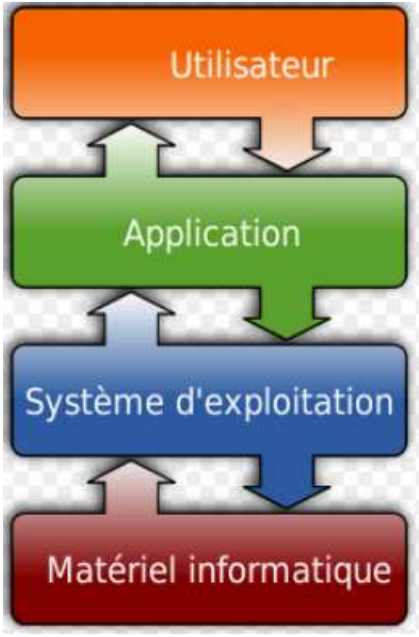
LES SYSTÈMES D'EXPLOITATION



par Simon Plouffe
novembre 2010

Introduction

1) Les machines Turing-complètes, ENIAC
2) SLOC, pyramides et gros systèmes
3) Mac, Windows et le GUI
4) La forêt UNIX, cygwin
5) Les SE modernes
6) IBM, histoire des mainframes
7) La virtualisation, milliards de transistors
8) Les systèmes embarqués, voiture, GPS, Ipod
9) Les gros systèmes, le top 500 et le calcul distribué

Conclusion

Introduction

On reviendra fréquemment à ce diagramme qui explique bien en image. On dira OS plutôt que S.E. pour des raisons pratiques.

On s'intéresse à la case bleue précisément. Elle est séparée dans le diagramme, en pratique l'histoire montre que le matériel influence et détermine comment la case bleue est écrite, qui influe sur la case verte et qui même à la limite influence la case orange.

Un programmeur de COBOL qui utilise une machine centrale IBM (mainframe) n'a pas la même tête qu'un programmeur JAVA. Pourquoi ?, c'est une question de génération.

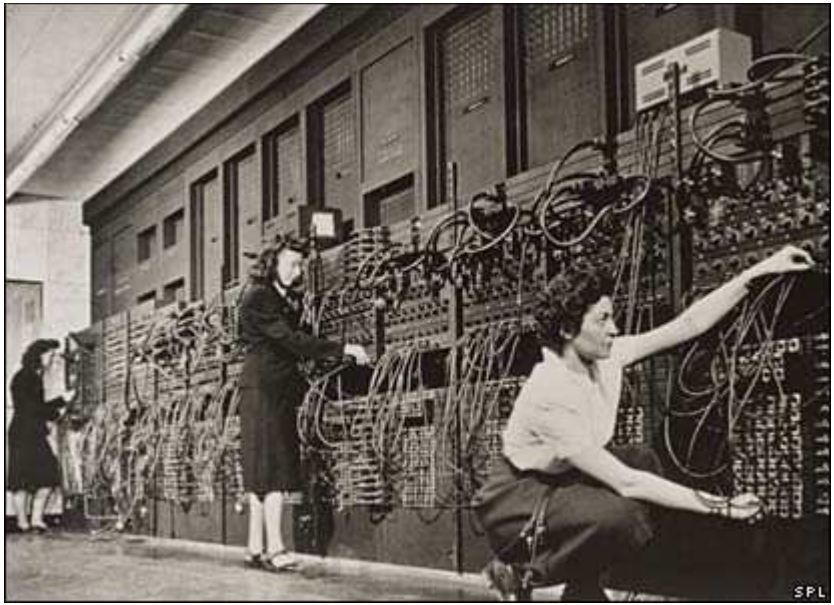
Nous essayerons de couvrir tous les aspects autour de l'OS.

Le GUI (graphic user interface), le wysiwyg (what you see is what you get). Le fameux code 45.

Les termes en anglais seront employés la plupart du temps.

Le diagramme est la situation actuelle, il y eu une évolution assez spectaculaire depuis 1946.

1946 est la date ou la première machine Turing-Complète a existée, c'était l'ENIAC (on y reviendra).

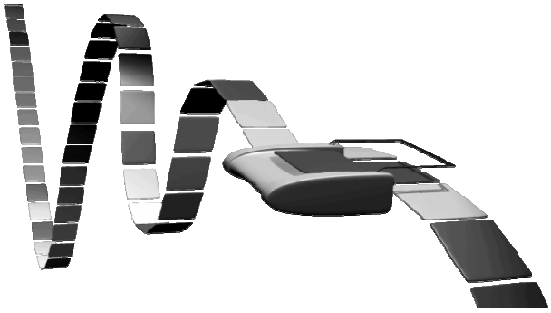


Une machine est Turing-complète si elle a le *pouvoir* des machines de Turing. On imagine ici un ruban infini avec des cases ayant une mémoire suffisante et qui ait la possibilité d'être récursive.

Le ruban est lu par une tête de lecture qui prend une décision, on a besoin d'un registre d'état, qui peut décider d'aller à gauche ou à droite.

C'est minimal mais un certain Church (Alonzo) a montré que même minimale, elle peut se débrouiller avec n'importe quel problème en autant qu'on puisse écrire les instructions sur le ruban.

En résumé, on peut y construire des tables de décision. Tables de décision = tables de vérité en logique avec des V. F, 0 ou 1. La plupart des langages modernes sont Turing-complets, pour l'être au sens générique il faut la possibilité de permettre la récursivité. Le SQL est devenu Turing-complet en 1999.



Le ruban de la machine

Il existe des modèles en LEGO ou en miroirs, certains en ont construit une avec un vrai ruban, un feutre qui écrit 0 ou 1 et une gomme à effacer.

Mais l'ENIAC, la machine ou le S.E. étaient pris l'un dans l'autre, la machine était inutilisable sans les 6 femmes programmeuses et pionnières n'avaient pas été là pour les programmer. Elles venaient du bureau de calcul de la Moore School. Elles étaient attitrées au calcul de trajectoire des obus, calculer des tables de tir avec des calculatrices de

bureau électromécaniques. C'est suite à ce besoin que les créateurs ont construit l'ENIAC : pour pouvoir calculer plus vite les tables de tir. C'était le premier ordinateur qui pouvait calculer l'endroit ou l'obus allait tomber avant que celui-ci n'arrive sur la cible.

Voici une partie du S.E. de l'ENIAC :

La plupart étaient diplômées en mathématiques.

ENIAC

8 pieds x 3 pieds x 100 pieds ou 67 mètres carrés sur le plancher, 66 mètres cubes, 30 tonnes, 17468 tubes, 5 millions de soudures.
100000 additions par seconde (5000 x 20), 10 chiffres de précision, 357 multiplications ou 38 divisions par seconde.
En 1954, ils ont réussi à le laisser allumé pendant 116 heures sans panne, un record.

Pour ceux d'entres-vous qui avez déjà reçu une carte de vœu qui joue de la musique : le petit chip à l'intérieur de la carte est ≈ 1 Eniac. (référence : le site de slashdot).

Mais le S.E. était en fait, du câblage qu'il fallait refaire à chaque nouveau programme. Il était général et 'general purpose' on pouvait le programmer selon le besoin.

Ils en ont profité en 1949 pour calculer E et Pi à 2037 et quelques décimales en 70 heures. On peut faire 1048576 décimales en 2,03 secondes maintenant.

Le langage était un langage machine, pas du langage machine comme l'assembleur mais 'un' langage machine. Et sans l'aide des 6 programmeuses pionnières et dévouées, elles avaient des doctorats en mathématiques pour la plupart, cet ordinateur était inutilisable. Le S.E. était ces personnes triées sur le volet pouvant y avoir accès et comprendre le problème. Elles recevaient régulièrement des requêtes de mathématiciens et physiciens pour la mise en place d'un calcul particulier. C'était la première qui était électronique et basée sur le modèle de Von Neumann.

L'ENIAC a fonctionné 9 ans, ensuite arriva l'EDSAC, MANIAC, UNIVAC, …

Et quelqu'une a eu la bonne idée de créer un langage : le FLOW-MATIC permettant d'écrire en anglais courant ou presque les opérations de programmation machine nécessaires. Elle se disait que ces gens spécialisés n'étaient ou ne seraient pas toujours disponible.

Plus tard, elle participa à l'élaboration du COBOL qui est né peu après.

Et pour coder le langage machine concisément, elle en a profité pour créer le premier compilateur pour ces langages. Elle en profita pour écrire une brique de 500 pages permettant d'utiliser ces machines en 'AC'.

Cette pionnière en fait a jeté les bases du concept (qui n'existait pas à l'époque) du premier système d'exploitation.

En fait ; et c'est très important : *L'évolution du langage informatique a permis de prendre le contrôle de la machine. Il manquait encore quelques morceaux qui sont nés après (les GLUE langages), on y reviendra. C'est cette évolution qui est le début des S.E.*

Le mystère de Grace Hopper (un jeu de mot avec grasshopper : criquet en anglais).

Grace Murray Hopper (1906-1992).

Elle avait un doctorat en mathématiques.

Elle a été co-inventrice du langage COBOL.

Elle a écrit le premier compilateur en 1951.

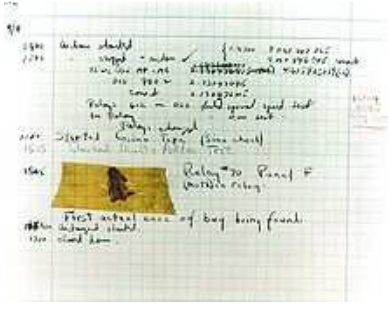
Elle a travaillé pour Digital jusqu'en 1985.

Elle était amiral (marine US).

Elle a un destroyer à son nom, le USS Hopper.

Elle a 8 médailles dont l'une est de la maison blanche qui porte son nom.

C'est elle qui a trouvé le premier bogue en 1947.



Le papier avec le papillon scotché est maintenant au musée Smithsonian. Quand elle est partie à la retraite, ils ont déployé un porte-avion et le USS Hopper pour lui remettre une dernière médaille.

En fait le COBOL est l'un des langages les plus utilisés dans le monde.

Selon une étude, il y aurait 200 milliards de lignes de COBOL en opération en ce moment et 5 milliards qui s'ajoutent chaque année.

Ce sont les banques, assurances, EDF, pétrole, la grosse industrie qui utilise le plus ce langage. (1 page de COBOL (80 x 24) = environ 1 ligne d'un langage 4GL). Un langage 4GL, comme le FOCUS, SQL, ABAP-4, presque le RPG.

En gros, ils font les payes avec ça presque partout, certains utilisent PeopleSoft par exemple mais il y a quand même des programmes en COBOL à l'interne…

Et l’axiome #1 d’une banque est que si ça marche, on ne change rien, sécurité des petits terminaux pour les cartes bleues ; seulement 40 bits de sécurité.

COBOL date de 1959 et régulièrement on entend dans les milieux informatiques que le langage est mort de sa belle mort et que tout ça sera fini dans 5 ans…

COBOL fonctionne sur des mainframes dont l’accès et la sécurité sont en béton, il est assez difficile de hacker un mainframe, c’est considéré impossible. Certains programmes de paye (comme à la TG) ont 30 ans. J’ai moi-même travaillé à un système de paye (150000 employés des hôpitaux au Québec, la moitié du personnel), les programmes avaient + de 30 ans et étaient couverts de ‘patchs’, une horreur…, des sorties en XML et postscript programmées en COBOL. Authentique !. D’ailleurs IBM met au défi n’importe qui de hacker un mainframe, authentique aussi.

Mais c’est quoi cette folie de 200 milliards de lignes de code ?

En fait on calcule la complexité d’un système avec le SLOC (source lines of code). C’est le nombre de lignes de code : C’est bancal comme mesure : OUI.

1 ligne = 30 minutes de travail pour un programmeur.

4000 lignes de code = 1 année personne. Une personne travaille 2000 heures par an mais ça varie selon les pays (1650 et quelques aux pays-bas). 2300 dans d’autres pays.

L’ensemble des programmes COBOL représente 50 millions d’années-personne.

Bel exemple ici mais en réalité c’est bien pire que ça…

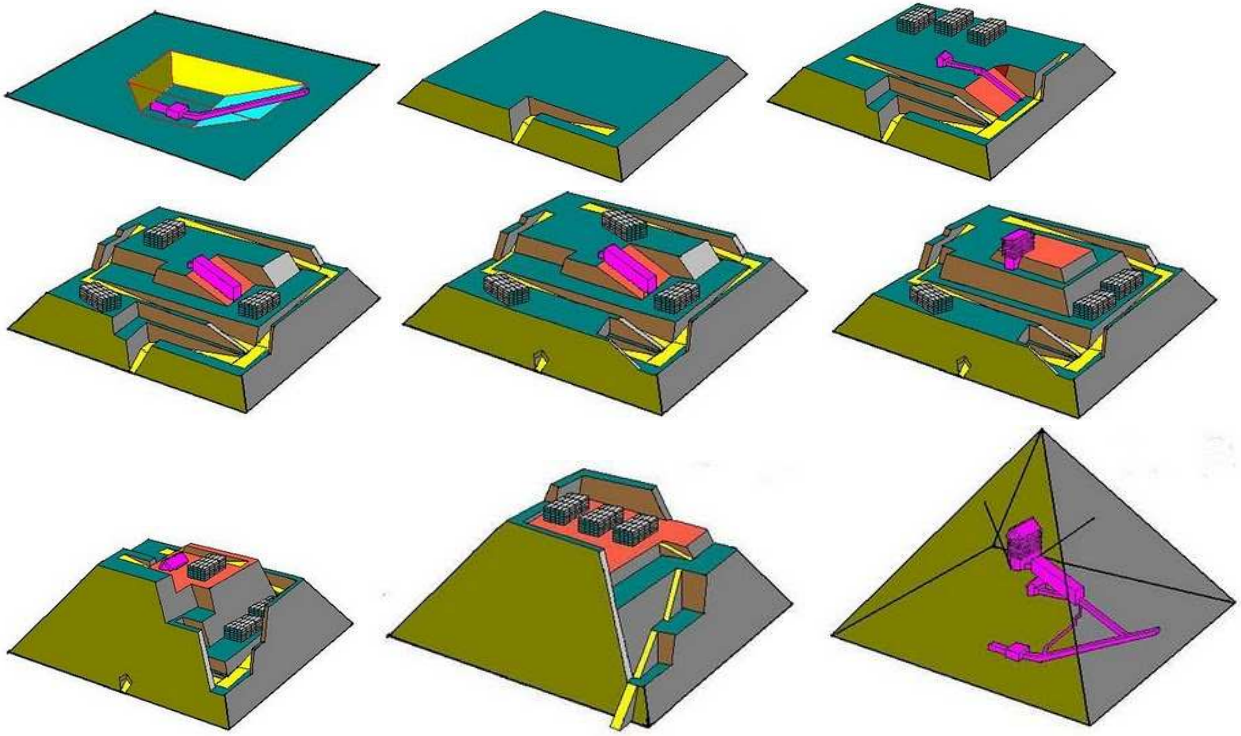
C	COBOL

`#include <stdio.h>`
 	
`int main(void) {`
 	
`  printf("Hello World");`
`  return 0;`
`}`
	000100 IDENTIFICATION DIVISION.
	000200 PROGRAM-ID. HELLOWORLD.
	000300
	000400*
	000500 ENVIRONMENT DIVISION.
	000600 CONFIGURATION SECTION.
	000700 SOURCE-COMPUTER. RM-COBOL.
	000800 OBJECT-COMPUTER. RM-COBOL.
	000900
	001000 DATA DIVISION.
	001100 FILE SECTION.
	001200
	100000 PROCEDURE DIVISION.
	100100
	100200 MAIN-LOGIC SECTION.
	100300 BEGIN.
	100400 DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
	100500 DISPLAY "Hello world!" LINE 15 POSITION 10.
	100600 STOP RUN.
	100700 MAIN-LOGIC-EXIT.
	100800 EXIT.

Lines of code: 5 (excluding whitespace)	Lines of code: 17 (excluding whitespace)

Non, ce n’est pas très élégant comme langage mais RPG est bien pire.

En fait, les personnes-années peuvent être comparées à une construction comme les pyramides de Khéops, 400 000 p.-a. = 20000 esclaves x 20 ans.



On compte la taille en lignes, 1 ligne (selon mon calcul) = 45 caractères environ.

Voici une table pour LINUX

Operating System	SLOC (Million)
Debian 2.2	55-59[2][3]
Debian 3.0	104[3]
Debian 3.1	215[3]
Debian 4.0	283[3]
Debian 5.0	324[3]
OpenSolaris	9.7
FreeBSD	8.8
Mac OS X 10.4	86[4]
Linux kernel 2.6.0	5.2
Linux kernel 2.6.29	11.0
Linux kernel 2.6.32	12.6[5]

Donc, le debian 5.0 est equivalent à 81 000 ans de travail pour 1 personne.
Cygwin (un faux-vrai unix) = 36000 années, Windows 7 (estimation) = 100 millions de lignes = 25000 ans.

…Sachant la taille en gigas nous donne la quantité en personne-année équivalent. 1 programmeur(euse) peut produire 2 lignes par heure de travail (déboguée bien sûr). Ce sont des standards. On reparlera de ça avec les 'bloatware' comme Acrobat 9, Itunes, NERO 9.0, ; Bloatware = obésiciel ou boufficiel, le pire de tous est Norton.

J'ai toujours dit que :

Si Unix est un système d'exploitation
alors une pile de brique est une maison.

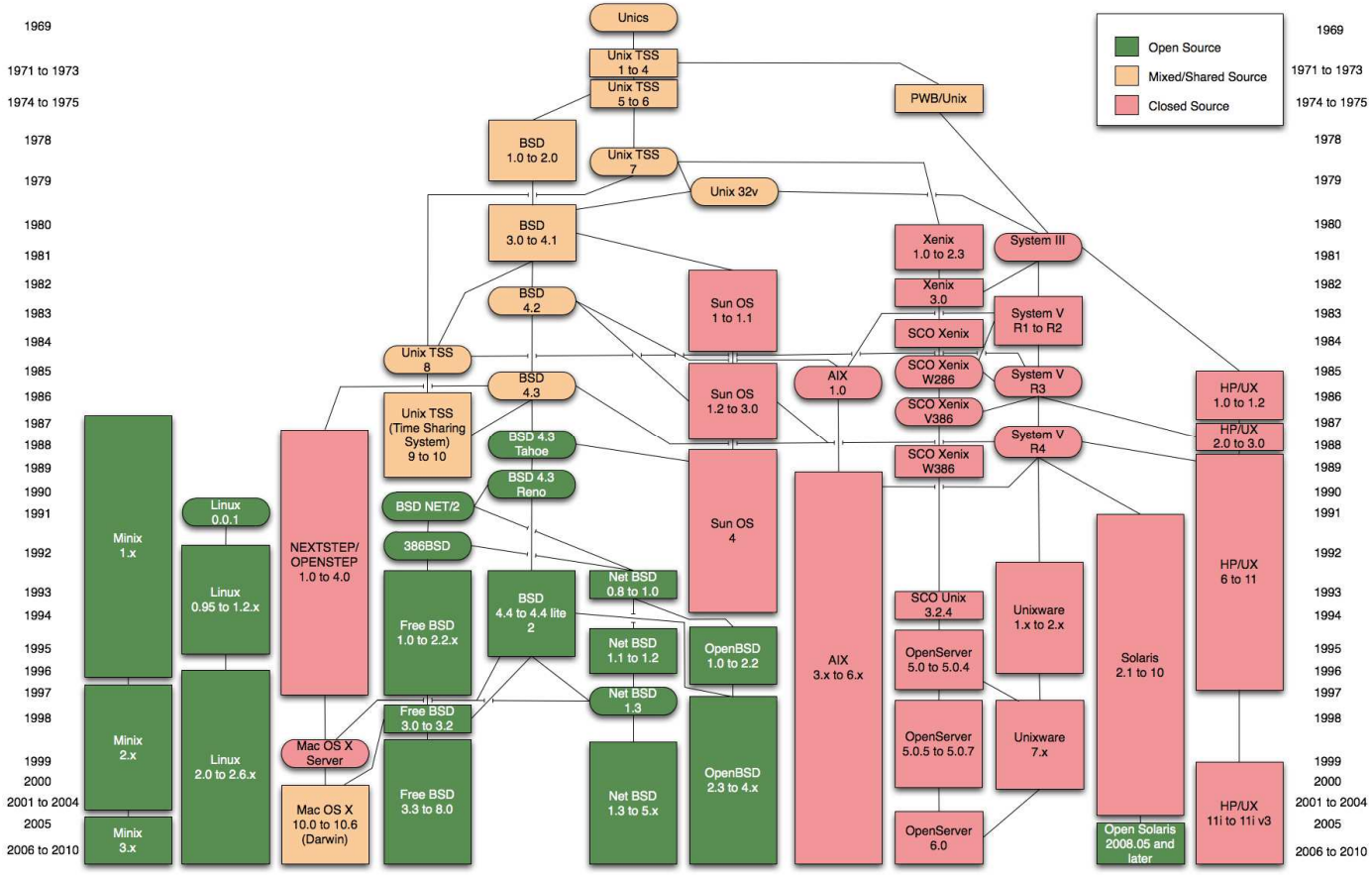
mais bon j'ai revu mon proverbe à ce sujet depuis, Ubuntu est pas mal… C'était à l'époque ou Unix n'était travaillable qu'en mode ligne de commande.

Il y a une analogie entre ce que fait un programme COBOL et EXCEL. On prend 2 ou 3 tables de valeurs dans une même feuille et on effectue un calcul en déplaçant le curseur 1 case à la fois, pour parcourir toutes les cases et préparer une 4ème table. Les va-et-vient du curseur en absolu et relatif donne une bonne idée d'un programme COBOL, l'image est assez exactement la même, un beau spaghetti.

La genèse d'UNIX

En 1969, Ken Thomson décide de créer un système sur un PDP-7.
Il le crée écrit en langage d'assembleur.
2 choses essentielles au niveau conceptuel se produisent ;
Ken et Dennis s'aperçoivent que le langage Assembler est lourd et largement insuffisant, ils tentent alors de l'écrire en langage TMG d'abord et BCPL ensuite.
BCPL est très puissant mais encore insuffisant.
Ils vont au-delà de BCPL et créent le langage C, après B c'est C….
Le langage C en fait est assez proche des commandes de base du système lui-même, la simplicité étant de mise. (question : pourquoi a-t-on nommé le langage C++ ?).
Ils créent donc, les commandes awk, sed, fgrep, … (on y reviendra).

Awk (de Aho, Weinberger et Kernigham) est un langage en soi, on peut gérer une base de données avec awk. La plupart des commandes Unix ont 3-4 lettres. Des livres existent sur awk, très intéressants.
Le 2ème jalon conceptuel est que les 'devices' sont des fichiers, tout est un fichier quitte à avoir des permissions et des statuts particuliers.
En fait, le C et l'Unix (le système) sont étroitement liés. La naissance d'Unix est la naissance du C.


Open Source
Mixed/Shared Source
Closed Source
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2010
Unics
Unix TSS 1 to 4
Unix TSS 5 to 6
Unix TSS 7
PWB/Unix
BSD 1.0 to 2.0
Unix 32v
BSD 3.0 to 4.1
BSD 4.2
Sun OS 1 to 1.1
Xenix 1.0 to 2.3
Xenix 3.0
SCO Xenix
System III
System V R1 to R2
Unix TSS 8
BSD 4.3
Sun OS 1.2 to 3.0
AIX 1.0
SCO Xenix W286
SCO Xenix V386
SCO Xenix W386
System V R3
System V R4
HP/UX 1.0 to 1.2
HP/UX 2.0 to 3.0
Unix TSS (Time Sharing System) 9 to 10
BSD 4.3 Tahoe
BSD 4.3 Reno
BSD NET/2
386BSD
Sun OS 4
AIX 3.x to 6.x
HP/UX 6 to 11
Minix 1.x
Linux 0.0.1
Linux 0.95 to 1.2.x
NEXTSTEP/ OPENSTEP 1.0 to 4.0
Free BSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite 2
Net BSD 0.8 to 1.0
Net BSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
SCO Unix 3.2.4
OpenServer 5.0 to 5.0.4
Unixware 1.x to 2.x
Solaris 2.1 to 10
Minix 2.x
Linux 2.0 to 2.6.x
Free BSD 3.0 to 3.2
Net BSD 1.3
OpenBSD 2.3 to 4.x
OpenServer 5.0.5 to 5.0.7
Unixware 7.x
Mac OS X Server
Mac OS X 10.0 to 10.6 (Darwin)
Free BSD 3.3 to 8.0
Net BSD 1.3 to 5.x
OpenServer 6.0
Open Solaris 2008.05 and later
HP/UX 11i to 11i v3
Minix 3.x

Pour des raisons légales, Bell Labs (ATT) à l'époque ne pouvait fournir que des équipements téléphoniques ou télégraphiques. Unix étant un logiciel, il a été décidé en 1975 de donner le code source aux universités à des fins éducatives ou moyennant une somme très modique (Berkeley, devenu BSD après).

Étant ouvert par nature, les programmeurs s'en sont donnés à cœur joie pour développer toutes les saveurs qu'on connait.

En principe, Unix ou Linux sont parfaits mais il manquait 1 chose : Depuis 1969, le CLI était viable, l'éditeur ed ou vi ou emacs (sans entrer dans les guerres de religion), et arriva l'ère de l'Apple IIe, le macintosh, le PC d'IBM, le wysiwyg et le GUI qui allait changer la donne.

La souris : Englebart 1965, XeroX à Palo Alto ont même vendu leur système à Jobs et Wozniak n'y voyant pas l'intérêt.

Les commandes awk, sed grep cat, vi, au nombre d'environ 160 mais une installation CYGWIN compte 2937 exécutables, la quantité varie passablement d'une installation Unix à une autre (voir AIX d'IBM ou le Unix d'Apple).

Voici en gros en quoi consiste le système Unix.

cat,cd,chmod,chown,chgrp,cksum,cmp,cp,dd,du,df,fsck,fuser,ln,ls,lsat,tr,lsof,mkdir,mount,mv,pwd,rm,rmdir,split,touch,umaskat,chroot,cron,exit,kill,killall,nice,pgrep,pidof,pkill,ps,pstree,sleep,time,top,wait,env,finger,id,logname,mesg,passwd,su,sudo,uptime,wall,who,whoami,write,awk,comm,csplit,cut,ed,ex,fmt,head,iconv,join,less,more,paste,sed,sort,strings,talk,tail,tr,uniq,vi,wc,xargs,alias,basename,dirname,echo,expr,false,printf,test,true,unset,inetd,host,ifconfig,netstat,nslookup,ping,rlogin,netcat,traceroute,find,grep,locate,whereis,which,apropos,banner,bc,cal,clear,date,dd,file,help,history,info,lp,lpr,man,pax,size,tee,tput,type,uname,whatis,yes.

Tout ce qui roule sous Unix est une savante combinaison de ces programmes, collés ensemble à l'aide de scripts plus ou moins automatisés.

Qu'est-ce qui est jaune, très petit et extrêmement dangereux ?

--Un canari qui a le mot de passe du root.

Les différentes saveurs d'Unix (voir diagramme) sont telles qu'à cause des saveurs locales du shell (bash, ksh, sh,…) il y a se sérieux problèmes de portabilité. On ne peut pas changer un système pour un autre en important simplement les scripts et utilitaires d'un seul coup (pas une bonne idée). Pour le AIX d'IBM et le unix d'apple c'est voulu…à mon avis.

Collés ensemble : c'est le mot, ce qui permet au système de fonctionner sont les programmes GLUE ou le *GLUE langage*. Le système est carrément dysfonctionnel s'il n'y a pas ces programmes.
Justement, la simplicité d'Unix a l'époque était que le système lui-même était presqu'entièrement fait de ces programmes courts mais très efficaces.
Beaucoup plus tard sont apparus des utilitaires d'abord qui sont devenus de véritables langages comme le PERL, Python. PERL veut dire (si on lit le man jusqu'à la fin) : Pathological Eclectic Rubbish Lister ou ramasseur d'ordures pathologique, scripteur de m…. pathologique.
Sans blagues, le PERL est un langage très puissant mais un peu difficile à déchiffrer.
Au final, Unix fonctionne avec ces utilitaires et scripts.
--Pendant des années mon inverseur à Montréal, Vancouver a fonctionné avec Unix et quelques commandes, perl et look. Look

permet une recherche binaire dans un fichier trié. Look est une commande tirée du dictionnaire d'Unix mais qui existe aussi en version perl tout aussi efficace. Le nouveau site est
http ://www.plouffe.fr/simon/
(un peu de pub).
Le BASIC, apparu en 1963 par un prof voulant rendre accessible les ordinateurs du collète Dartmouth. Le programme se voulait général (all purpose). On a reproché longtemps au BASIC d'avoir des boucles en spaghettis et surtout des GOTO. Des thèses on été écrites sur l'effet nocif des boucles avec des GOTO, le langage PASCAL, ADA ont été créés dans ce but.

Mais le Basic s'est tellement répandu grâce à Microsoft, Apple, Atari, etc que le VBscript est apparu sous Windows comme un programme

GLUE malheureusement. Le langage est maintenant structuré mais une fois le mauvais pli des GOTO bien ancré…

Ici le BASIC dans la plupart des PC IBM, Apple et autres était en ROM. C'est Apple qui a poussé assez loin cette logique d'inclure les routines avancées directement sur la carte-mère. Quickdraw en est un bel exemple. Le OS d'Apple à l'époque était écrit en C, il le fut en Pascal au début (apple IIc). Les routines du Quickdraw étaient très avancées.

-Certains algorithmes de diffusion utilisés pour les images BITMAP comprenaient des sous-routines qui elles-mêmes faisaient appel à des algorithmes fractals. On sait comment Windows gère les BMP… : aucune compression.

C’est pour cette raison, qu’à l’époque il était impossible d’avoir l’équivalent du MacPaint sur PC, les images n’étaient carrément pas les mêmes, justement à cause des routines ultra-spécialisées du Quickdraw de la ROM du Macintosh. Rien qu’en regardant une image bitmap on savait si ça venait d’un PC ou d’un MAC, à l’époque des guerres de religion PC-MAC. Le NeXT qui a suivi était dans le même esprit. L’OS était assez indissociable de la machine.

Une firme (Silicon Graphics) a même poussé bien plus loin en incluant carrément des circuits DSP très avancés directement sur la carte-mère, rendant impossible pour les autres d’avoir accès à des programmes d’imagerie. Ces différences sont maintenant obsolètes car tout est software. SGI a été vendu, revendu et acheté par CRAY (ou l’inverse), ils font maintenant dans le GROS sur Linux. Il y a eu longtemps une

différence assez notable entre la carte-mère d'un MAC et d'un PC, ce qui différenciait surtout était la taille de la ROM du MAC. C'est pour cette raison que le MacPaint n'avait que 22K. QuickDraw a demandé l'équivalent de 4 années-personne en développement (wikipedia fr).

Cette concurrence entre PC et MAC, SGI et les autres a forcé SGI à développer Open-GL justement à partir de la ROM des chips DSP, le logiciel est maintenant portatif et est utilisé un peu partout.

Ils ont voulu qu'il soit gratuit pour contrer surtout Microsoft avec son DirectX ou Direct3D (et les bibliothèques sous jacentes).

Curieusement, Google Earth a été développé en Open-GL mais fonctionne très mal sur un PC qui bien sûr a le Direct3D, en essayant les 2 (on peut choisir dans les options) on voit la différence et bien sûr les bogues…

Qui dit graphiques, dit consoles de jeu. On sait toute la bataille commerciale qu'il y a entre Xbox et les autres pour le monopole des stations de jeu… Open GL a été adopté par presque tout le monde sauf

les américains avec Microsoft (comme les gallons, pouces, …)

Un mot sur la souris, inventée en 1963 par Engelbart. Qui n'a pas ouvert la première fois une souris pour voir comment ça peut bien fonctionner(!).

Au début avec le macintosh, les mouvements de la souris comparés à ceux obtenus sur un IBM pc étant notablement différents. Le seuil d'accélération sous Windows a longtemps utilisé une routine bête qui rendait le calcul balistique très inconfortable sur un pc. Avec Windows XP, le calcul balistique est beaucoup mieux et similaire à celui du Mac ou d'autres. Le programme sous macintosh a longtemps été dans la ROM de la machine ce qui n'était pas le cas des PC.

Il faut comprendre que la résolution est de l'ordre de 1600 points par pouce et qu'une souris optique moderne éclaire la surface à un rythme de 1 000 000 de fois par seconde.

Pour les forts en math, la souris utilise un algorithme qui avait un net avantage au début sur les macs. La raison : le contrôle balistique demande qu'il y ait quelque chose de prévu pour la $2^{ème}$ dérivée de la vitesse.

Oui en effet, accélération uniforme = dérivée constante, accélération de l'accélération = dérivée seconde.

Justement, l'algorithme pour Windows utilisait semble t-il 1 seul palier que l'utilisateur pouvait changer, mais qui doublait à chaque pas ce qui rendait l'utilisateur hargneux. Encore aujourd'hui la différence entre les algorithmes sur mac OSX et PC Linux ou Windows en rend quelques uns encore furieux, mais on peut corriger le tir (c'est le mot) en changeant les paramètres.

Cette évolution très pointue sur la façon dont la souris se déplace n'est pas pour rien.

En fait : une bonne partie de l'effort de programmation dans un S.E. est lié à des détails vraiment insignifiants au niveau du GUI (graphical user interface).

Tous les S.E. qui ont été développés sur pc (mac, amiga, ibm pc, pc) depuis 1984 sont fortement imprégnés de GUI, ce qui est assez différent d'Unix qui est resté très longtemps attaché au CLI seul.

Les mainframes sont et seront encore longtemps liés au CLI, justement c'est le but de la campagne d'IBM, de recruter des nouveaux…

Mais pourquoi ?, c'est que l'utilisateur est extrêmement chatouilleux.

Prenons un exemple très simple : Une différence subtile mais qui n'en est pas moins importante entre les PC et les mac (OSX et Windows 7) est que :

-L'ombre de la souris et des fenêtres est plus diffusé sur le mac.

-Le pointeur (curseur) est plus rond sur mac.
-Le coin supérieur gauche ou droit est plus arrondi, diffusé sur mac, les coins supérieurs des fenêtres sur Windows 7 sont plus carrés.
-Mac utilise Quartz (le successeur de QuickDraw) + openGL, windows utilise Direct3D.

Une bonne façon de se rendre compte de ce détail ; essayez de trouver des thèmes sur internet qui changent l'aspect de votre pc en mac, je déconseille de faire ça.

Linux avec Ubuntu (Gnome) est similaire à Windows 7.

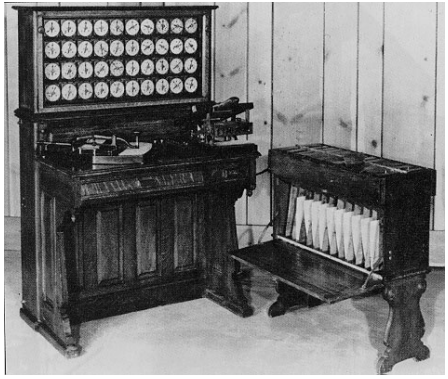
Mac OSX

Windows 7 avec les options d’affichage, c’est Windows NT déguisé en fait…

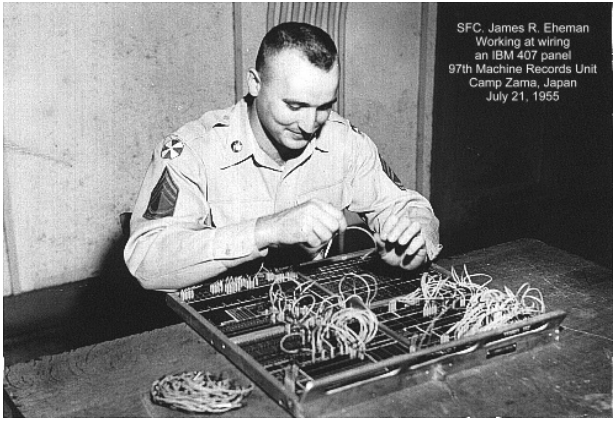
Les Mainframes (ordinateur central, mais personne ne dit ça).

On retourne à l'ENIAC, le besoin qui a créé cette machine en était un de calcul (tables balistiques, recherche, Von Neumann) mais il y a un autre besoin qui est de classer.
En fait une machine ça fait 2 choses : ça classe et ou ça calcule.

Justement, le classement : l'ancêtre de ce besoin a été créé lors du recensement américain en 1890, Hollerith a inventé la carte perforée (punch card).
Si on regarde de près en fait, IBM a été fondé en 1896, la compagnie TMC, tabulating machine corporation.



En fait c’est l’histoire du traitement de l’information.
En 1949, le IBM 407 Accounting machine était une tabulatrice qui faisait des rapports (150 lignes/minute), elle lisait un programme établi par un programmeur. (image du champion de 1955). Mais ces machines existaient depuis 1920 bien avant l’ENIAC. C’est après qu’on a eu l’idée de connecter les 2 : un ordinateur électronique et un mécanographe.



Tabulatrice : (wikipedia, fr)

Lisant les cartes venant en général de l'atelier de saisie (perfo/vérif), il en traitait le contenu en suivant un *programme matérialisé* par un tableau de connexion. Ce programme spécifiait les calculs à effectuer et la disposition des résultats sur l'imprimante de sortie. La tabulatrice pouvait également donner des ordres à une perforatrice connectée en sortie pour produire des cartes récapitulatives utilisées dans de nouveaux traitements. Donc, tri de données et rapport tabulé, en bout de ligne ce machin est presqu'un système d'exploitation en soi. Les organes de calcul sont des totalisateurs qui peuvent effectuer 150 additions par seconde Pour la soustraction, on additionne le complément à 9. Pour la multiplication, on utilise des tables de Pythagore. Les divisions se font par soustractions avec décalages successifs. ENIAC : 100000 additions/seconde…

La technologie était totalement électromécanique. En 1950 commence à apparaître la possibilité de connecter à la tabulatrice un calculateur électronique chargé d'effectuer les calculs complexes.

Quelques jalons : • 1890 : Premier recensement américain effectué selon des méthodes mécanographiques en fonction du brevet déposé par Herman Hollerith.
1896 : Hollerith fonde la Tabulating Machine Co, principale racine de ce qui deviendra plus tard (1924) IBM
1920 : l'ingénieur norvégien Fredrik Rosing Bull construit les premières machines mécanographiques européennes.
1920 : première tabulatrice avec impression (Hollerith)
1925 : première trieuse horizontale (Hollerith)
1928 : IBM lance la carte perforée à 80 colonnes à trous rectangulaires qui deviendra , après des années de procès sur les brevets, le standard de la

profession, et lance la même année la première tabulatrice à soustractions, ouvrant la voie aux applications comptables. Les trous carrés font des confettis carrés, une horreur dans les fêtes et anniversaires, on vous en reparlera bien des années après…
1930 : enregistrement d'alphanumérique sur les cartes 80 colonnes
1934 : la vitesse d'impression portée pour la première fois à 150 lignes/minute (Bull) grâce à l'invention du tambour d'impression, record qui tiendra 17 ans.

1940 : René Carmille, chef du service national des statistiques, crée un fichier national de recensement de la population pour lequel est inventé le numéro national qui deviendra le code INSEE. Sous couvert de ce travail officiel, il met sur pied clandestinement le fichier de mobilisation de l'armée d'armistice, qui sera utilisé pour appuyer le débarquement d'Afrique du Nord en 1942.

Cette idée de connecter les machines mécanographiques avec un ordinateur électronique a donné naissance à l'IBM system/370.

Le système MVS du 370 était multi :

MVS : *Multiple virtual storage*

Multiprogrammation , plusieurs utilisateurs ou travaux en même temps, séquentiellement mais pour l'échelle humaine = en même temps.

Multitraitement, 1 ordinateur peut gérer entre 2 à 16 processeurs avec MVS-SP ou MVS-XA.

Mémoire virtuelle, transparent pour l'utilisateur. Le système lui peut *swapper* la section de mémoire dormante sur disque et laisser les segments de code qui sont les exécutables en mémoire.

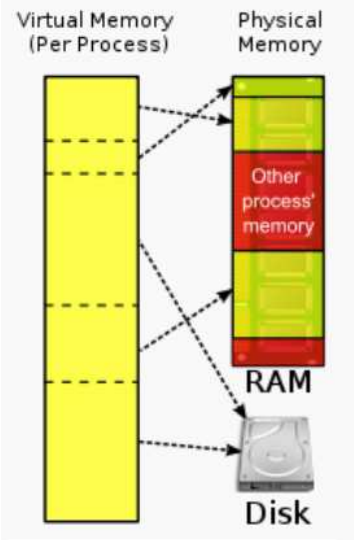


Schéma de principe de la mémoire virtuelle

MVS address spaces - global view

MVS (shared part of all address spaces)		
App 1	App 2	App 3
Shared virtual area (controlled by MVS)		

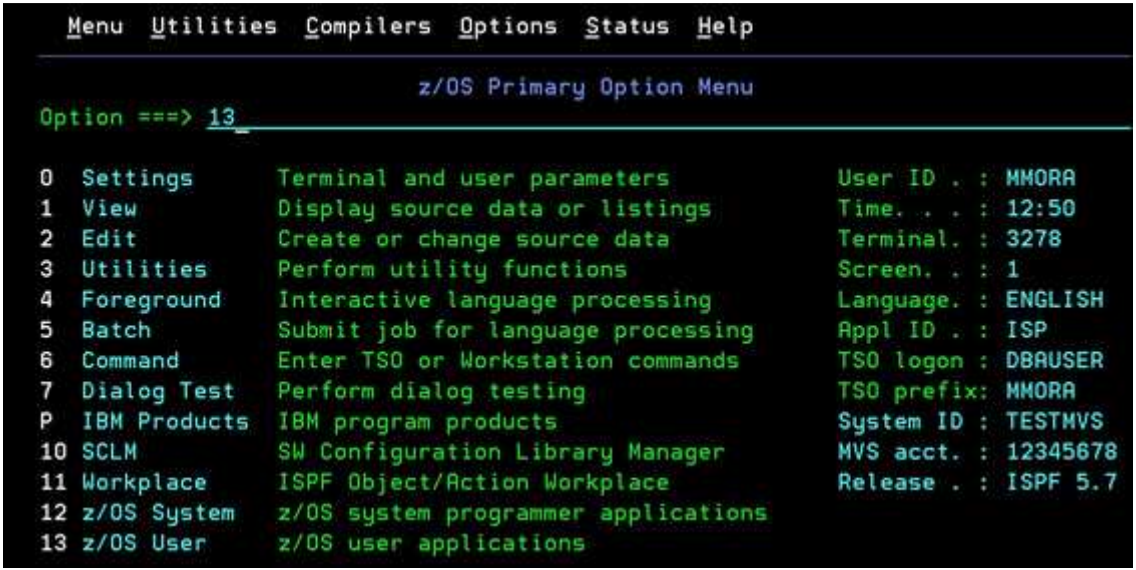
One application's view

MVS
App 1
Shared virtual area

Le système de fichier : 44 caractères maximum sous la forme de chaines séparées par des points , ex. SYSTEM.FILE08.DATA005 , en fait c'est un système hiérarchique comme sur les mac (system 7, 8, 9). L'utilisateur sur mac a l'impression d'avoir des répertoires et dossiers mais ce n'est qu'une longue chaine de points ou de séparateurs, (/) sur Windows.

RACF , *resource access control facility* n'a rien de facile, RACF contrôle tout et enregistre tout, la sécurité est basée sur les listes des répertoires accessibles. RACF agit aussi en identifiant des motifs sur les noms des fichiers et ou répertoires. Il y a 3 niveaux : utilisateur, groupe et ressource. Ça parait simple mais dites-vous que sur Windows avec l'accès réseau n'importe qui peut effacer des répertoires au complet sans laisser de traces et bien pire avec des scripts sur cygwin…

L’accès sur un mainframe ne se fait principalement que par le SDSF



Le menu est contrôlé par ISPF qui permet de choisir la couleur(!) du menu, les items de la liste et d’accéder en édition aux fichiers via XEDIT

ou ISPF. Le système 370 a été le premier à l'échelle industrielle à utiliser la VM, *virtual machine* .

Le CMS, conversational monitoring system, en fait qui venait des labos de Cambridge (nom original) et du labo de Grenoble, c'est à cette époque qu'a été conçu l'éditeur de texte XEDIT par Xavier de Lamberterie (d'où le X) et ressemblait à ceci : on voit ici une partie d'un JCL (*job control language*),le JCL est aussi appelé le Jean-Claude Language à Montréal. XEDIT permet d'éditer ou voir à peu près n'importe quoi y compris les exécutables compilés, d'éditer un fichier en hexadécimal ou en hexadécimal et compressé.

Ne pas confondre XEDIT de Xedit ou autres variantes comme X-edit. XEDIT faisait partie du CMS, CMS était un système renégat en parallèle avec MVS-TSO. IBM n'a jamais voulu en faire un vrai système pour des raisons de contrôle, on peut y faire un peu ce qu'on veut et surtout lancer des jobs en temps réel qui peuvent exploser en mémoire. TSO sur MVS quant à lui est beaucoup plus contrôlé puisque chaque *job* est lancée par un JCL ou un script qui lui alloue (*allocate*) des ressources de

disques ou de fichiers bien précis. L'interface avec le système de CMS était soi SDSF (le menu) ou en mode libre avec l'invite de commande directement. À l'époque, il était possible d'installer CMS directement sur un IBM PC à l'aide d'une carte spéciale de 3 pouces d'épaisseur (7,5 cm). L'équivalent de XEDIT sur TSO est ISPF, très similaire mais moins interactif. Quand un programme COBOL roule sur mainframe, l'allocation mémoire est très précise : les bons programmeurs qui déboguent un programme font souvent des 'dumps' et examinent le contenu de la mémoire directement (pas moi en tous les cas!).

On résume ici :
TSO est un interpréteur de commandes, un sous-système de MVS.
ISPF : éditeur et gestionnaire de dialogues qui peut exécuter des scripts CLIST ou REXX, ces 2 derniers étant des GLUE programs comme le JCL.

CMS un sous-système de MVS.
XEDIT : similaire à ISPF mais sous CMS.
Data set : est un fichier sur IBM qui a une certaine organisation sur disque. Ils sont soi RECFM, de largeur fixe par exemple 80 colonnes pour des programmes COBOL ou FORTRAN, les RECFM peut être fixes ou variables, on dit RECFM=FB ou VB. Ces fichiers résident sur les disques, appelés DASD (*direct access storage device*).
Les disques (DASD) sur mainframe sont organisés en *Volumes*, *cylinders* et *tracks* (pistes en français mais personne ne dit ça).

Table 78. Calculated Values for Several DASDs

SPOOL Device Type								
	3380	3380E	3380K	3390-1	3390-2	3390-3	9340-1	9340-2
Device Characteristics								
MBYTES/VOLUME	630	1260	1890	946	1892	2838	1003	1502
BYTES/TRACK	47476	47476	47476	56664	56664	56664	46456	46456
TRACKS/VOLUME	13275	26550	39825	16695	33390	50085	21600	32340
Values based on BUFSIZE=3992 and TGSIZE=30								

C’est certainement compliqué à comprendre pour un néophyte d’autant plus que : pour un volume donné, une mauvaise définition du RECFM d’un fichier occasionnera une perte d’espace qui peut faire planter une job batch très facilement. On donne ici les ‘*calculated values*’ et non pas les valeurs réelles, comme on dit le blocage (blocks) fait varier passablement l’espace disponible.

Ces considérations sont en fait complètement virtuelles sur les systèmes récents comme le Z/OS (successeur de MVS), puisque les

disques sont sur des SAN, NAS, ou nuages de disques, enfin sur réseau dans des disques RAID. La nomenclature a été préservée pour la compatibilité des programmes. L'organisation est LOGIQUE, pas physique.
Si jamais on vous donne l'occasion d'aller visiter une salle d'opération, une vraie salle informatique d'un vrai mainframe : allez-y.
Le système 370 a été le premier à proposer des machines virtuelles ; plusieurs systèmes d'exploitation sur une même machine.

Le besoin a été créé parce que la migration d'une machine vers une autre, par exemple à un système d'une version antérieure aurait normalement nécessité d'avoir 2 machines, donc prix élevé.

La solution a été trouvée par Cambridge et Grenoble d'avoir en fait un CP (*control program*). En fait le 370 était appelé VM/370.

Le programme permettant de piloter est appelé *hypervisor* et on distingue 2 types.
De type 1 ou natif ou *bare-metal*
De type 2 dit *hosted* ou invité, on a donc une 3ème couche logicielle, un système qui roule à l'intérieur d'un autre.
Sur les processeurs modernes comme Intel existe une option appelée VT qui permet une meilleure gestion du CPU et de la mémoire, sur AMD également. On peut installer VMware même si l'option n'est pas cochée dans le BIOS, par contre c'est moins rapide (64 bit impossible).
La différence entre le niveau 1 et 2 est par exemple sur un pc :
Niveau 1 : Installation d'Ubuntu via l'utilitaire wubi.exe. Il installe Ubuntu nativement mais en trichant sur le MBR ou le GRUB si vous préférez. En bootant le PC on a le choix de partir avec Windows ou

Ubuntu, la même chose existe sur mac avec le *bootcamp*. Un bon conseil, traficoter le MBR, le GRUB peut s'avérer désastreux si on fait une bévue. D'où la meilleure idée d'utiliser un hyperviseur de niveau 2. Voir : Tatouage des disques, traficotage d'Apple sur machines Intel, etc. Niveau 2 : avec VMware on peut avoir Windows qui roule et avoir une machine virtuelle comme Ubuntu, XP. On peut en principe installer Mac OSX mais il y a une triche au niveau BIOS qu'Apple se garde bien de révéler. Donc, on y arrive en se procurant une copie piratée du OSX d'Apple et avec VirtualBox. QEMU par exemple permet l'installation d'une gamme assez vaste d'OS créé par Fabrice Bellard (le calcul de pi).

On a parlé ici que des OS standards. On ne couvrira pas les OS avec micro-noyaux et systèmes en temps réel dédiés ou *embedded*.

Pour aller plus loin consulter wikipedia tout d'abord (anglais et français), mots-clés :VxWorks (robots sur mars), robots ASIMO, routeurs linksys, robots industriels.
L'ensemble des programmes utilitaires d'un mainframe est une culture en soi.
L'univers du *mainframe* est limité à un écran 80 x 24, 80 colonnes de 24 lignes, un peu plus avec les nouveaux terminaux.
La plupart des machines périphériques existantes ont été crées, inventées et standardisées par IBM : le disque dur (1956). Écran plat à plasma à 4 écrans (1983). Le dérouleur de bande, l'imprimante à impact rapide, etc. Ils ont des milliers de brevets.

L’alphabet est l’EBCDIC (8 bit), c’est comme l’ASCII des pc mais tout mélangé. Il existe 6 versions documentées de cet alphabet au moins et des variantes crées par des concurrents pour mélanger encore plus.

Les utilitaires du mainframe

JCL : job control langage, il permet de faire l'allocation des ressources, c'est un interpréteur (GLUE), qui sert à tout, noter les // en colonnes 1 et 2.

```
//IS198CPY JOB (IS198T30500),'COPY JOB',CLASS=L,MSGCLASS=X
//COPY01   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=OLDFILE,DISP=SHR
//SYSUT2   DD DSN=NEWFILE,
//            DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(40,5),RLSE),
//            DCB=(LRECL=115,BLKSIZE=1150)
//SYSIN    DD DUMMY
```

Essentiel à connaître pour un programmeur(euse) mainframe.

DFSORT, SYNCSORT ou CA-SORT permet de trier des fichiers mais l'appel de ces programmes se fait à l'interne du JCL en réservant 3 fichiers, original, copie de travail et sortie. Plus compliqué qu'Unix...

Les lignes du JCL sont des cartes de contrôle comme les cartes perforées, toujours à 80 colonnes, l'emplacement en colonne est essentiel. La plupart des langages mainframes sont stricts sur les colonnes, comme le FORTRAN et surtout le COBOL.

IEBGENER, permet la copie de data sets séquentiels ou partitionnés, tous les JCL ont un appel au IEBGENER sinon y'a rien qui fonctionne.
ICKDSF, permet d'initialiser des disques (DASD), plus rébarbatif comme nom faut le faire(!).
IEFBR14, permet d'effacer des data sets, appelé aussi 14 février (histoire du registre 14…).

Exemple de programme qui envoie un courriel.

```
//IEBGENER JOB  ACCT,'DATA COPY',MSGCLASS=J,CLASS=A
//NORMRC   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD *,LRECL=80
HELO <SYSTEMID>
MAIL FROM:<USERID@SYSTEMID>
RCPT TO:<USERID@SYSTEMID>
DATA
From: <USERID@SYSTEMID>
To: <USERID@SYSTEMID>
Subject: Test Mail

TEST MAIL FROM MAINFRAME
.
QUIT
/*
//SYSUT2   DD SYSOUT=(B,SMTP),LRECL=80
//SYSIN    DD DUMMY
```

Encore une fois, IBM a imposé son standard dans tous les aspects du mainframe. La nomenclature des messages, le nom des périphériques, les utilitaires sont tous originalement d'IBM.

Comme on l'a vu avec les machines mécanographiques, le besoin de traitement existait indépendamment des ordinateurs conçus pour le calcul comme l'ENIAC. La raison est commerciale d'abord. Il fut un temps ou la compagnie de thé LYON's a créé une machine de toute pièce juste pour facturer ses clients et faire la comptabilité, le LEO III qui disposait d'un haut-parleur directement branché sur le CPU ce qui aidait les programmeurs à savoir ou en était la machine. Les programmeurs appréciaient ce haut-parleur.

Donc, essentiellement un besoin de traiter des données, calculer des payes, faire un rapport comptable. Étant donné que ce sont surtout les banques et assurances qui ont ces besoins il était naturel que le mainframe s’y impose en force.

Donc, mis à part, les installations prévues pour des applications scientifiques comme la NASA, le gros du traitement informatique dans le monde a toujours été les transactions commerciales (pour les mainframes). Soit commercial ou bancaire.

-Il se transige environ 4000 milliards de $ par jour sur les marchés de change. C'est le gros du trafic boursier. Le temps de réaction des programmes ultrasophistiqués est de l'ordre du 1000ème de seconde. Avant même que quelqu'un n'ait même l'idée de vendre ou d'acheter et de se rendre à son clavier pour taper ENTER, la transaction est déjà passée, aucun humain ne peut réagir aussi vite (humain = 0,7 secondes au mieux).

-Carrefour (100 milliards d'euros par an), représente à peu près 4000 euros par seconde en transaction de ventes seulement (mon calcul).

-Wal Mart, le plus gros détaillant au monde, 2 millions d'employés, la base de données des ventes a 280 Téraoctets. Il en existe de plus grosses mais pour la recherche (climat, bio, LHC, CERN).
Le traitement de ces bases est comptable, débit, crédit, sommes. Mais aussi ces entrepôts de données servent pour le marketing. Le degré de précision est phénoménal et leur connaissance du client en est machiavélique (tous les grands distributeurs font des statistiques, voir Nielsen, en principe les données de ventes (CB) ne sont pas géographisées bien sûr…(les points air miles = 7% de plus de vente). Sur ce point précis, et ça justifie l'existence de ces grosses machines. IBM est champion mondial pour les transactions. Leur machine la plus puissante peut effectuer 10 300 000 transactions par minute. On parle ici d'écriture en base. La base est DB2, le système est AIX. Dans la

courte liste des plus grosses bases de données commerciales il y a IBM et Oracle essentiellement. Windows (sql server et cie) arrive à peine au 9ème rang. À toute fin pratique, ces machines sont soi sur Mainframe avec un Unix local (virtualisé pour IBM) et utilisent soit Oracle ou DB2.
NOTE : pour les grandes chaines de distribution (carrefour, Wal-Mart), 25% du chiffre d'affaire annuel se fait dans la période de Noël.
Ces mêmes transactions qui pour la plupart sont faites avec la carte bleue ; VISA et Mastercard se séparent la planète en 2.
Par exemple, VISA a traité 62 milliards de transactions en 2009 : 2000 transactions par seconde. La même chose pour MasterCard.
62 milliards de transactions, ça fait combien en euros ??!

Système	TPM	coût	base	OS
IBM Power 780 Server Model 9179-MHB	10366254	1.38 USD	DB2 9.7	AIX Version 6.1
Sun SPARC Enterprise T5440 Server Cluster	7646486	2.36 USD	Oracle Database 11g Ent. Ed. w/Real Application Clusters w/Partitionin	Sun Solaris 10 10/09
IBM Power 595 Server Model 9119-FHA	6085166	2.81 USD	IBM DB2 9.5	IBM AIX 5L V5.3
Bull Escala PL6460R	6085166	2.81 USD	IBM DB2 9.5	IBM AIX 5L V5.3
HP Integrity Superdome-Itanium2/1.6GHz/24MB iL3	4092799	2.93 USD	Oracle Database 10g R2 Enterprise Edt w/Partitioning	HP-UX 11i v3
IBM System p5 595	4033378	2.97 USD	IBM DB2 9	IBM AIX 5L V5.3
IBM eServer p5 595	3210540	5.07 USD	IBM DB2 UDB 8.2	IBM AIX 5L V5.3
PRIMEQUEST 580A 32p/64c	2382032	3.76 USD	Oracle Database 10g R2 Enterprise Edt	Red Hat Enterprise Linux 4 AS

			w/Partitioning	
PRIMEQUEST 580 32p/64c	2196268	4.70 USD	Oracle 10g Enterprise Ed R2 w/ Partitioning	Red Hat Enterprise Linux 4 AS
HP ProLiant DL580 G7	1807347	.49 USD	Microsoft SQL Server 2005 Enterprise x64 Edition SP3	Microsoft Windows Server 2008 R2 Enterprise Edition
IBM System p 570	1616162	3.54 USD	IBM DB2 Enterprise 9	IBM AIX 5L V5.3
Bull Escala PL1660R	1616162	3.54 USD	IBM DB2 9.1	IBM AIX 5L V5.3

Mais pourquoi le virtuel ? : Plusieurs bonnes raisons.
D'abord le coût de façon évidente.
La simplicité, une installation de Windows 7 assez complète avec 50 gigas d'espace disque n'occupe que 12 gigas réels sur disque et quelques fichiers faciles à déplacer au lieu de 100000.

Facile à déplacer ; vous savez tous comment il est presqu'impossible de déplacer une installation d'OS simplement en copiant le C (!!), ou même une installation d'un logiciel sous Unix (pas une bonne idée).
Que l'installation soit avec 1 couche supplémentaire ou presque native (voir petit film, on triche un peu avec le hardware).

La performance est pratiquement la même et pour un client + appli standard avec un utilisateur lambda : pas de différence notable.

Le principal avantage : l'OS se transporte très facilement et peut donc être déployé et reproduit tant qu'on veut.

Mais : un tas de machines VM n'est pas magique pour autant.

Exemple : une base de données est virtualisée sur un serveur ce qui est une bonne idée. Mais qui dit écriture (update, delete) sur une base implique que forcément le fichier (la copie VM) sera fragmentée.

La base sera fragmentée mais ça c'est normal, le programme de base a normalement des mécanismes pour veiller au grain. Par contre il faut veiller à ce que le fichier VM lui-même ne le soit pas trop, c'est une fragmentation de 2ème niveau en fait, forcément la couche logicielle en a 1 de plus! Bonne question : Si on veut que ça roule bien, dans quel ordre et combien de fois doit-on défragmenter les données/fichiers ?

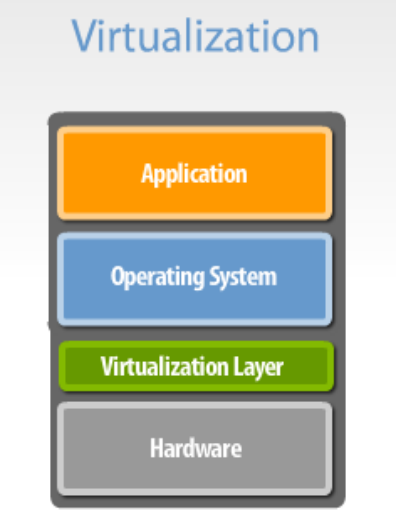
Si c’est NTFS, les fichiers sont empilés vers le fond, si c’est Unix avec ext4 c’est plus intelligent mais la fragmentation existe QUAND MÊME.

Un calcul simple, 1 installation Windows 7 peut compter facilement 100000 fichiers, donc si le fichier VM est fragmenté un peu : il n’y a pas de dégradation de la performance mesurable mais si on laisse aller ; dégradation significative. Les programmes de gestion VMware (très complets) ou VirtualBox ne tiennent pas compte de ça. On peut défragmenter avec VMware Workstation (payant ou gratuit 60 jours).

Ne pas oublier qu’en lecture pure, un bon disque dur récent arrive à lire 100 mégaoctets de données en 1 seconde facilement, 100 megs pour une mémoire de 4 gigas c’est peu. Exemple avec XML, Itunes.

http://download3.vmware.com/media/vam/vam_demo.html

Qui dit machine virtuelle dit aussi application virtuelle. Un raisonnement simple consiste à dire : pourquoi ne pas mettre tout qu'il faut pour un utilisateur lambda encapsulé dans 1 système transportable et/ou téléchargeable directement du serveur ?



La préemption, le multitâche et le time slicing.

La préemption est l'acte d'interrompre une tâche informatique et de la repartir plus tard sans intervention directe, cet acte se produit selon une commutation de contexte, *switch context*. Normalement c'est l'OS qui décide si une tâche est légère ou lourde selon que soit le thread utilise ou non certains registres du CPU.

En fait l'OS décide s'il faut mettre une partie du contexte d'un processus en mémoire cache (antémémoire en français), comme on le sait la mémoire sur un ordinateur est toute la même : c'est la vitesse d'accès qui caractérise la mémoire. La mémoire la plus rapide étant les registres, ensuite les caches de niveau 1, 2, 3 en ordre, la mémoire vive, le disque dur, lecteur DVD, les rubans, disquettes et même papier.

On distingue les OS comme étant soit préemptifs ou coopératifs, *collaborative*. Les systèmes préemptifs sont les plus récents.

Système	Codage	Mono-utilisateur	Multi-utilisateur	Mono-tâche	Multitâche
DOS	16 bits	X		X	
Windows3.1	16/32 bits	X			non préemptif
Windows95/98/Me	32 bits	X			coopératif
WindowsNT/2000	32 bits		X		préemptif
WindowsXP	32/64 bits		X		préemptif
Windows7	32/64 bits		X		préemptif
Unix / Linux	32/64 bits		X		préemptif
MAC/OS X	32 bits		X		préemptif
VMS	32 bits		X		préemptif

Certains systèmes ont été conçus pour être préemptifs dès le départ comme Unix, Amiga, BeOS (défunt) et le NeXTSTEP, d'autres le sont devenus : à partir de Windows NT 3.1 et Mac OSX depuis 2001.

Il faut distinguer multitâche avec le time slicing.

Avec les systèmes TSO/MVS ou VM/CMS, c'est simultané en apparence seulement, TSO voulant plus dire que plusieurs utilisateurs pouvaient être connectés via un terminal simultanément. Le terme Time Sharing n'est plus utilisé, on utilise multi-tasking plutôt. Historiquement, le premier système préemptif complet a été OS2 qui d'ailleurs a été utilisé dans les DAB pendant quelques années, maintenant c'est Windows XP (selon mes informations). Les distributeurs de billet de la SNCF utilisent Windows XP, sont très lents, n'ont qu'une seule tâche à effectuer et prennent 20 minutes pour booter…mais je ne sais pas

pourquoi.! Voir les nombreuses pannes informatiques de la SNCF, le plus drôle est l'explication qu'ils en donnent…

On peut se rendre compte du comportement du système à la façon dont les programmes qui sont gelés en mémoire pouvaient être interrompus ou non avec CTRL-ALT-DEL ou l'équivalent sur Mac et le comportement du système selon qu'on est en Windows 95 ou Windows 7, la même évolution a eue lieu avec le Mac entre le système 9 et Mac OSX.

Les super machines

91,8% des supercalculateurs utilisent Linux, en hausse de presque 1% depuis la fameuse liste des top500 de 1993.

La liste top500, est une liste des 500 plus grosses machines sur terre. Établie en 1993, à l'époque le CRAY Y-MP pouvait faire 13 Gflops.

1 flop = 1 opération en virgule flottante à double précision. 1 GFlop = 1 milliard d'opérations par seconde.

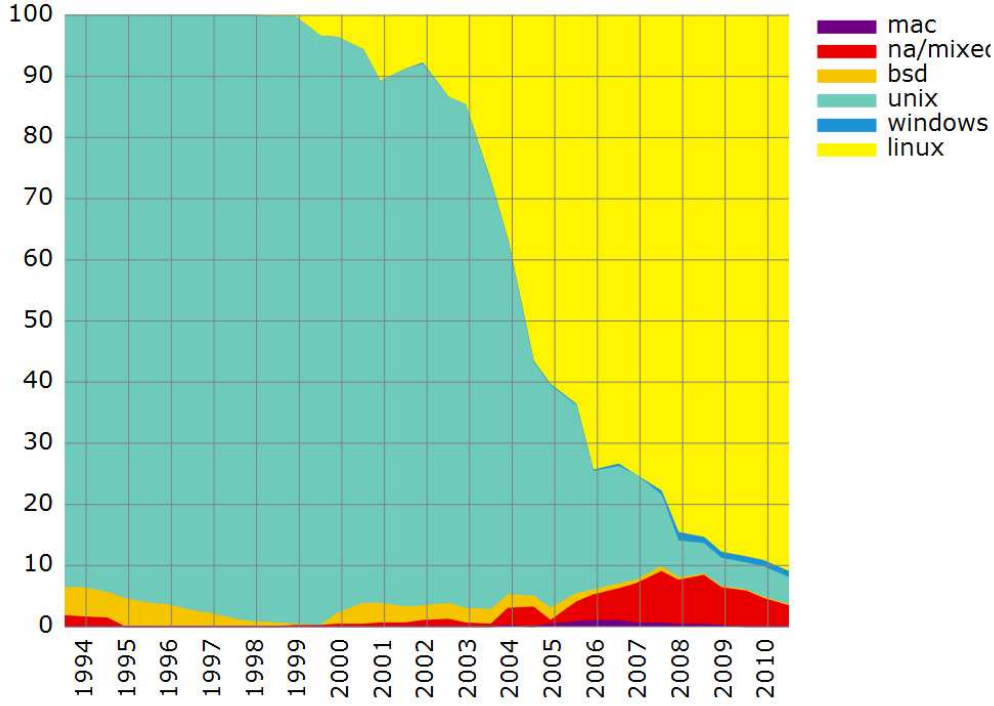
Le Cray Y-MP : aussi rapide qu'un portatif aujourd'hui, un humain normal peut se procurer un pc qui équivaut aux meilleures machines sur terre d'il y a seulement 10-12 ans.!

Une différence assez marquante s'est établie depuis 1993. Le système linux est devenu omniprésent. Le système du Cray était COS (Cray operating system), une espèce de système monté maison par quelques anciens de CDC (control data), du fait sur mesure. Le nom Cray vient de Seymour Cray, un ancien de CDC qui est parti à son compte.

Ces systèmes montés maison, ou home made n'existent plus sauf pour des applications très particulières. En d'autres mots, tous les gros systèmes aujourd'hui utilisent Linux.

Pour le hardware, il y a eu également une tendance vers l'empilement des pc. (Je m'explique). Une machine comme le CRAY était custom made, ou montée exprès, l'architecture et même la forme en demi-cercle était pour aller plus rapidement, l'argument de la longueur des fils très connu depuis Grace Hopper…

Maintenant, la tendance est d'avoir en fait 1 pc bien monté x un certain nombre de fois. Les pc qui sont utilisés sont les mêmes que n'importe qui peut acheter directement au magasin du coin. Sauf qu'évidemment la réseautique, la LATENCE du système n'est pas de l'ordre de grandeur d'un simple PC.



Parts de marché des systèmes depuis 1994.

De plus, il faut comprendre que ces machines sont faites pour le calcul parallèle. Si un problème linéaire, d'un seul tenant, un calcul unique simple mais très long demande beaucoup de calcul, ces machines n'en sont pas ou peu capables. Elles sont très efficaces SI le problème à résoudre peut se briser en milliers de morceaux indépendants. Le calcul météo, l'indexation d'internet (le monstre de Google) en sont des exemples. Google utilise d'ailleurs cette approche. Par contre : l'engin Google n'est pas considéré comme un supercalculateur parce que les machines de leur système ne sont pas toutes reliées dans un même système, même s'ils en ont 450 000. Les spécifications techniques du monstre Google sont intentionnellement vagues, ça fait

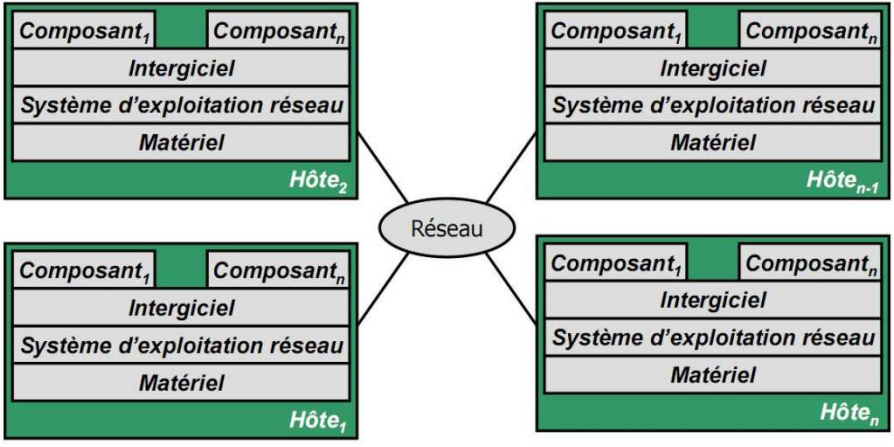
partie du mystère. On connait les spécifications que de façon indirecte (factures d'électricité de leurs centres).
La plus rapide au monde est maintenant chinoise.

Le monstre chinois.
On annonçait récemment que les chinois avaient réussi à construire l'ordinateur le plus rapide du monde.

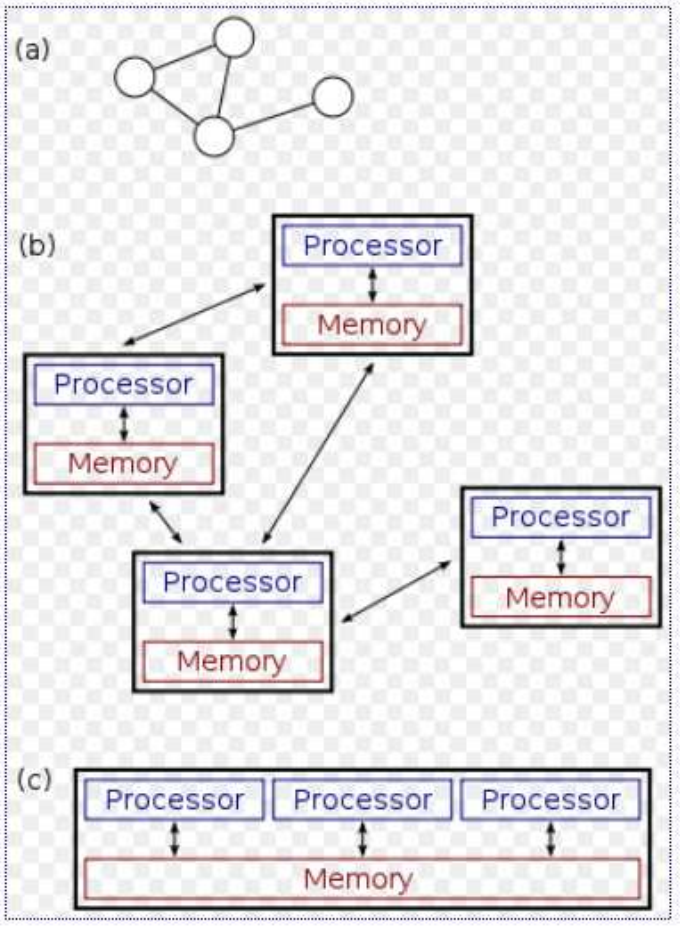
14336 processeurs Xeon X5670 (6 cœurs) à 2.93 Ghz, 1,170 milliards de transistors chacuns.
7168 cartes Nvidia M2050, 3 milliards de transistors, 448 cœurs.
2048 processeurs chinois (spécifications techniques non disponibles).
En tout : 3,75 x 10^13 transistors.

2,507 Pétaflops ou 2,507 x 10^15 opérations/seconde en virgule flottante double précision (15-16 chiffres décimaux).
Par comparaison, le cerveau humain contient 10^15 connections synaptiques dont la densité est de l'ordre de 1 milliard par millimètre cube.
Le OS est basé sur Linux bien sûr mais surtout l'architecture est en fait un bon PC équipé de 2 CPU Xeon couplés à 1 carte NVIDIA spécialisée + une très bonne connectique réseau. À noter que le précédent était le Jaguar CRAY qui faisait 1,2 Pétaflops.
Ça ce sont les supercalculateurs qui comme on le sait occupent autant de place que l'ENIAC mais qui vont des milliards de fois plus rapidement.

Dans la terminologie de calcul il existe des problèmes parallélisables et ceux qui sont trivialement parallélisables, tellement que c'en est embarrassant (*embarassingly parallel*).
Si un problème est E.P. il est peut être candidat pour les systèmes quasi-supercalculateurs. On classe les machines dans cette catégorie les systèmes de calcul basés sur internet. Ce sont des milliers voire millions de machines réservées pour un même calcul.
On appelle ces ensembles ou quasi-supercalculateurs : le calcul distribué. La différence notable est que la latence bien plus grande que dans un supercalculateur mais le



calcul étant totalement indépendant, cela importe peu par rapport au résultat.

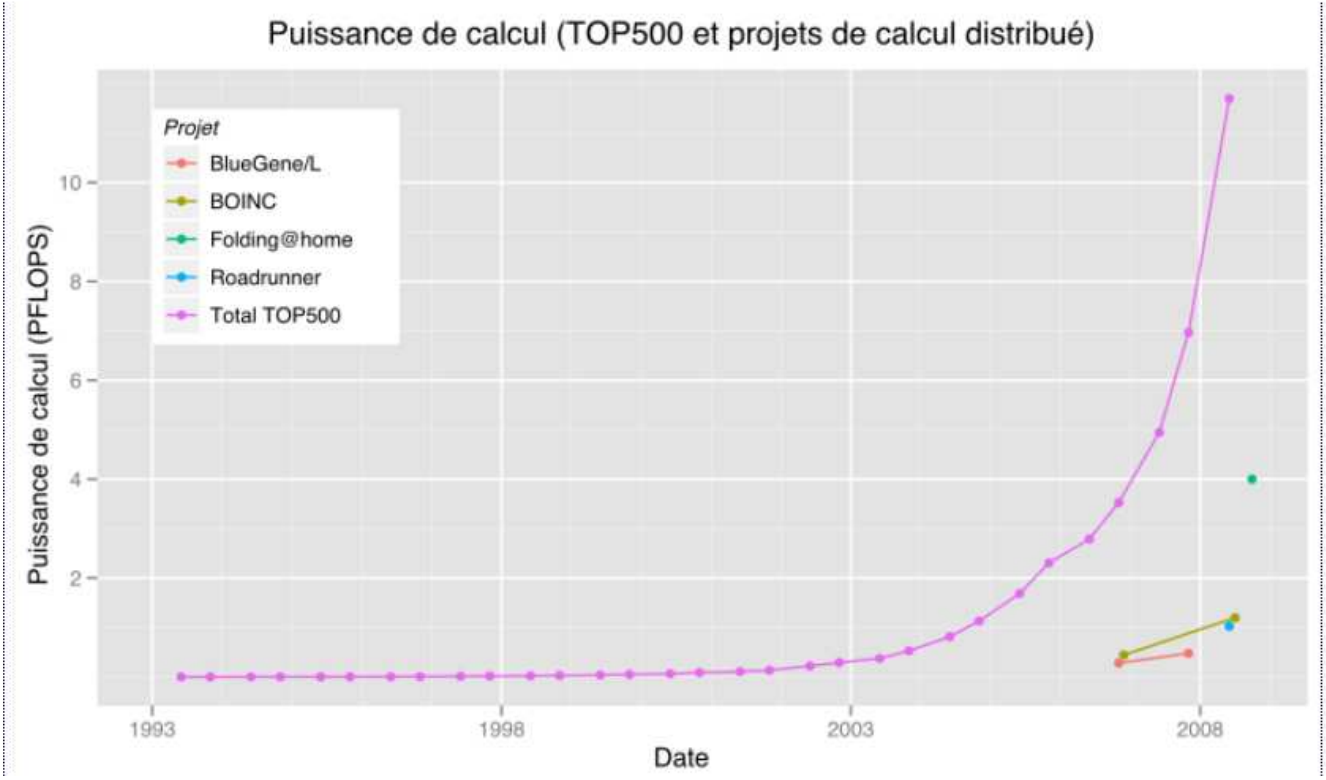


Modèle de calcul distribué (a-b) et parallèle(c).

On voit bien que le calcul est distribué de même que la mémoire, chaque ordinateur est indépendant. Tout est dans le passage des messages (sur internet).

Un des points forts de tels systèmes est la tolérance aux

pannes, ce qui n’est pas le cas d’ENIAC ou du monstre chinois.



On voit ici la courbe exponentielle et en petit les projets comme BOINC

La clé du fonctionnement de ces projets de calcul distribué est le passage des messages

Les projets actuels incluent :

Le GIMPS, recherche de grands nombres premiers de Mersenne, c'est grâce à eux que les 14 derniers ont été trouvés, le dernier est 2^43112609 – 1, 12,9 millions de chiffres. En fait les ordinateurs séparés essaient tous les exposants possibles.

Le SETI, analyse des signaux venant des radiotélescopes comme Aracibo.

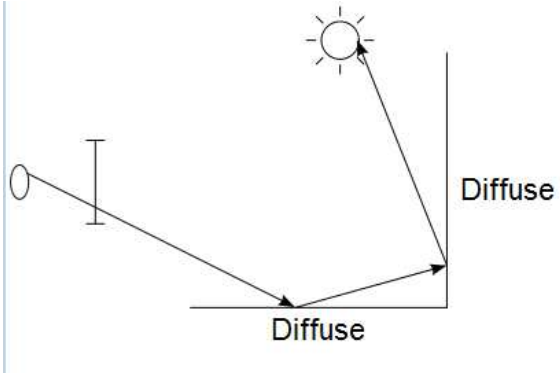
Calculs cryptographiques, des milliers de machines qui essaient de façon exhaustive toutes les combinaisons possibles du RC-72, c'est une clé de cryptographie basée sur 72 bits, donc 2^72 possibilités,

représente 4,72 x 10^21 clés à essayer, ça peut marcher mais c'est assez bête comme calcul.

La quantité de calcul que ces systèmes sont capables d'effectuer est calculée en nombre d'opérations.
Le SETI : un cumul de 10^22 calculs, considéré comme le plus gros projet de calcul jamais fait.
Le film NEMO, 10^18, Shrek III, 20 millions d'heures de CPU ou 10^19.

Pi de X ou $\pi(x)$, le nombre de nombres premiers inférieurs à X, le calcul a atteint 10^21. Zimmermann et Deléglise.
Les films d'animation utilisent Unix et des programmes spécialisés comme Renderman, on appelle ces ensembles de machines : *render farm* . Ce sont des pièces énormes remplies de PC qui font du rendu d'image (fermes de rendu). Une seule scène de Shrek 1 au début

compte 1 million de feuilles d’arbre qui bougent…Le ray-tracing est un calcul essentiellement indépendant.



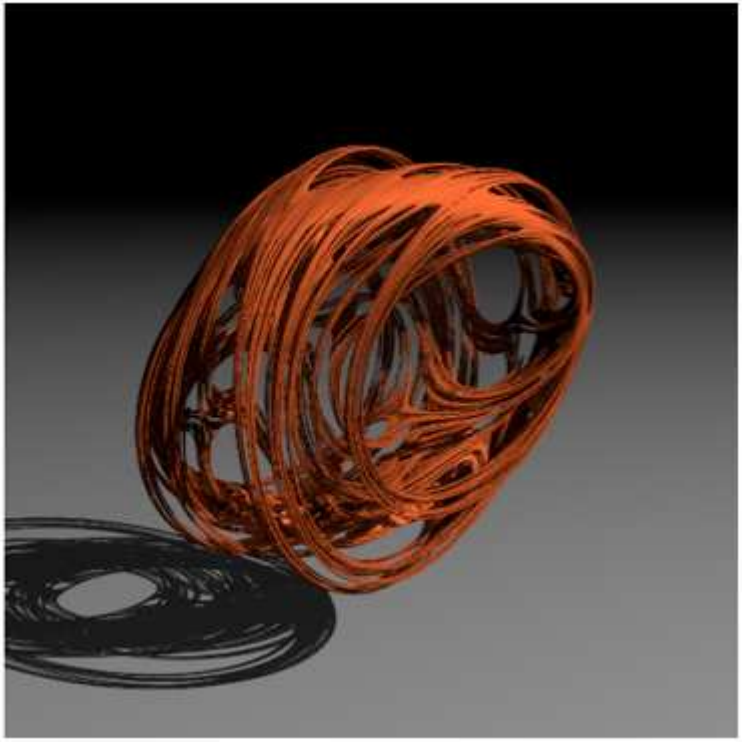
Les calculs scientifiques des gros calculateurs sont essentiellement en fortran ou en C. Le fortran a régné longtemps et est encore largement utilisé, tellement qu'en fait il existe des programmes de traduction entre les 2 languages. Les calculs de ray-tracing utilisent massivement Unix, les programmes spécialisés Renderman et cie, les bibliothèques Open-GL.

Ces derniers justement sont simples (voir images) mais il y en a beaucoup ! C'est ce besoin qui a donné naissance aux cartes graphiques comme NVIDIA. D'ailleurs le dernier né des gros calculateurs en utilise 7168, de vrais monstres de calcul en soi mais en

théorie. Ces cartes n'ont pas la même capacité de calcul qu'un Intel haut-de-gamme qui utilise des instructions de haut niveau (instructions SSE n).

Certains programmes tout-terrain comme Mathematica 8 peuvent prendre avantage des GPU (comme la NVIDIA M2050) à l'aide de messages passés par le Math-link, une facilité du programme qui permet d'adresser la GPU. Voici un exemple de rendu grâce à Mathematica couplé à un(e) GPU. Maple (version canadienne) peut le faire aussi si les opérations sont de type matriciel.

Out[2]=



Certains hackers ont réussi à craquer un code cryptographique MD5 à l'aide d'une carte NVIDIA qui peut en essayer 502 millions par seconde.

Par contre, l'écran est inutilisable durant ce calcul… et ça chauffe !

Références, bibliographie

-Andrew Tanenbaum, *Systèmes d'exploitation*, Pearson Education France, 2008, 3e éd.
-Laurent Bloch, *Les Systèmes d'exploitation des ordinateurs. Histoire, fonctionnement, enjeux*, Vuibert, 2003.
-Wikipedia (en) Operating Systems, (plus complet qu'en français), http://en.wikipedia.org/wiki/Operating_system

résumé
http://www.commentcamarche.net/contents/systemes/sysintro.php3

Le Cobol : http://www.cobolportal.com/developer/future.asp?bhcp=1

Les 6 programmeuses de l'ENIAC :

http://www.witi.com/center/witimuseum/halloffame/1997/eniac.php
http://www.ada-online.be/frada/spip.php?article102
http://www.witi.com/center/witimuseum/halloffame/1997/eniac.php

Site du TPC : http://www.tpc.org/tpcc/results/tpcc_perf_results.asp
Les langages GLUE (en anglais) : http://en.wikipedia.org/wiki/Glue_language
Les aspects visuels des fenêtres du macintosh :
http://www.knowledgerush.com/kr/encyclopedia/MAC_OS_X/

Documentation sur IBM :
http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?resultof=%22racf%22 toute la documentation est librement accessible pour tous les systèmes y compris MVS appelé Z/OS maintenant.

Définition des cylinders, tracks, volumes sur les DASD IBM :
http://publib.boulder.ibm.com/infocenter/zos/v1r9/index.jsp?topic=/com.ibm.zos.r9.hasa300/tekduse.htm

Diagramme des familles Unix détaillé :
http://upload.wikimedia.org/wikipedia/commons/1/11/Unix-history.svg

Alphabet EBCDIC :
http://publib.boulder.ibm.com/infocenter/zos/v1r11/index.jsp?topic=/com.ibm.zos.r11.csfb400/e2aa2e.htm