogni novità bisognava riscrivere tutto.

Tornando all'excursus sull'hardware, nel tempo, il mini inglobò l'interfaccia grafica e con essa il concetto di singolo utente per tutta la potenza disponibile, diventando una workstation.

Nel frattempo si era giunti al concetto di microcomputer, un arnese dalla potenza limitata e assolutamente non collegabile in rete ma che stava tranquillamente su un tavolo. La sua storia è nota a tutti, quindi la saltiamo per giungere direttamente alle conclusioni, ▶

che vedono il personal diventare una workstation o un mini di fascia bassa.

Ma anche il software evolveva. Dicevamo prima che il sistema operativo comunica con l'hardware tramite una sua sezione inferiore, detta kernel (nucleo), che fornisce dei servizi lungamente ritenuti monolitici. Studi relativamente recenti hanno portato alla stratificazione anche di questo concetto, che ora è suddiviso in più moduli che eseguono compiti specializzati, quali possono essere la gestione delle periferiche, dei file, dei processi e delle comunicazioni, ciascuno affidato ad un modulo software talmente indipendente dagli altri che può risiedere su una macchina diversa in rete: ciascun modulo offre dei servizi di sistema, ed è quindi un **server**; chiunque ne richieda le prestazioni è un **client**. Se ne desume che il concetto di server o di client non è localizzato su un hardware specifico, ma dalla funzione che viene svolta. La delocalizzazione della potenza di calcolo e l'interconnessione in rete ha poi confinato a compiti ininfluenti quel lavoro differito che se svolto dal mainframe e distribuito su carta rendeva impossibile la verifica dei risultati, e sviliva la qualità delle informazioni necessarie a prendere decisioni. Viceversa avere sul proprio tavolo i risultati aggiornati permette di integrarli in uno spreadsheet e di avere sott'occhio gli elementi di scelta più utili.

L'affermazione del concetto di rete e di client/server, del quale parleremo diffusamente nel resto dell'articolo, ha portato ad un cambiamento della terminologia: il desktop che accede a risorse in rete si chiama client, e il mini che controlla solo parti specifiche del lavoro (la rete, il database, il file system) si chiama server. E siamo ai giorni nostri. Resta da porsi una domanda: che fine fa il mainframe? Beh, è semplice: se la sua potenza elaborativa non è più così importante, anche se esiste qualche progetto di reingegnerizzarli in tal senso, resta la capacità nelle comunicazioni e di gestione di database di enormi dimensioni. E poiché l'attuale tecnologia delle workstation e dei server non arriva bene ai gigabyte interrogati in parallelo, ecco che il mainframe fa il database server in ambienti **transazionali**. *Do the right size*, sembra dire il mercato.

Due esempi, email e RPC

Per giungere al client/server, un primo esempio molto interessante e storicamente valido è quello della posta elettronica o email, i cui concetti iniziali, per quanto abbiano seguito un'enorme evoluzione, sono tuttora validi. Scambiare messaggi tra utenti richiede un editor, un'interfaccia con il sistema, un meccanismo d'invio e consegna, e poi dal lato del ricevente un'altra copia dell'interfaccia e dell'editor. Lasciando perdere l'editor restano due moduli, uno utente e uno di sistema, che vengono chiamati User Agent e Mail Transfer Agent. È interessante notare che in questa semplice ed antica struttura sono già contenuti i concetti di layering e di client/server.

Inizialmente il meccanismo di **MTA** era semplice: ogni sistema aveva una directory per le caselle postali di ciascun utente, e il posto di raccolta dei messaggi era in un'altra directory. Lo **UA** depositava i messaggi nella seconda, e ciclicamente l'MTA leggeva il destinatario e depositava il messaggio nella sottodirectory del destinatario. Il trasferimento in rete era affidato a meccanismi primitivi, ovvero di indirizzo diretto (vedere l'articolo sul TCP/IP su questo stesso numero, nella rubrica su

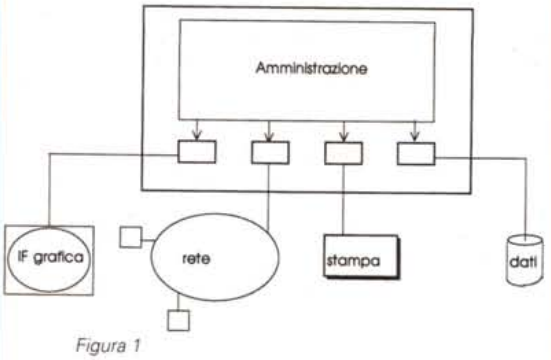


Figura 1

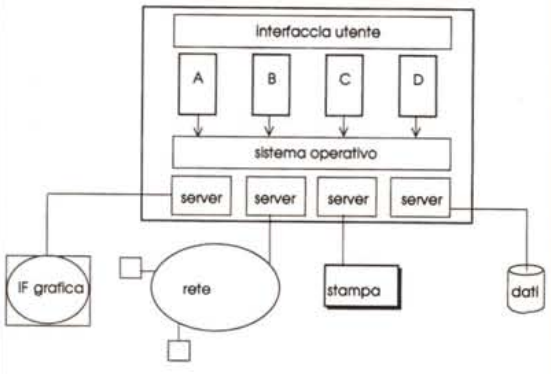


Figura 2

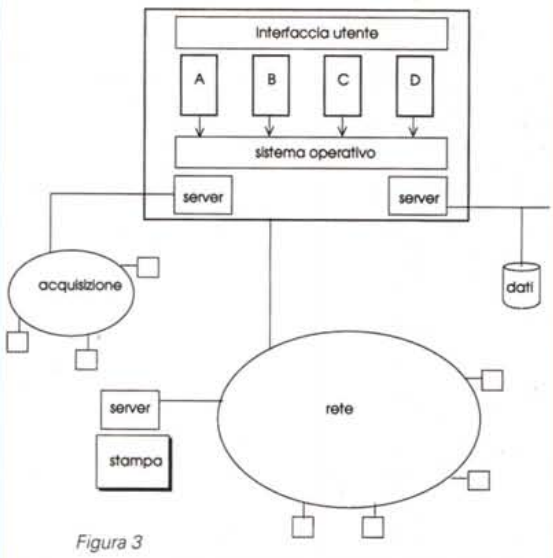
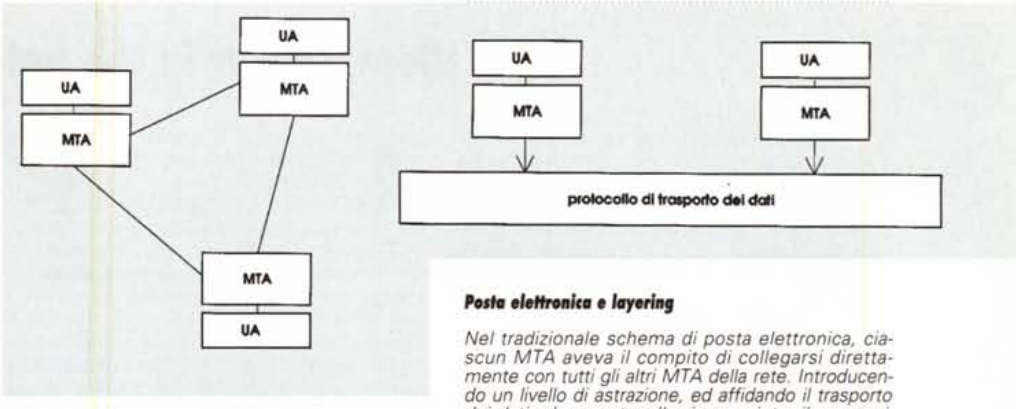


Figura 3

Dal monolitico al client/server

L'evoluzione dell'architettura informatica. In figura 1 c'è la tradizionale monoliticità del mainframe, con l'applicazione (ad es. l'amministrazione d'azienda) sotto forma di un unico, enorme programmone che indirizza tutto direttamente. Più che un sistema operativo c'è un certo numero di moduli software che accedono alle periferiche e offrono alcune utility.
In figura 2 il software è più strutturato: l'applicazione è stata scomposta in moduli visti sotto un'unica interfaccia utente, ed anche la gestione delle periferiche è stata integrata e completata fino a diventare un vero sistema operativo, addirittura implementato come client/server anche se tutto risiede sullo stesso hardware.
In figura 3, infine, il client/server è effettivo e distribuito. Una grande rete ha preso il sopravvento e gestisce le stampe con un apposito hardware, mentre il mainframe è visto come un elemento della rete che accede ai dati in qualità di file server e controlla una rete di acquisizione. Dal punto di vista filosofico nulla vieterebbe di dedicare sei server specifici ad uno dei moduli dell'applicazione.



Posta elettronica e layering

Nel tradizionale schema di posta elettronica, ciascun MTA aveva il compito di collegarsi direttamente con tutti gli altri MTA della rete. Introducendo un livello di astrazione, ed affidando il trasporto dei dati ad un protocollo riconosciuto, il meccanismo si semplifica sia dal punto di vista grafico che da quello concettuale. Questo meccanismo è detto layering.

Unix). Nel tempo si svilupparono dei sistemi più evoluti, fino ad arrivare al protocollo **SMTP**, Simple Mail Transfer Protocol, che definisce l'indirizzo di Internet.

A guardarlo bene, però, per considerarlo c/s al meccanismo di mail mancava qualcosa: lo UA non chiedeva esplicitamente un servizio, ma più che altro se lo aspettava, cioè sapeva che sarebbe stato svolto ma non quando. È questa una tipica conseguenza della filosofia di Unix, che sa come per ottenere il miglior risultato complessivo bisogna lasciare margini significativi a ciascuno dei singoli compiti. Il vero c/s, però, funziona in un altro modo: il client pretende un certo servizio, che verrà messo in coda agli altri che vengono eseguiti dallo stesso server, che è perennemente in ascolto. Talvolta il risultato può essere lo stesso, ma la richiesta viene accettata subito, ed il controllo del sistema è totale. In pratica quello che serve per essere classificati c/s è la possibilità di richiedere che un software posto su un'altra macchina inizi a lavorare per noi su nostra esplicita richiesta: noi effettuiamo quello che si dice una Remote Procedure Call o **RPC**.

L'approccio delle RPC prevede proprio che un cliente che richiede l'opera d'un server e ne ottiene una risposta è concettualmente equivalente ad un programma che chiama una procedura e ottiene un risultato: in entrambi i casi, infatti, il client inizia l'azione e aspetta l'arrivo dei dati, mentre il server non inizia mai una comunicazione. Nel caso tradizionale l'elaboratore che ospita i due moduli è lo stesso, mentre se la chiamata è remota, ovvero distante, si lavora su hardware diversi connessi in rete, ma la differenza non è nota al client.

In realtà non esiste una sola implementazione delle RPC, e le principali differenze sono due, l'effettiva trasparenza del meccanismo di rete e il livello d'implementazione. ▶

Sun, client/server in the enterprise

Il **rightsizing** è il processo di riconsiderare le risorse informatiche per vedere in che modo e a che prezzo seguono le necessità del business, onde sfruttarle al massimo mantenendo un vantaggio sulla concorrenza. Nella pratica consiste nel processo di migrazione dai sistemi attuali, comunque concepiti, ad ambienti **client/server** in rete, che offrono una più facile modellabilità della potenza sulle esigenze aziendali. Questa operazione ha due risvolti principali: ridurre i costi senza modificare il business oppure modificare il processo aziendale e quindi guadagnare terreno. Talvolta è possibile avere entrambi gli effetti.

Quale che sia la scelta, le domande alle quali rispondere sono quattro, tutte centrate sull'informazione:
- Dove si genera ed usa?
- Come e con che frequenza viene usata?
- Con che prontezza ci serve?
- Come viene distribuita attualmente?

Il punto focale dell'idea Sun è quindi il database. Tornando ai due diversi obiettivi, riduzione dei costi o miglioramento produttivo, partiamo dal primo caso. Solitamente il terreno migliore per intervenire è la riduzione dei costi di elaborazione e di quelli dello sviluppo delle applicazioni: ad esempio il solo costo mensile di una soluzione **DB2** su mainframe costa quasi quanto l'intera licenza d'un **RDBMS** sotto Unix, ed inoltre lo sviluppo sotto Unix costa circa un decimo dell'analogo processo su mini o mainframe.

Se invece l'obiettivo è di incrementare la produttività, ovvero guadagnare in quello che gli anglosassoni chiamano *competitive advantage*, bisogna ripensare l'intero processo produttivo, ridefinendo politiche, processi e tecnologie usati. Generalmente in questo caso il perno del discorso è il tempo perso nei vari passaggi, quindi la soluzione si trova nell'eseguire in tempo reale applicazioni critiche prima elaborate in batch. Tale soluzione inoltre semplifica la scelta dei manager, perché offre la disponibilità di dati in tempo utile e in formato manipolabile da strumenti desktop che loro possono usare direttamente.

Ma i vantaggi del rightsizing non si fermano qui. Una volta scelta la soluzione client/server, infatti, si scoprono alcuni ulteriori benefici. Il più importante è che l'organizzazione del sistema informativo risultante veste l'attività, dando all'azienda una struttura focalizzata e non più generalmente e pesantemente gerarchica, con efficienti comunicazioni tra i reparti e costi di gestione minori. Inoltre gli investimenti in informatica sono garantiti dall'interoperabilità con il sistema preesistente.

Finora abbiamo parlato delle esigenze del business, ma per l'aumento di efficienza richiesto esiste anche un punto di vista tecnologico. La tecnologia client/server, che frammenta l'elaborazione su componenti di vario formato collegati in rete, offre varie alternative per incrementare la potenza, a seconda della componente sulla quale si agirà. Tre i possibili livelli d'azione:
- client, aumentandone il numero;
- server, aumetando la potenza di quelli esistenti o aggiungendone altri;
- rete, allargando la banda disponibile di Lan o Wan.

Un modello filosofico

Dal punto di vista metodologico Sun propone un modello di rightsizing a sei fasi:
1) definizione del progetto;
2) applicazioni esistenti, e loro reingegnerizzazione;
3) il nuovo modello;
4) migrazione dal vecchio al nuovo;
5) implementazione del nuovo;
6) verifica e modifiche.

Parlando del primo problema la distinzione non è di scarsa importanza. Infatti rendere il meccanismo di rete implicito nel linguaggio, al più specificando un ulteriore parametro in fase di chiamata di sistema, equivale a spostare il problema sul compilatore: è lui che leggendo l'ulteriore parametro deve generare il codice della connessione, e ovviamente lo fa a livello di eseguibile, per cui qualsiasi problema si venga a creare prevede una profonda conoscenza del meccanismo di generazione del codice eseguibile oltre che un'esatta interpretazione del meccanismo di rete. Viceversa differenziando i nomi delle chiamate di sistema, quindi agendo a livello di codice sorgente, l'interpretazione dei problemi viene più semplice al prezzo d'una perdita dell'assoluta trasparenza. L'implementazione delle chiamate remote, che coinvolge molte altre considerazioni, è tuttora un punto di grande dibattito nella comunità internazionale, tanto che esistono perlomeno tre approcci di grande rilievo, quelli di Sun, ISO ed OSF rinforzato dall'adesione di Microsoft.

Una breve nota anche sul livello d'implementazione. Tradizionalmente l'RPC, in quanto basato su meccanismi di rete con risposta, viene considerato un servizio del transport. Nella terminologia Internet questo equivale ad essere un servizio del trasporto; in quella OSI sarebbe un servizio di livello 5 (essendo il trasporto un livello 4), ma poiché tale sistema tende a spostare tutte le applicazioni sul livello 7, l'implementazione ufficiale è a questo livello, il che – come sempre – aggiunge qualche ulteriore complicazione in cambio d'una visione più chiara e più orientata al futuro. MC

Glossario

API: application programming interface, definizione d'una metodologia completa di tool per scrivere del software per una specifica soluzione.

Batch, elaborazione a lotti: modalità di assegnazione dei lavori ad un sistema di grande capacità di calcolo ma scarsa capacità interattiva (mainframe e mini). I vari lavori erano suddivisi in blocchi detti lotti o *batches*. La caratteristica principale era che i risultati venivano resi disponibili dopo un lungo tempo, quindi in differita e non immediatamente o quasi come oggi.

Client: un processo che si connette chiedendo un servizio. Nell'hardware è una macchina di limitate capacità elaborative, magari con interfaccia utente amichevole, che chiede sulla rete le risorse che non ha (dischi, dati, procedure...).

Cooperative computing: fase di passaggio dall'elaborazione monolitica al client/server nella quale la logica del programma resta monolitica ma alcune risorse in rete vengono chieste ed indirizzate direttamente.

DB2: la tecnologia di database su mainframe di IBM.

Downsizing: riduzione delle dimensioni d'un sistema senza perdere in funzionalità.

Front-end: il primo nome dell'interfaccia utente.

Layering: stratificazione, ovvero suddivisione d'un problema in più livelli con la regola che ciascuno chieda servizi solo a quello immediatamente inferiore e fornisca altri servizi a quello immediatamente superiore.

Mainframe: unità centrale di grandi dimensioni, oggi capace di elaborazioni di media entità ma di elevata capacità di gestione di archivi di grandi dimensioni.

Modalità differita: vedi batch.

Remoto: dall'inglese *remote*, che come in italiano vuol dire distante, ma viene usato più frequentemente, e vuol dire elaborato su un'altra macchina.

Rightsizing: procedimento che dimensionando le risorse informatiche sulle necessità del business dà a ogni necessità le giuste capacità.

RPC: remote procedure calls, chiamate a procedure remote, ovvero localizzate non sulla macchina chiamante ma su un'altra in rete.

Server: processo che attende chiamate per svolgere il suo compito. Nell'hardware, un server è una macchina dedicata ad una specifica attività (ad esempio il database) e dimensionata su questa esigenza.

Transazione: richiesta di dati in tempo reale. Il tipico esempio è l'emissione dei biglietti aerei, che coinvolgono migliaia di chiamate quasi contemporanee.

WAN: wide area network, in italiano rete geografica, ovvero connessione tra sistemi distanti anche migliaia di chilometri.

Leo Sorge è raggiungibile tramite MC-link alla casella MC6750 e tramite Internet all'indirizzo mc6750@mclink.it

Da Capo

ActionNote 4000 Lit. 3.950.000*

E' nato il subnotebook che vi permette di fare a meno del computer da tavolo, e anche di quello sotto il tavolo: uno strumento potente e preciso in uno chassis elegantemente grigio scuro e piacevolmente piccolo e leggero.

Pensate, un 486 che si porta veramente nella valigetta, senza ingombrarla tutta, e con tutta la memoria RAM per lavorare sempre tranquillamente anche con le applicazioni "pesanti" di chi ha davvero un'azienda in mano (8 MB).

C'è un pratico e capace Hard Disk rimuovibile da 120 MB in cui tenere tutti i programmi e tutti i dati che vi servono, e c'è un display retroilluminato VGA con ben 64 toni di grigio per lavorare bene anche nelle applicazioni grafiche.

Se poi volete trasformare ActionNote 4000 in un desktop senza compromessi, oltre all'attacco per il monitor a colori trovate anche la porta per la tastiera esterna, una porta parallela, una seriale e un utilissimo slot PCMCIA type II, che può essere utilizzato per aggiungere memoria di massa, altra RAM, per mettervi in rete o per un modem/fax che vi mette in contatto diretto in ogni momento con il mondo intero.

Le dimensioni sono 25x19 (x3,8) cm, ed il peso è un ottimo 1.75 Kg con Hard Disk e batterie, ricaricabili NiCd, che assicurano circa 3 ore di autonomia. L'alimentatore di rete è fornito di serie, come il sistema operativo (DOS versione 6.0), il disk drive esterno per floppy da 1.44 MB, la elegante borsa per il trasporto e Windows 3.1, forse il più diffuso ambiente virtuale di lavoro al mondo.

Particolarmente utili la trackball integrata (niente scomodi cavi, niente pezzi aggiunti o sporgenti) e le funzioni di SUSPEND e RESUME, che permettono di riprendere il lavoro dal punto esatto in cui l'avete interrotto mettendo la macchina in standby.

Se questa incredibile novità vi interessa, è meglio prenotarlo in fretta (la tiratura è limitata); se poi volete saperne di più, guardate qui sotto.

EPSON
Una precisa scelta.

Se vi interessa sapere dove acquistare i prodotti Epson, chiamate il numero verde

167-801101

se invece volete maggiori informazioni, compilate e spedite il coupon qui accanto a: Epson Italia S.p.A. v.le F.lli Casiraghi 427 - 20099 Sesto S. Giovanni (MI) Fax 02/2440750

☐ Vorrei saperne di più sull' Epson ActioNote 4000.
☐ Inviatemi anche in omaggio il catalogo aggiornato dei computer Epson.

Nome _____
Cognome _____
Società _____
Via _____

Questo annuncio è stato creato con prodotti Epson®. * Prezzo IVA esclusa.