

Architecture : Mac OS X un système d'exploitation moderne ?



I. Les origines / Présentation

- **Besoins et Héritage de Mac OS X**
- **Technologiques**

2. XNU + Darwin

- **La structure du noyau**
- **Le sous-système BSD**
- **Quelques particularités**

3. Les couches hautes

- **Quartz**
- **API's**
- **Les promesses de “Snow Leopard”**





Vocabulaire : Mise au point

UI + dépendances

BSD

Kernel

Aqua
Closed Source
(90%)

Darwin 9.5
Open Source
[http://
www.opensource.apple.com/
darwinsource/](http://www.opensource.apple.com/darwinsource/)

MacOS X
“Léopard”





MacOs “Classic” 1984 / 2003 (RIP)

Philosophies :

- => Finder
- => Cacher la complexité
- => It Works, pas d'opérations inutiles
- => ERGONOMIE

Défauts :

- => le code! (pas portable : beaucoup d'assembleur)
- => le système n'est pas moderne
- => Il a fallu trouver une solution radicale!





8 projets infructueux (!) : pink, copland, startrek, TalOs, Raptor...

Volonté d'un OS moderne (1996) :

- Mémoire protégée
- Multitâche préemptif
- Mémoire virtuelle
- Portable
- Modulaire
- Réseau



Rachat de la dernière chance...

- Be trop cher, pas d'expérience commerciale
 - Next, Steve Jobs, NextStep
- => problème : pas dans la philosophie de Mac OS classic...





Le plus ancien système d'exploitation en développement

=> 1987

=> 1975 pour le noyau : RIG puis Accent et Mach

Provient de OpenStep (NextStep) V.4.2

=> S'appuie sur tous les fondamentaux Unix : système, APIs, sécurité...

=> Orienté Graphique + Console

=> Programmation orientée objet et graphique

=> Api riche





RIG
≈ Mach 0.8, mid 1970s
Accent
≈ Mach 0.9, circa 1979

Mach 1.0
Project started in 1984
USENIX paper in 1986

4.3BSD

Mach 2.0

NEXTSTEP 0.8
October 1988

Mach 2.5

NeXT additions to Mach

NEXTSTEP 1.0
September 1989

2.0
September 1990

Mach 2.6

2.1
March 1991

3.0
September 1992

Mach 3.0

3.1
May 1993

3.2
October 1993

3.3
February 1995

OpenStep
Specification
1994

OPENSTEP 4.0
July 1996

4.1
December 1996

4.2
January 1997

Apple buys NeXT
February 4, 1997

Documents de “Mac OS X Internals: A Systems Approach”





NextStep en quelques mots...

- Mach pour le Kernel
- 4.3 BSD pour le Sous-système Unix
- API's de Haut niveau Orientées Objet
- Graphique 2D : Postscript
- Système orienté professionnels
- Orienté Réseau
- Closed-source
- “Moderne”





MacOS X ajoute :

- XNU pour le Kernel “X is Not Unix”
- FreeBSD pour le Sous-système Unix
- Système fondamental Open-Source (Darwin)
- API spécialisées par activités (graphique, son, image, video, animation...)
- Serveur graphique PDF + OpenGL
- Interface uniforme proche de macOS classic
- Certifié UNIX03 par l’openGroup ! (leopard) :
1742 interfaces de programmation standardisées





Darwin = Kernel + BSD + Services Apple
=> Distribution OpenSource de la partie UNIX de
MacOs X (Actuellement 10.5.x = Darwin 9.x)

Kernel XNU

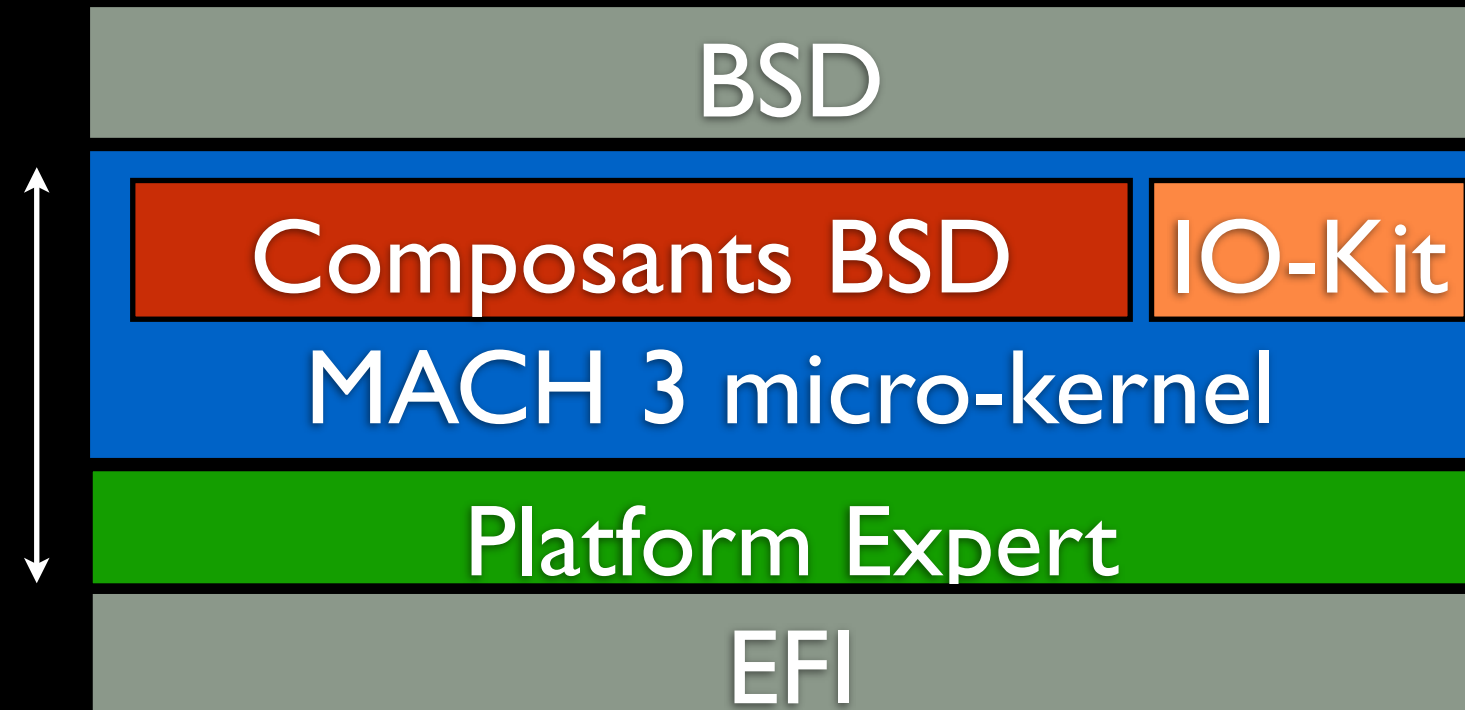
- => Kernel dit hybride
- => Mach 3 : Micro-noyau
- => Composants BSD
- => Utilisation de codes et techniques éprouvées (Unix), commercialement viables
- => Environnement d'exécution natif :
Mach-o





Le noyau XNU : Mach 3 + FreeBSD 5

XNU
Kernel-Space



Mach 3	Composants BSD
Mémoire virtuelle	Compatibilité Unix / Posix
RPC	Signaux
Ordonnanceur préemptif + SMP	Sécurité Unix
Protection mémoire	Couche réseau / TCP-IP
Temps réel	Processus BSD
Communication inter-process	Virtual File System
	Liste de contrôle d'accès



Kernel XNU = assemblage de plusieurs composants qui ensemble forme en apparence un Kernel hybride

=> Une seule Tache

=> Chaque composant s'exécute en tant que groupes de Threads

Platform expert : (~ HAL) Isolation des spécificités liées à une architecture

=> Portabilité

Driver changeable à chaud (Kernel Extention ou Kext)

API des Kexts orientée objet : IO Kit





Sous-système BSD de Darwin

Aqua

Shell Text (Optionnel)

Core Services

Launchd

Librairies BSD

XNU

Darwin



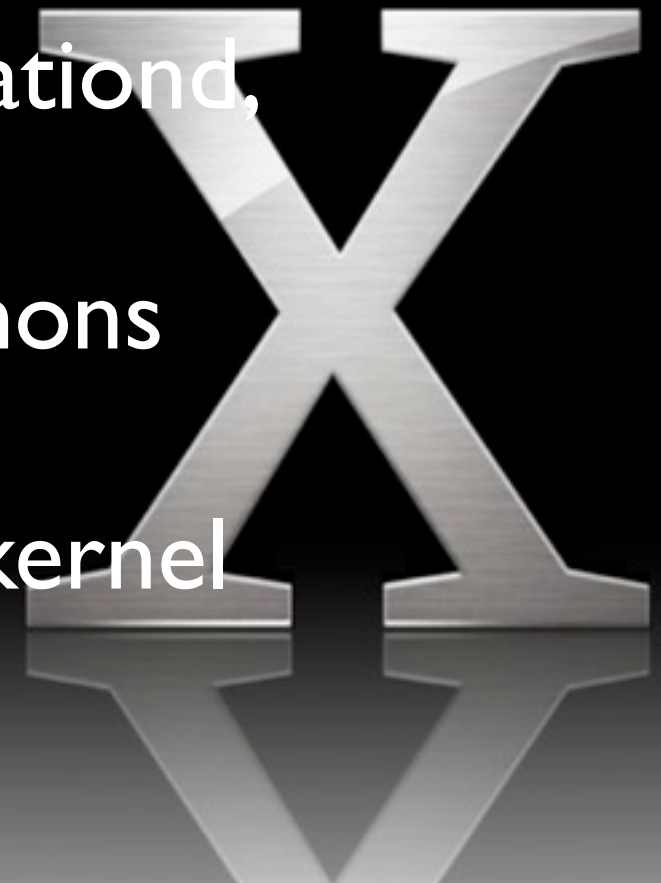


Au dessus du noyau :

- => Sous-environnement BSD (FreeBSD)
- => Services Unix standard = Core Services (Daemons)
- => Services Apple = Application Services (Services)

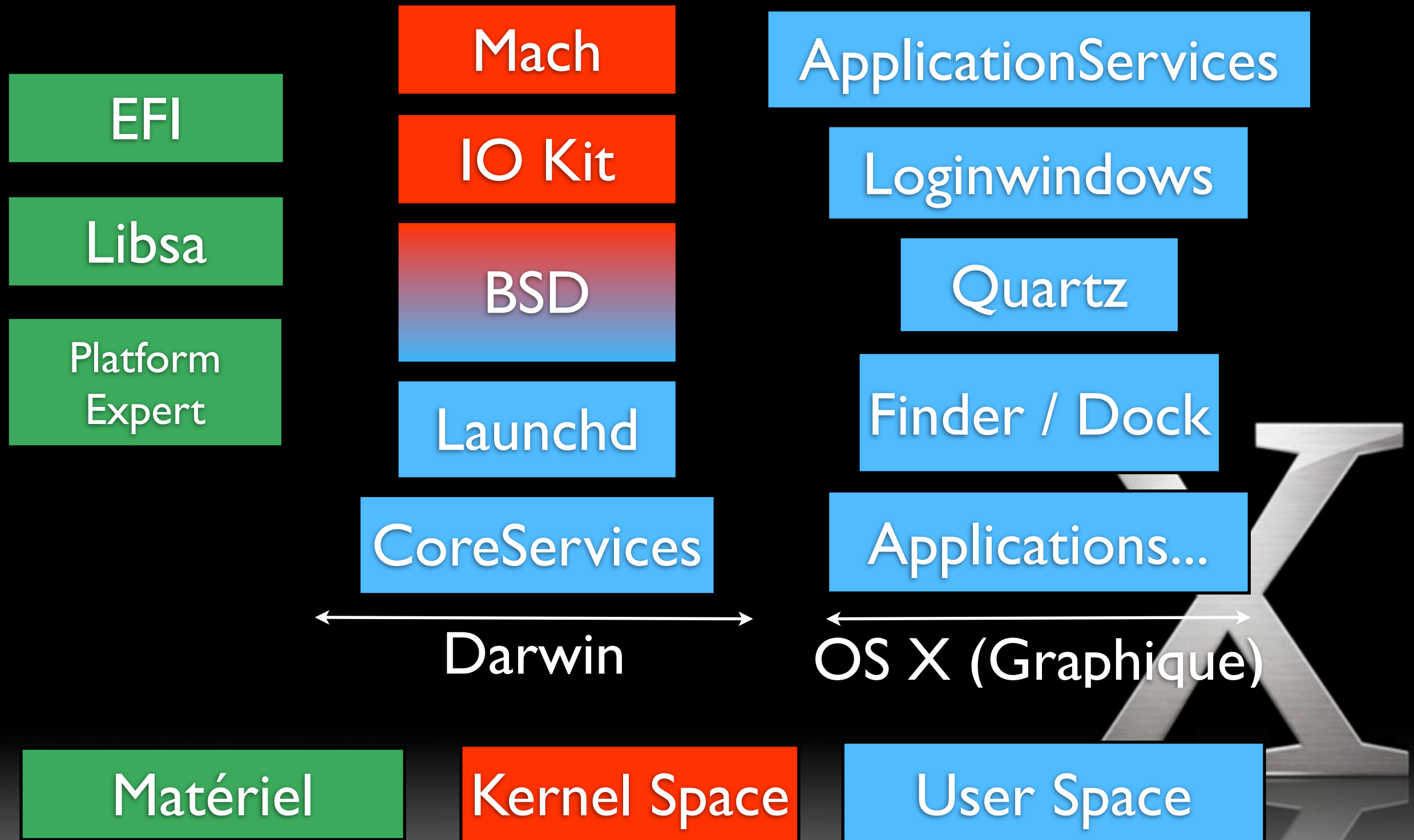
La plupart des Core Services sont fondamentaux pour fonctionnement du système de base

- => Exemple : dynamic_pager, diskarbitrationd, Kextd
- => SystemStarter : Chargement de démons
- => Securityd : Sécurité / cryptographie
- => Notifyd : Transmet des message du kernel
- => mDNSResponder : Bonjour





Initialisation du Système





Diviser (les fonctionnalisés) pour contrôler (l'évolution)

=> La modularité architecturale permet de prévenir les évolutions technologiques et systèmes même lourdes

Mac OS X essaie de se défaire d'anciennes traditions Unix peu confortables :

Utilisation de Property List (XML) pour enregistrer les paramètres systèmes et applicatifs réparties en domaines
`org.apple.xxx.yyy`

=> Remplace les fichiers plats





Pas d'utilisation par défaut de l'authentification BSD => OpenDirectory (basé sur OpenLDAP)

```
##
# User Database
#
# Note that this file is consulted directly only when the system is running
# in single-user mode. At other times this information is provided by
# Open Directory.
#
# This file will not be consulted for authentication unless the BSD local node
# is enabled via /Applications/Utilities/Directory Utility.app
#
# See the DirectoryService(8) man page for additional information about
# Open Directory.
##
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
```



OSX et le Support Multi-Architecture

Supports de différentes architectures matérielles

=> “Cross-platform” par design

=> Généricité du code

=> Platform Expert

Portages : CISC / RISC : Intel, PowerPC, ARM

Support (élégant) du 32-bit / 64-bit sur une même architecture matériel compatible

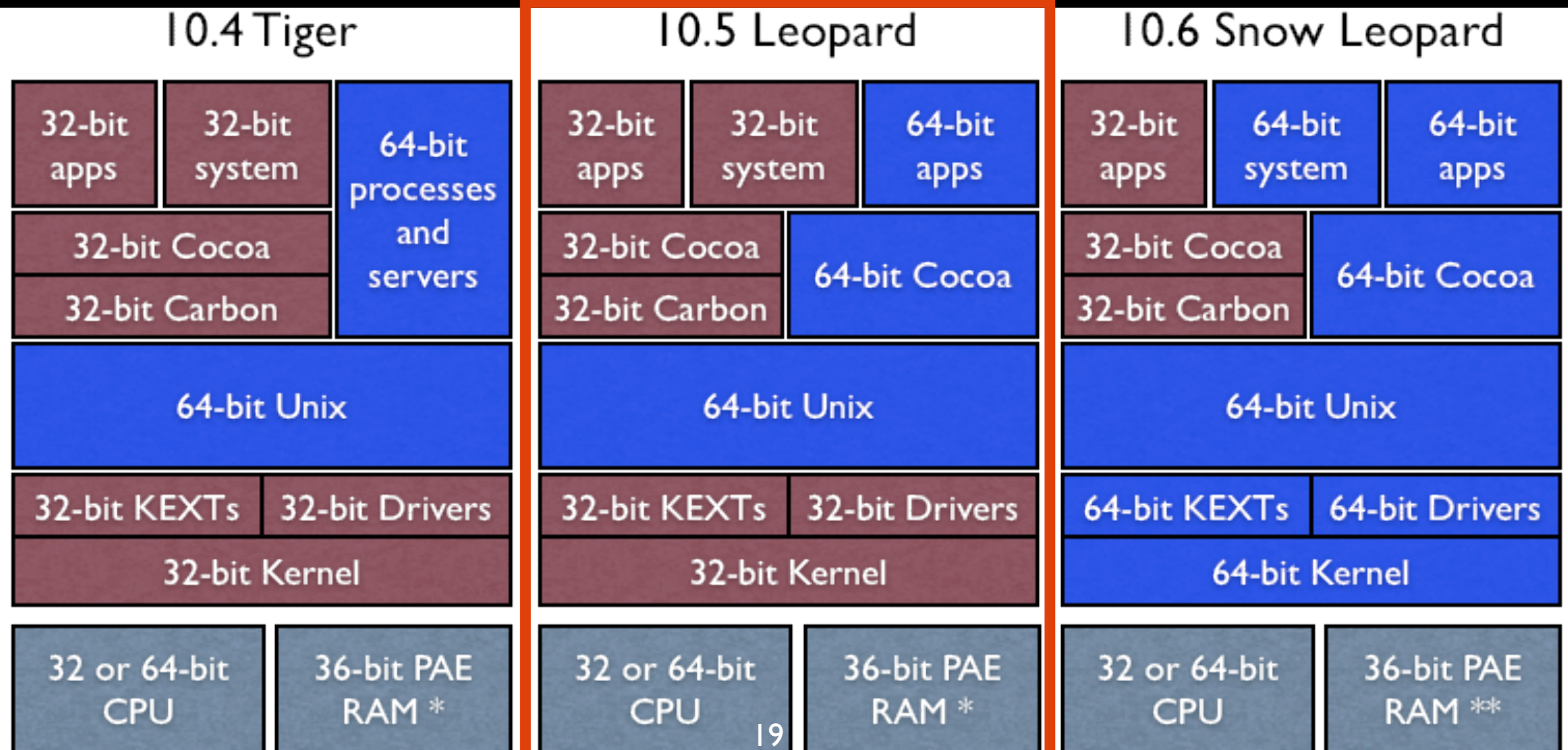




Tiger et Leopard : OS 32-bit supportants les applications 64-bit (incrémentalement) mais Drivers 32-bit

Snow Léopard : OS entièrement 64-bit avec support 32-bit natif !

=> Seule contrainte : Drivers et Plugins 64-bit obligatoirement





Enjeux ?

=> Technico-commerciaux

Intérêt du 64-bit ?

=> Adressage mémoire 16 TO (voir plus...)

=> Registres supplémentaires (Intel)

=> Calculs des grands nombres en
1 instruction

1 seul binaire, Comment est-ce possible ?

=> Universal Binary

=> Mach-o





Mac OS X sais créer et lancer des binaires multi-architectures, ce sont les “universal binaries” ou “fat binaries”

Comment ? Format d'exécution Mach-o le prévoit :

- => Le header précise le type de processeur
- => Le nombre d'architectures du Fat Binary
- => L'offset du début de chaque binaire
- => Le nombre d'octets de chaque binaire
- => Concaténation des binaires

Compilé par une version modifiée de GCC 4.x

OS X Intel : exécution du code PPC par un traducteur d'instruction “Rosetta”





OSX et ses fichiers...

XNU Travail sur une abstraction de systèmes de fichiers

- => VFS (Virtual file system)
- => Kauth, encapsule les “Acces Control List”
- => DiskArbitration, CoreService, Gère les volumes

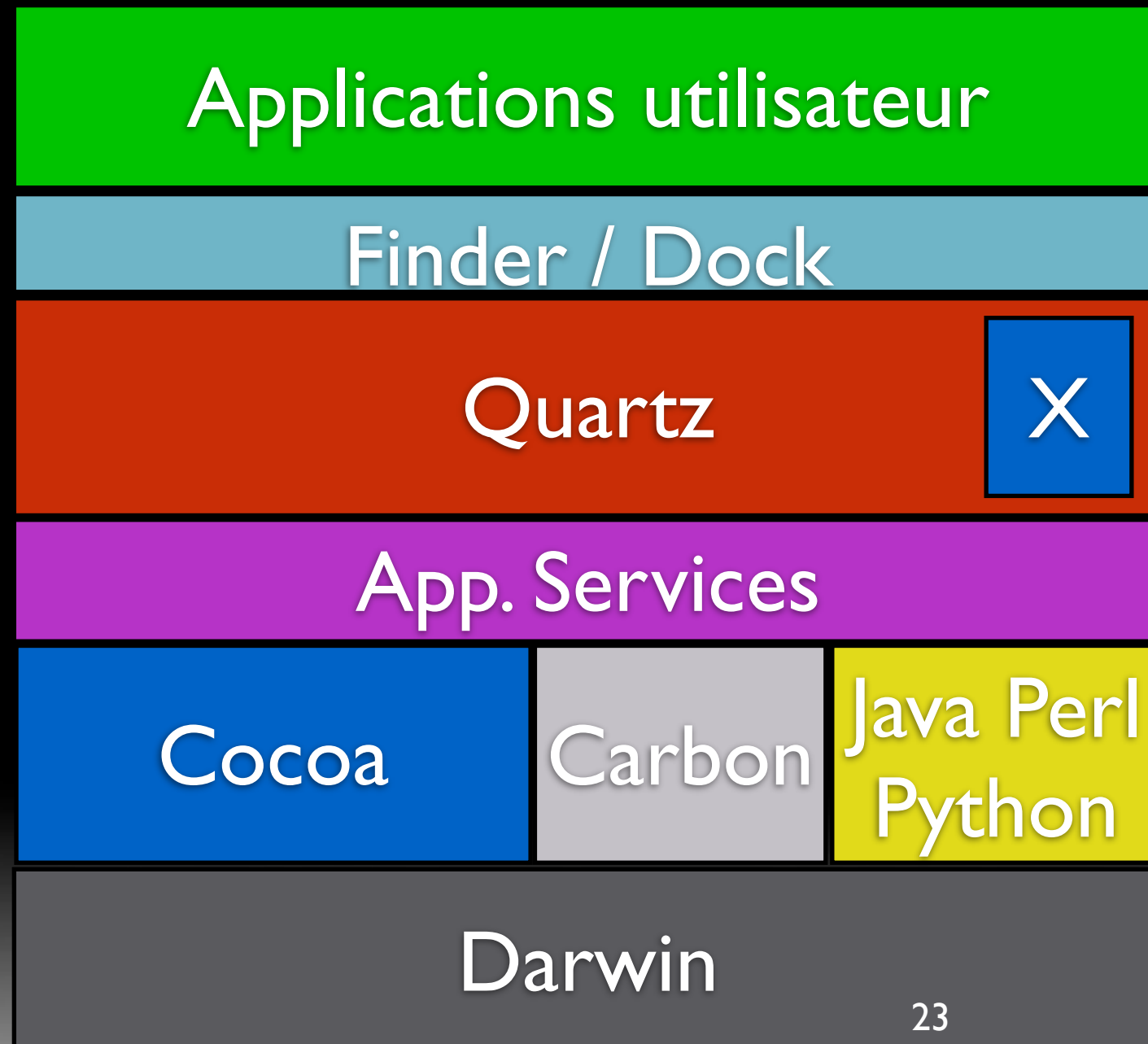
HFS+, FS par défaut

- => Ancien mais complet
- => Comparable à NTFS
- => Journalisé, Forks, Suivi de liens...
- => Défragmentation à la volée
- => Défauts : Pas de cryptage (Mais FileVault, CoreService), pas de compression
- => Mac OS X sait exploiter ces fonctions!
- => Avenir : ZFS ?





“L’esprit” MacOS X

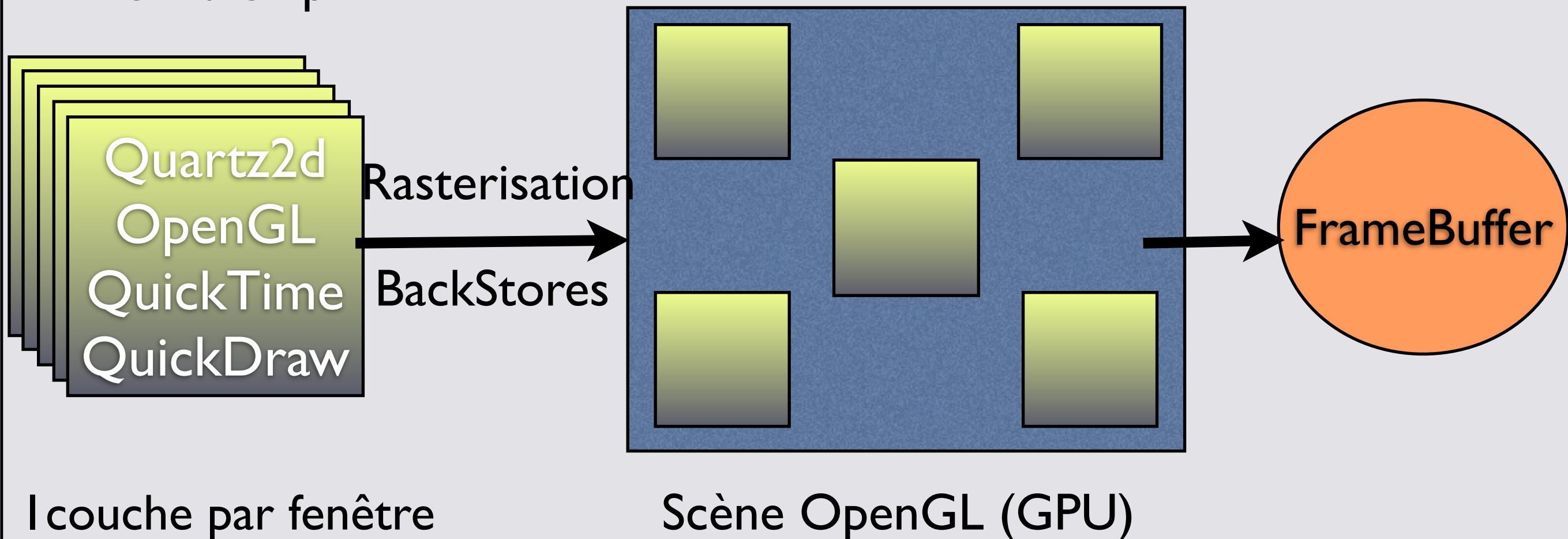




L'affichage dans Mac OS X

Quartz2D CPU ou
GPU si QuartzGL,
Rendu en pdf

Composeur Quartz Extreme





Api's



Carbon : En C, en cours de dépréciation

=> Pas de 64 bit

=> Compatibilité descendante (phase de transition)

Cocoa : L'Api -NATIVE- hérité de NextStep

=> Objective-C 2 (SuperSet du C)

=> 32/64-bit

=> Garbage collection

=> Tous les nouveaux Framework (et Cores)

=> Incite aux structures de type MVC



Core Image



Core Audio



Core Video²⁵



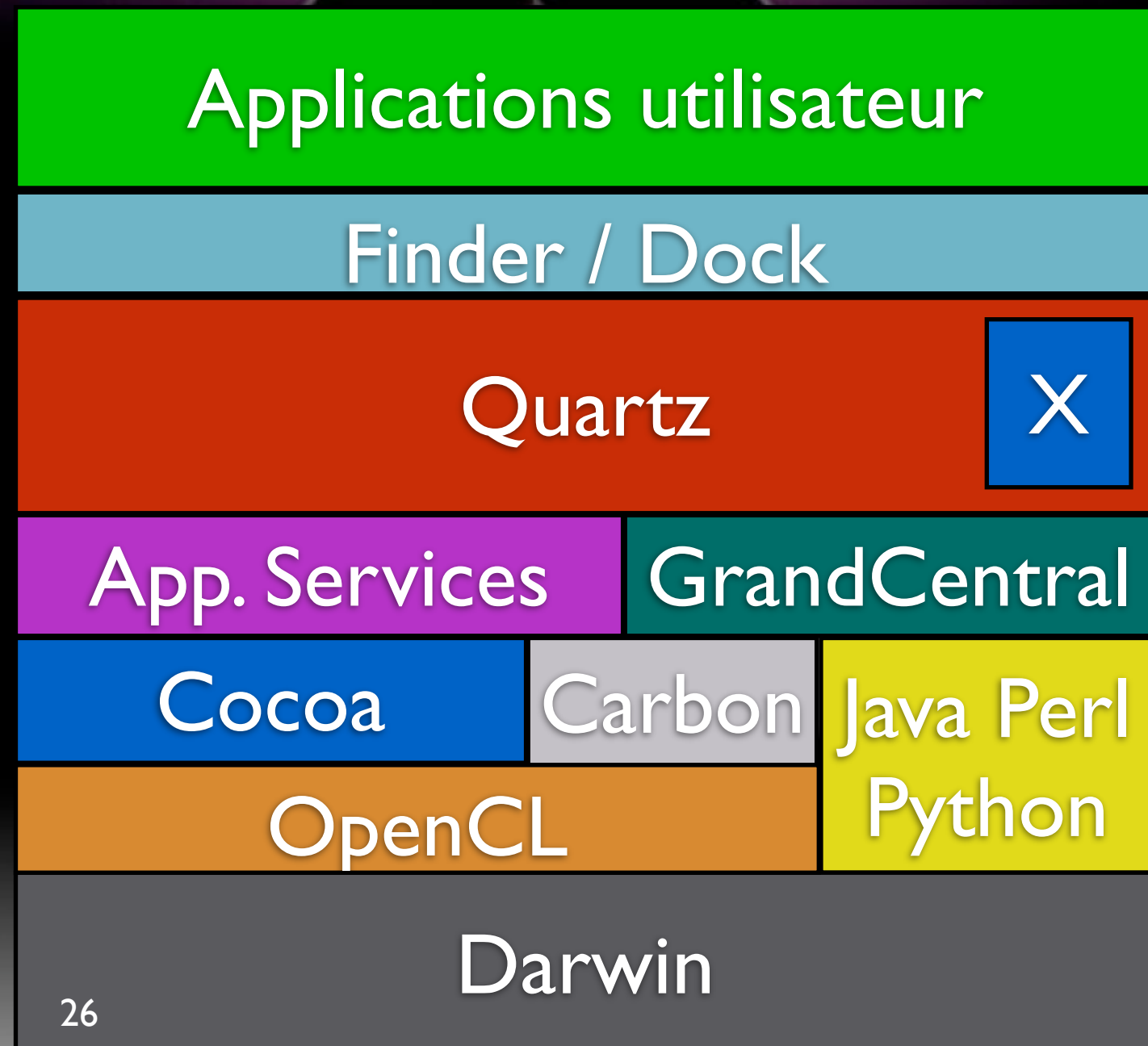
Core Animation



Mac OS X Snow Leopard

L'innovation au coeur.

Ajouts :
OpenCL
GrandCentral
64-bit
...





OpenCL

- => API pour utiliser le processeur Graphique GPU en tant qu'unité de calcul générique
- => Utilisation d'un langage intermédiaire (Bytecode)
- => Utilisation d'OpenCL dans tous les niveaux du système
- => Ratification en un temps record!
- => Indépendant du matériel





GrandCentral

- => “Super Ordonnaceur” + Api
- => Utilisation dans tous les niveaux du système
- => Vision du travail en tant que flux, découpage en paquet
- => Prédiction des points de congestion des UC
- => Support d'un nombre illimité de CPU
- => pas de gestion de Threads manuelle

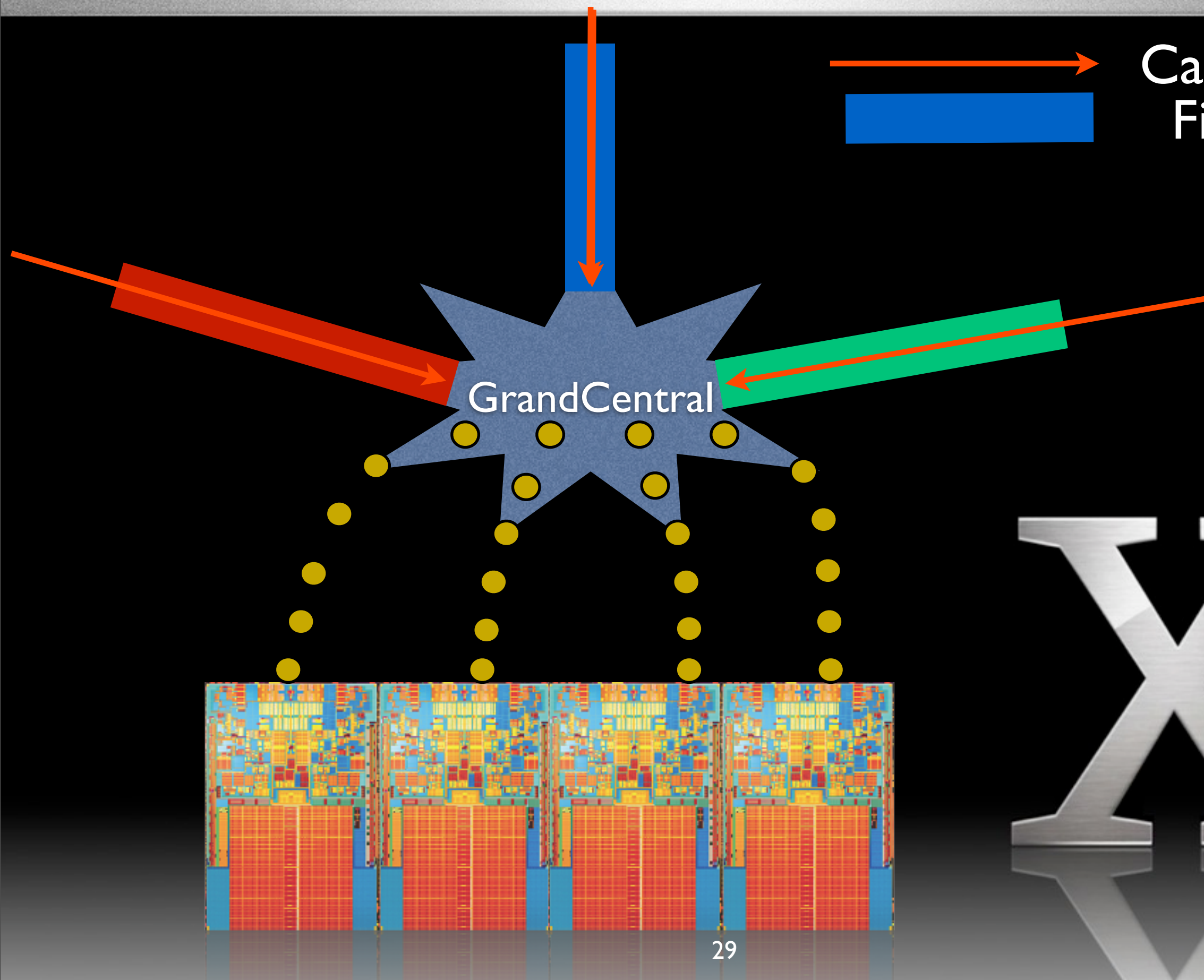




Grand Central



Calculus
Files





Moderne ? : OUI...

Hybride : Un système qui utilise de nombreux principes Unix mais qui apporte beaucoup de nouveaux concepts

Systeme modulaire et organisé en couche

Et... Résolument tourné vers l'avenir

=> Beaucoup de migrations coûteuses :

Classic -> OS X

PPC -> Intel (+ARM)

Carbon -> Cocoa

32-bit -> 64-bit

SMP -> Multiprocessing avancé

CPU -> CPU+GPU





Mac OS X Internals: A Systems Approach

By Amit Singh

www.arstechnica.com

www.opensource.apple.com





Démo

