Architecture de Ordinateurs

Module 8

Démarrage d'un système
Système d'exploitation
Langage et architecture

Dr. Yannick HERVE

V 1.0

Note

On ne s'intéresse ici au Système d'exploitation (OS) que comme la couche qui interface les applications avec le matériel

➔ Dernière couche de l'architecture

La vue « informatique » sera traitée dans les autres modules (cours de Pierre David)

Préliminaire
Démarrer l’ordinateur

Bootstrap (amorçage)

Comment démarrer une ordinateur qui n'a pas de programme en mémoire ? **Bootstrap** (ou Boot)

« boot » apocope de « bootstrap », languette de chaussure

Auteur inconnu mais origine identifiée : « Comment faire pour traverser un lac sans se mouiller ? Réponse : tirez suffisamment fort sur les languettes de vos bottes vers le haut pour marcher sur l'eau sans couler. »

Extrait de : *Les aventures du Baron de Münchhausen*, Erich Raspe, 1785 (« *to pull oneself up by one's bootstrap* » pour se sortir des sables mouvants)

Boot : principe

Le matériel est câblé pour charger le chargeur d'amorçage à localisation connue et fixe qui permet d'extraire un programme accessible via un périphérique de stockage permanent ou amovible.

Ce dernier est typiquement le noyau du système d'exploitation, qui s'installera en RAM (protégée) et appellera lui-même des programmes applicatifs dont l'interface utilisateur et les routines système de gestion.

Boot : PC de 1981 à ≈2010

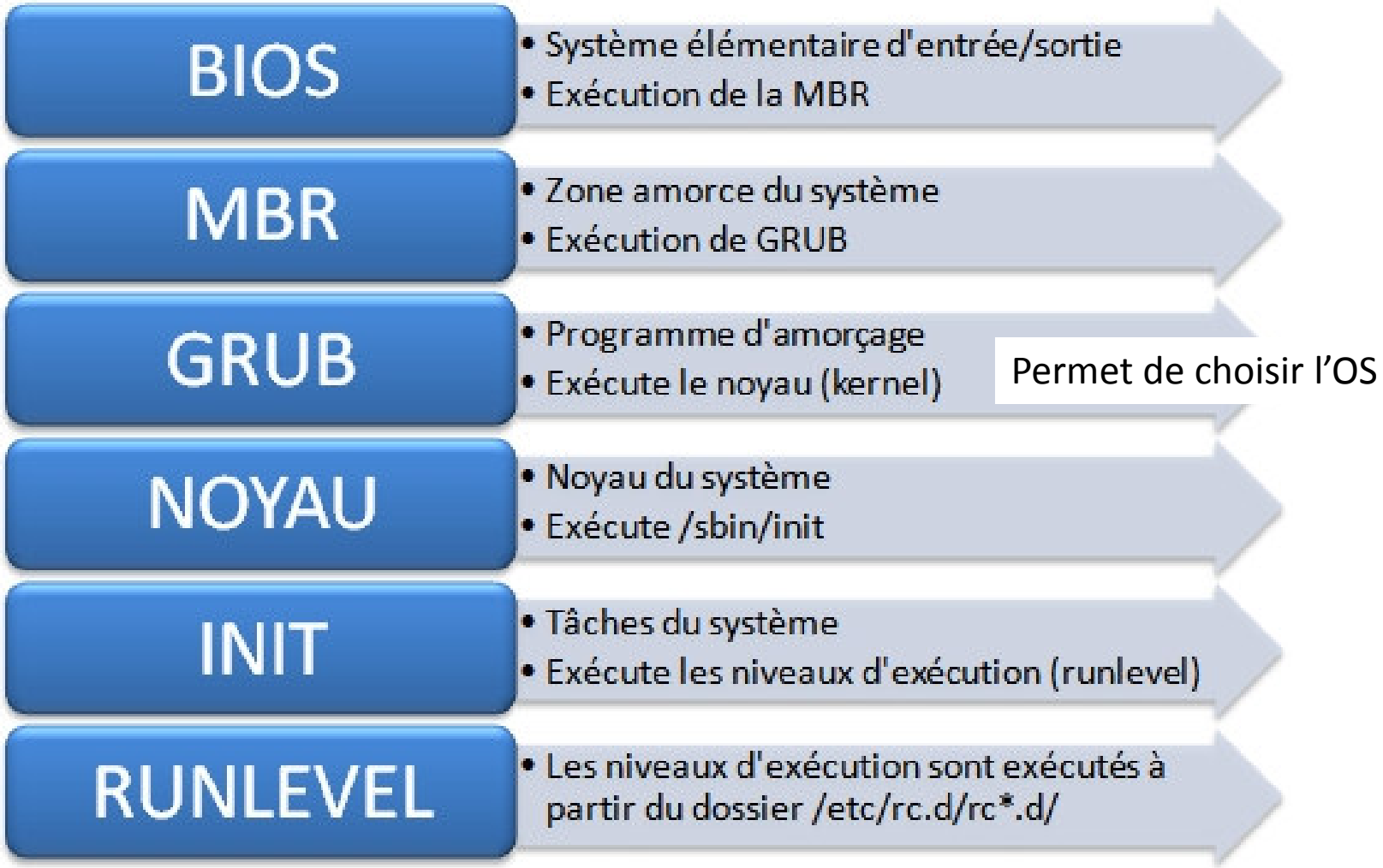
Histoire (avant) : interrupteurs, puis carte perforée

- Après reset le processeur exécute le BIOS
  - Adresse OxFFFF0, adresse de saut pour test POST
- Le BIOS charge le MBR (Master Boot Record)
  - 512 octets : Boot sector DD, disquette, CD, Flash…
- Le place à l'adresse 0X7C00 et l'exécute
- Ce code lance la chargement du reste
  - Problème : définitions limite mémoire à 2To
- BIOS → UEFI

Avant le Boot sur PC

- Mise sous tension : alimentation délivre un signal «Power-good» (PG) [ou «Power-OK» (PW-OK)]
  - L'alim met le signal PG à 0V si elle ne peut continuer à fournir des tensions correctes : courant trop fort, court-circuit, température, disparition source …
- La carte mère provoque le RESET du/des CPU
  - Le front montant de PG a exactement le même effet que lorsqu'on agit sur le bouton reset du PC.
- Un processeur lance la séquence de démarrage à la première adresse du BIOS

Boot LINUX Ubuntu



Boot en fonction de l'OS

- MS-DOS
  - IO.SYS et MSDOS.SYS : chargement DOS
  - DOS : lit CONFIG.SYS, charge COMMAND.COM, exécute AUTOEXEC.BAT
- LINUX
  - LILO (Lilo peut prendre la place du MBR) ou GRUB
- WIN 95/98
  - IO.SYS et WINBOOT.SYS
- WIN NT/2000
  - NTLOADER (fichier caché \ntldr puis `boot.ini …)

Boot sur petites machines

Machines autonomes spécialisées

- OS simple ou applicatif en ROM lancé au boot
  - Calculatrice,
  - Electro-ménager
  - Petit-objets
  - Jouets
  - …

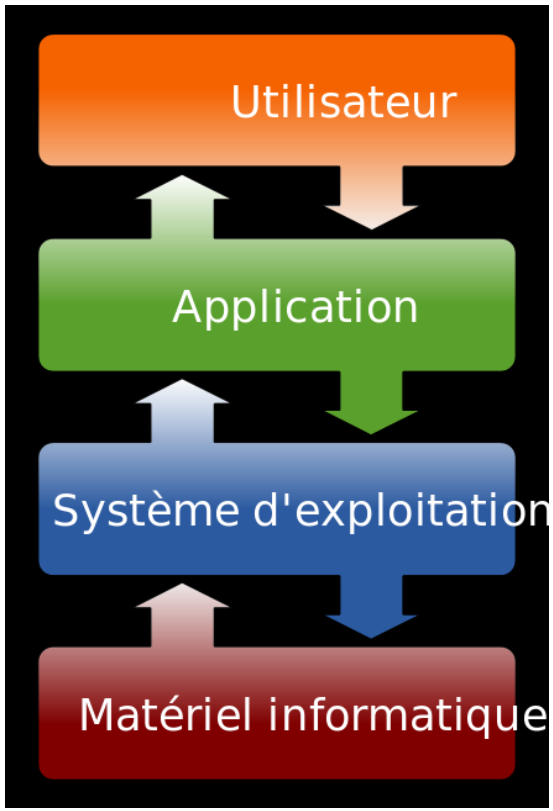
Système d’exploitation comme dernière couche de l’architecture

Système d’exploitation : définition

- Ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs.
- Il est entre le firmware (logiciel intégré au matériel) et les applications
- Au démarrage de l’ordinateur
  - Mise sous tension et séquence de reset
  - Séquence de boot (mémoire ROM, DD, Flash…)
  - Chargement et démarrage de l’OS

Système d’exploitation : services

- Offre un ensemble de service généraux et génériques (indépendants de l’architecture)



Norme IEEE-POSIX
Définition des appels standard

Schéma simplifié
des « couches »

OS et matériel

- Planifie exécutions (dont multitâches)
  - Associe/libère processus et processeur (priorité)
  - Equilibre charge sur les CPU/Threads
  - Evite les conflits de ressources
- Réserve mémoire (mémoire virtuelle/privilèges)
- Gère certaines communications interprocessus
- Gère les communications avec l'extérieur (drivers)
- Gère les processus système (non applicatifs)
- Assure la sécurité et la confidentialité

Pilotes (drivers) de périphériques

- Interface périphérique (hardware) et OS
  - Services uniformes, génériques et de haut-niveau
- Pour l'utilisateur : font partie de l'OS
- Stratégies
  - Accès direct en I/O (polling)
    - Rapide
    - Charge l'OS
  - Accès en interruption : demande et oubli
    - Plus lent
    - OS libéré

OS : Niveaux de privilège

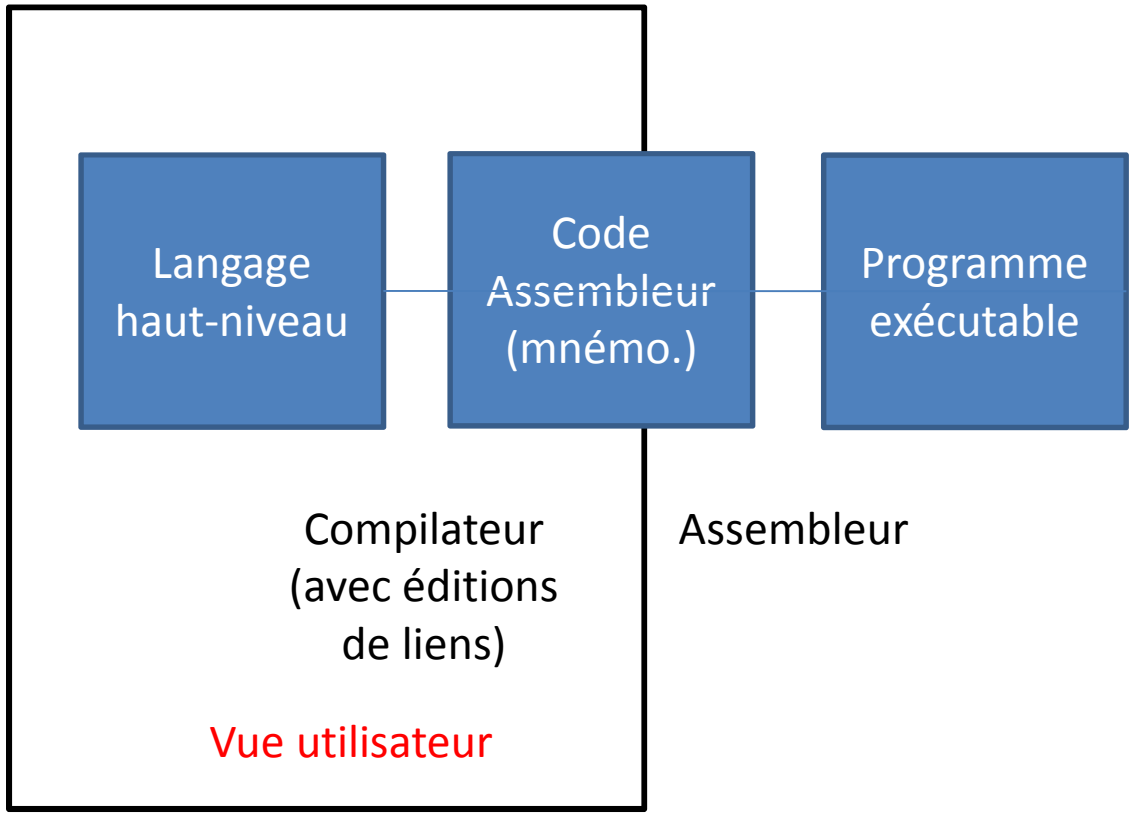
- Permet de définir des droits
  - Accès à des ressources
- Mode noyau
  - Accès mémoire réelle et périphériques
- Mode utilisateur
  - Accès mémoire virtuelle et appels systèmes
- Possibilités de niveaux intermédiaires
  - PC : 4 niveaux
  - Honeywell 6180 (histoire) : 8 niveaux

OS : appels système

- Routines système = routines d'interruption
- Appel système
  - Interruption logicielle
  - Traitement de l'interruption
    - Passage en mode noyau avec vérification
    - Très (très) lente
  - Traitement de la demande
  - Retour à l'application en mode utilisateur
- L'interface OS/Application s'appelle l'API

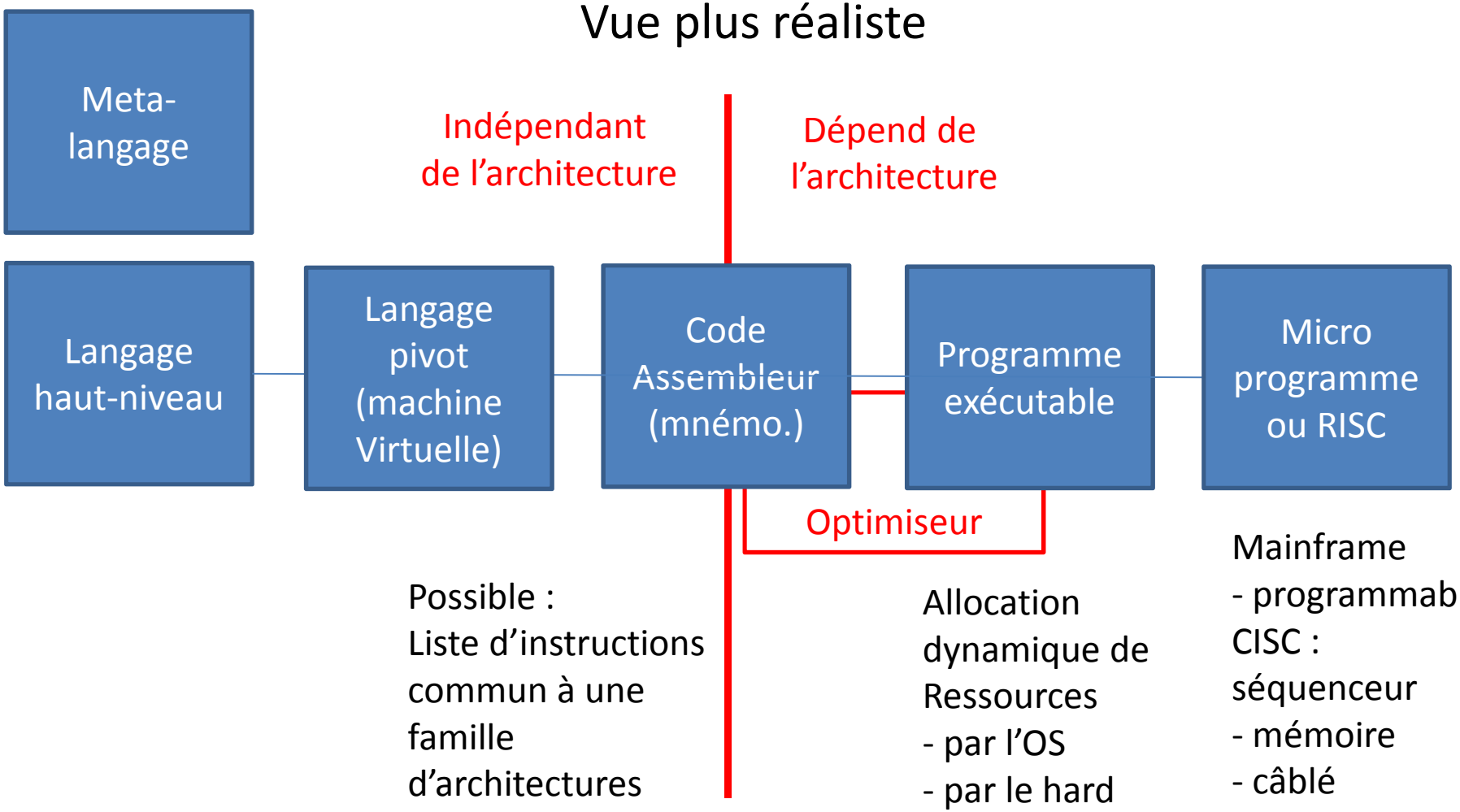
Du besoin à l'exécutable

Vue simplifiée

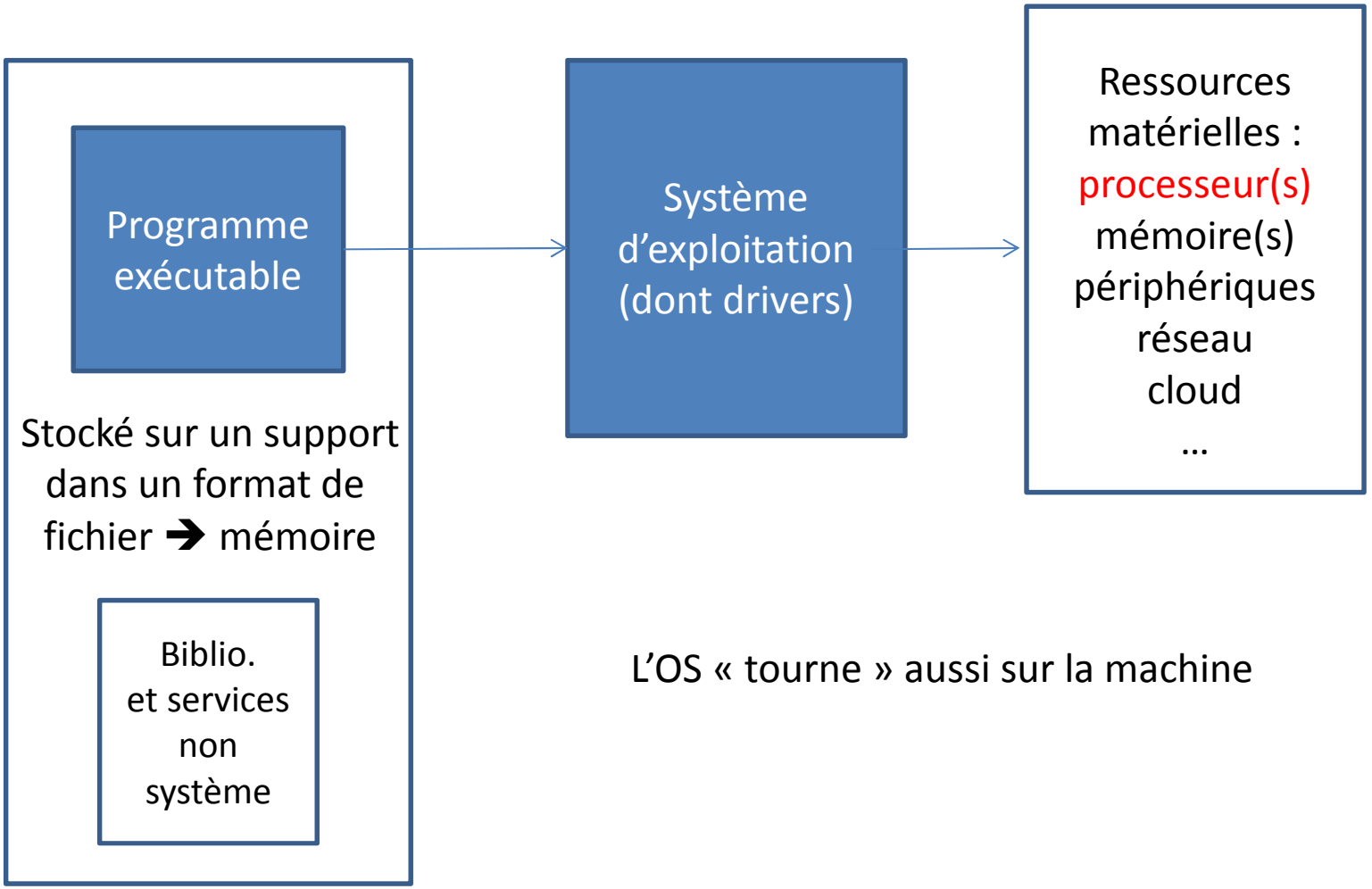


Du besoin à l'exécutable

Vue plus réaliste



De l'exécutable à l'exécution



L'OS « tourne » aussi sur la machine

Applications et architectures

Application et architecture(s)

- Efficacité : écrire en assembleur (ou C)
  - Faisable sur machines simples : MCU par exemple
  - Ingérable sur processeurs modernes
- Il faut aider les compilateurs/optimiseurs
- Deux cas
  - Assembleur de « haut-niveau » : C
  - Méta langages : JAVA

Complexité

Faire le lien entre

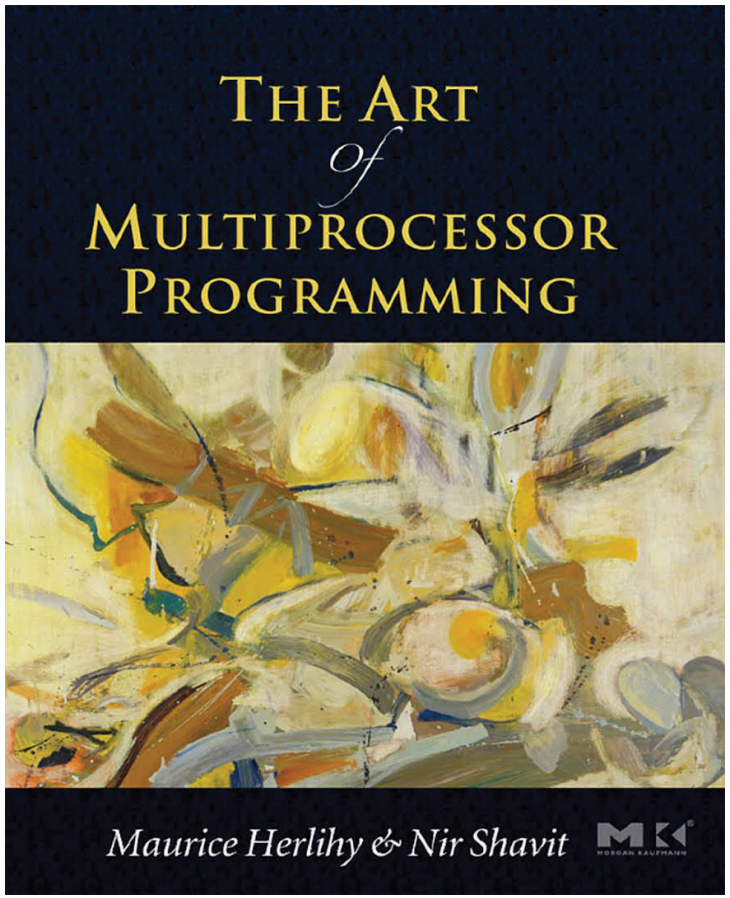
- une expression de (très) haut niveau
- et le niveau harware (dont algorithmes enfouis)
  - Out-of-order exécution
  - Scheduler
  - Cohérence cache(s)
- pour des architectures variables/inconnues

Il existe néanmoins des notions générales

Conseil de lecture

« The Art of Multiprocessor Programming »

Herlihy et Shavit



Etapes : appli ➔ exécution

- Compilateur
  - Traduire avec le jeu d'instruction de la cible
  - Possible étape intermédiaire sur machine virtuelle
  - Séparation CPU/GPU (explicite/implicite)
- Optimiseur
  - Réorganiser le code pour architecture interne
  - Peut se faire à la volée par le hard
- Ordonnanceur (scheduler)
  - Distribution des threads sur les cœurs
    - Niveau OS : association fixe ou variable
    - Niveau interne

Limites du multicore/SMT

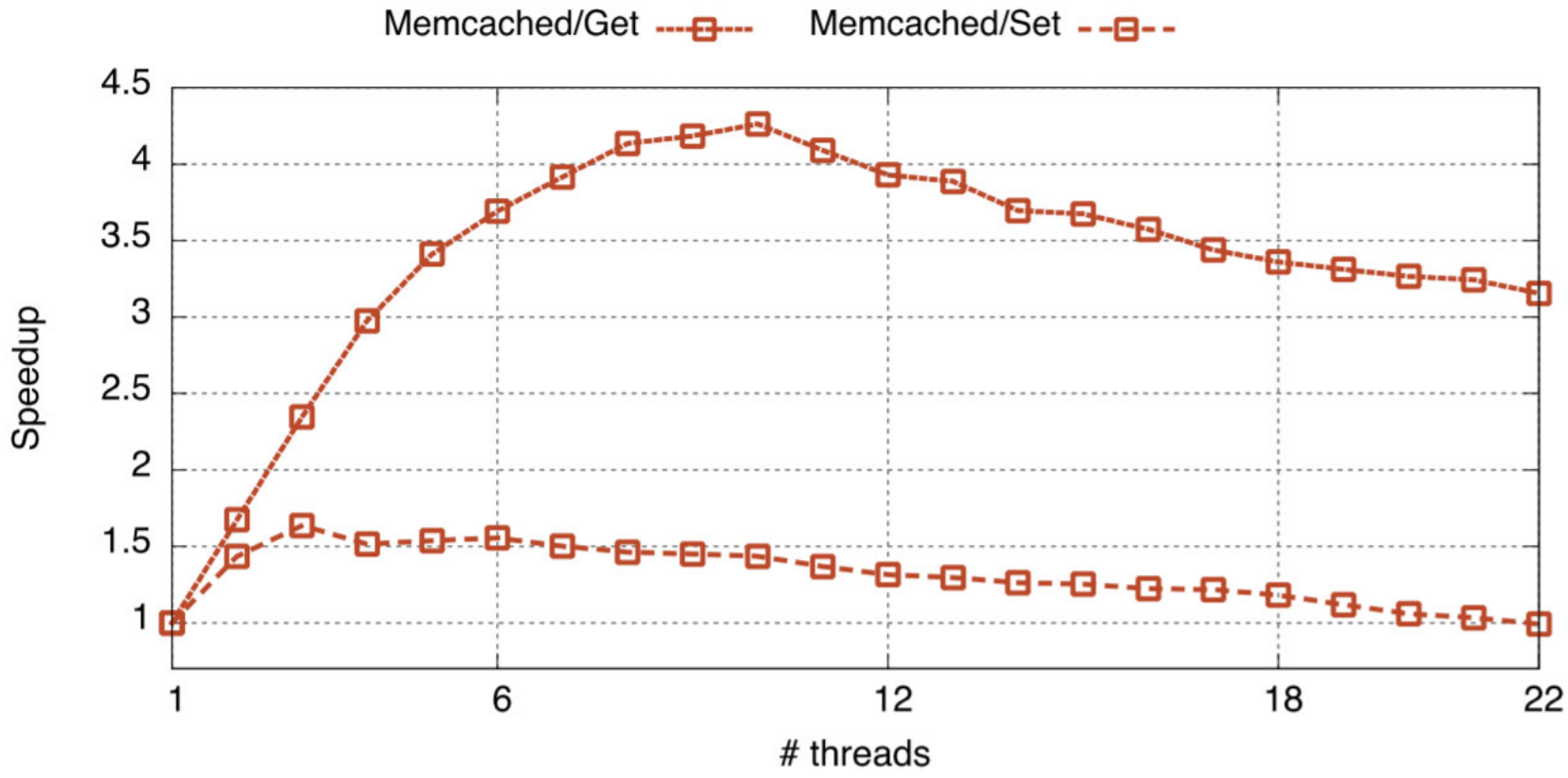
- Application/algorithmes non parallélisables
  - Ou pas assez parallélisables
- Loi d'Amdahl (1967)
  - Application générale
    - P% non parallélisable (souvent nommé overhead)
    - 1-P% parfaitement parallélisable (exé. tend vers 0)
      ➔ accélération maximale : a = 1/P (borne)
      - P =50% : a=2
      - P=10% : a = 10 (proche de la réalité)
      - P =1% : a =100
  - POUR UN NOMBRE INFINI DE PROCESSEURS

Loi d'Amdahl réelle

Notion de 'scalabilité'

- Gestion ↗ ➔ performances ↘

Deux courbes classiques de SpeedUp :e fonction de nombre de threads
On augmente jusqu'à un optimum puis on diminue

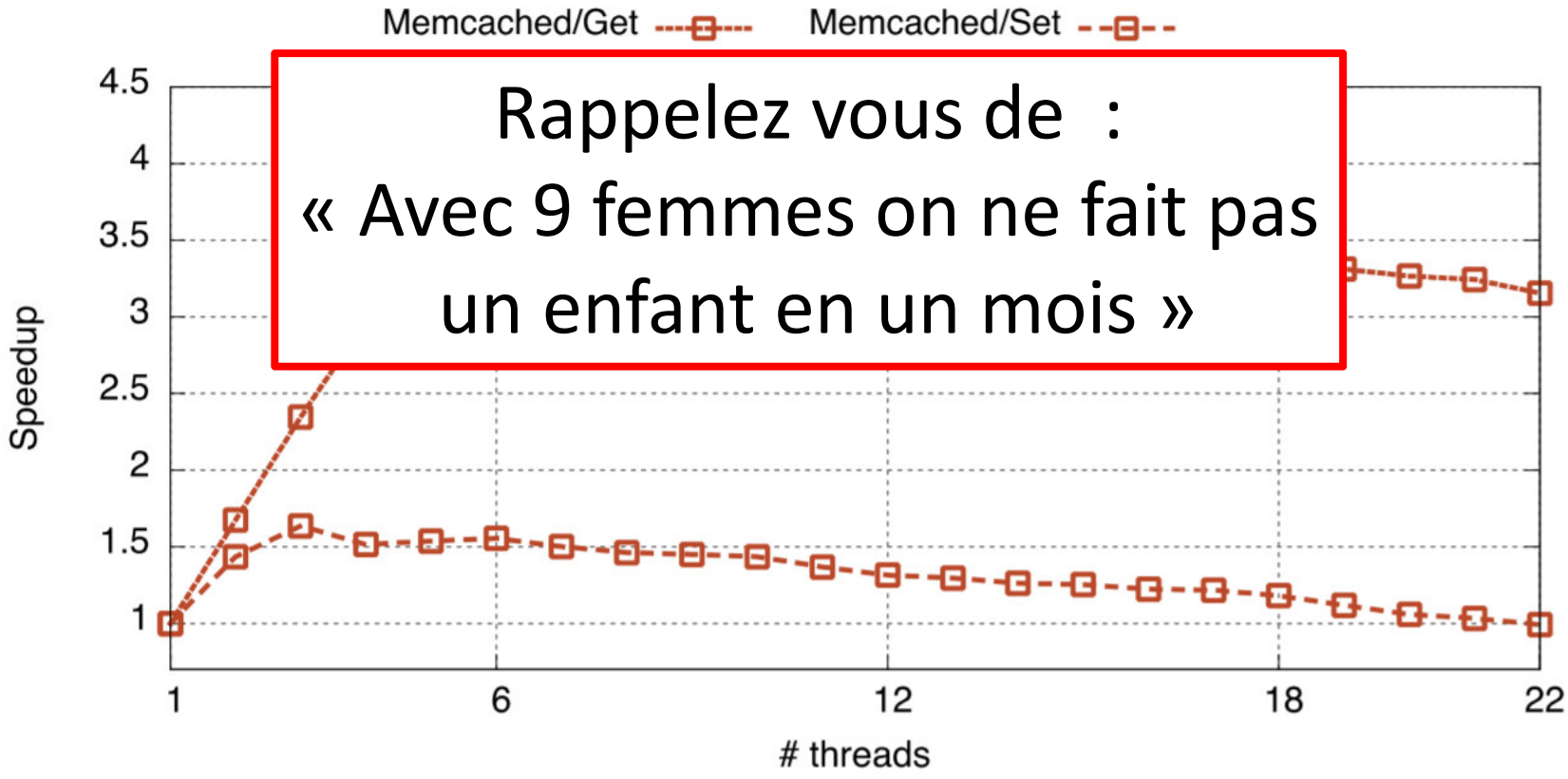


Loi d'Amdahl réelle

Notion de 'scalabilité'

- Gestion ↗ ➔ performances ↘

Deux courbes classiques de SpeedUp :e fonction de nombre de threads
On augmente jusqu'à un optimum puis on diminue



Rappelez vous de :
« Avec 9 femmes on ne fait pas un enfant en un mois »

Autres difficultés

Complexité des architectures

- Difficulté d'un emploi optimal (monitoring complexe)
- Pire : pas de moyen de savoir si c'est le cas

Compilateur/optimiseurs

- Difficulté de suivre le « rythme »

Innovation : Le matériel va plus vite que le logiciel

Loi de Wirth ➔Obésiciel

applis grossissent

vitesse appli croissent moins vite que celle du matériel

Encore des difficultés

Pour le multiprocesseur « large »

- Nœuds/modèle mémoire hétérogènes
- Insertion à chaud
  - Nœuds de calcul
  - Périphérie
- Réseaux hétérogènes/saturés/bloqués
- Pannes locales
  - Déconnexion temporaire non planifiée proc/périphs

Conclusions locales

- On est condamné pour l'instant à faire de la sous-utilisation des ressources de calcul
- Avec toutes les conséquences qui en découlent
  - Consommation (électriques) inutiles
    - ➔ impact sur les ressources
  - Prix trop élevés
  - Ressources de calcul surdimensionnées
  - Applications qui grossissent
  - Frustration des utilisateurs
- Beaucoup de travaux de recherche en cours

Edifiant, à lire

“THE LINUX SCHEDULER: A DECADE OF WASTED CORES”

Applications entre 4 et 137 fois plus lente queleur vitesse potentielle (2016)

http://www.i3s.unice.fr/~jplozi/prog_concurrente_2016/cours/linux_wasted_cores_15Xo2FHG.pdf

Bac à sable

Types d'optimisation

- Optimisation algorithmique
  - Gérer la complexité algorithmique
- Optimisation architecturale
  - Constantes plutôt que variable (bloc en cache)
  - 'Inline' plutôt que fonctions (switch contexte)
  - Fonctions plutôt que procédures (effets de bord)
  - Limiter dépendances (ex : duplication de données)
  - Localité des données (petits tableaux → cache)
  - Déclarer des threads / optimiser verrous
  - Utiliser algorithmes non-bloquants (sans verrous)

Algorithmes non-bloquants

Trois classes d'algorithmes sans verrou

wait-free ⇒ lock-free ⇒ obstruction-free

- Wait-free : Toute opération se termine en un nombre fini de pas. Famine impossible.
- Lock-free : Certains threads progressent tout le temps, pas forcément tous. Famine possible
- Obstruction-free : Si on arrête les autres threads qui peuvent nous ralentir, alors on arrive à terminer.

Algorithmes existants pour : pile, liste, liste chaînée

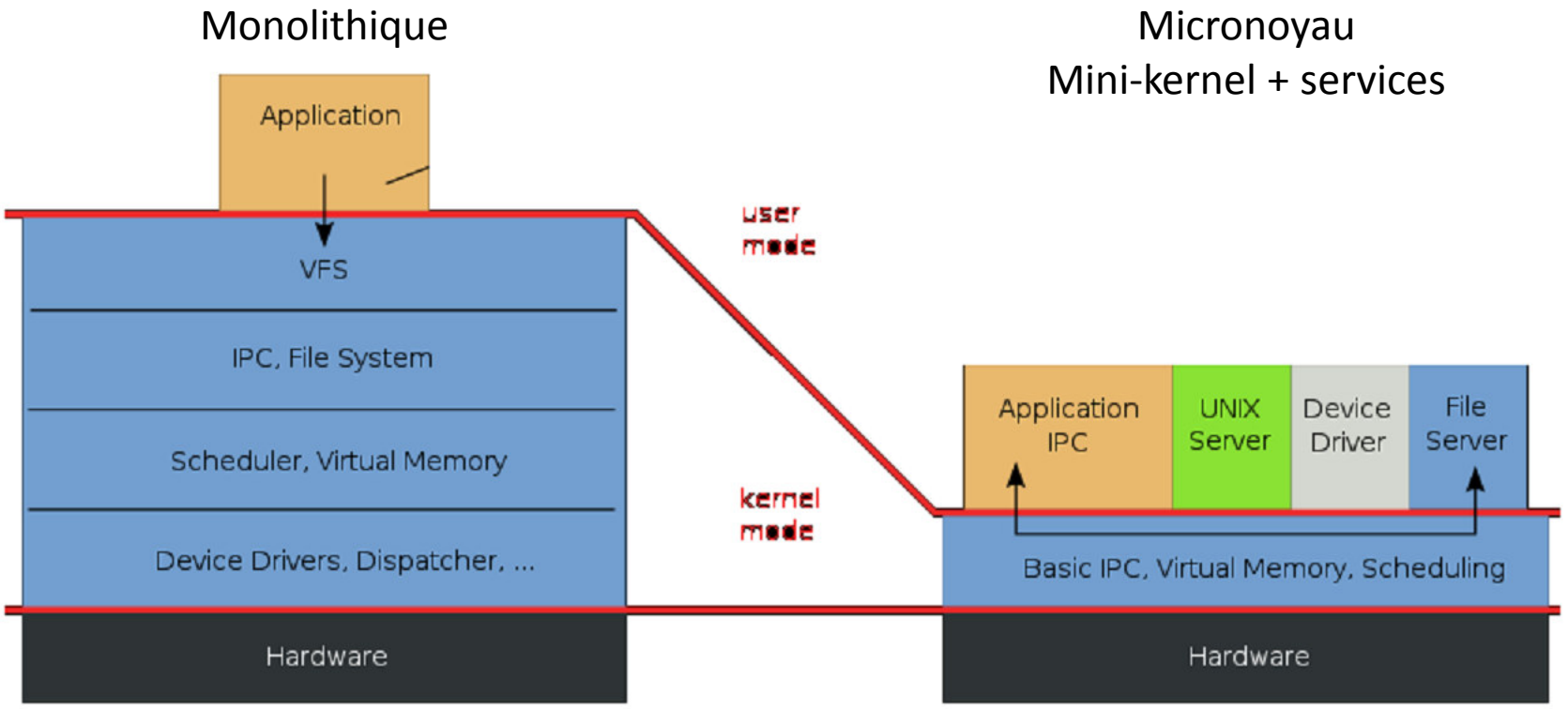
Autre méthodes
Mémoires transactionnelles

- Blocs courts et atomiques
  - Non séparables
- Transactionnels : Bloc atomique exécuté
  - Si modification de ses données pendant exécution
    - Abandon et reprise
    - Pas d'attente sur verrou
- Propriétés théoriques
  - Si A et B sont atomique alors composanbles

Pour aller (beaucoup) plus loin : http://www.i3s.unice.fr/~jplozi/prog_concurrente_2016

OS : architecture (couches)

Exécution événementielle : demande, (appel système), erreur, interruption

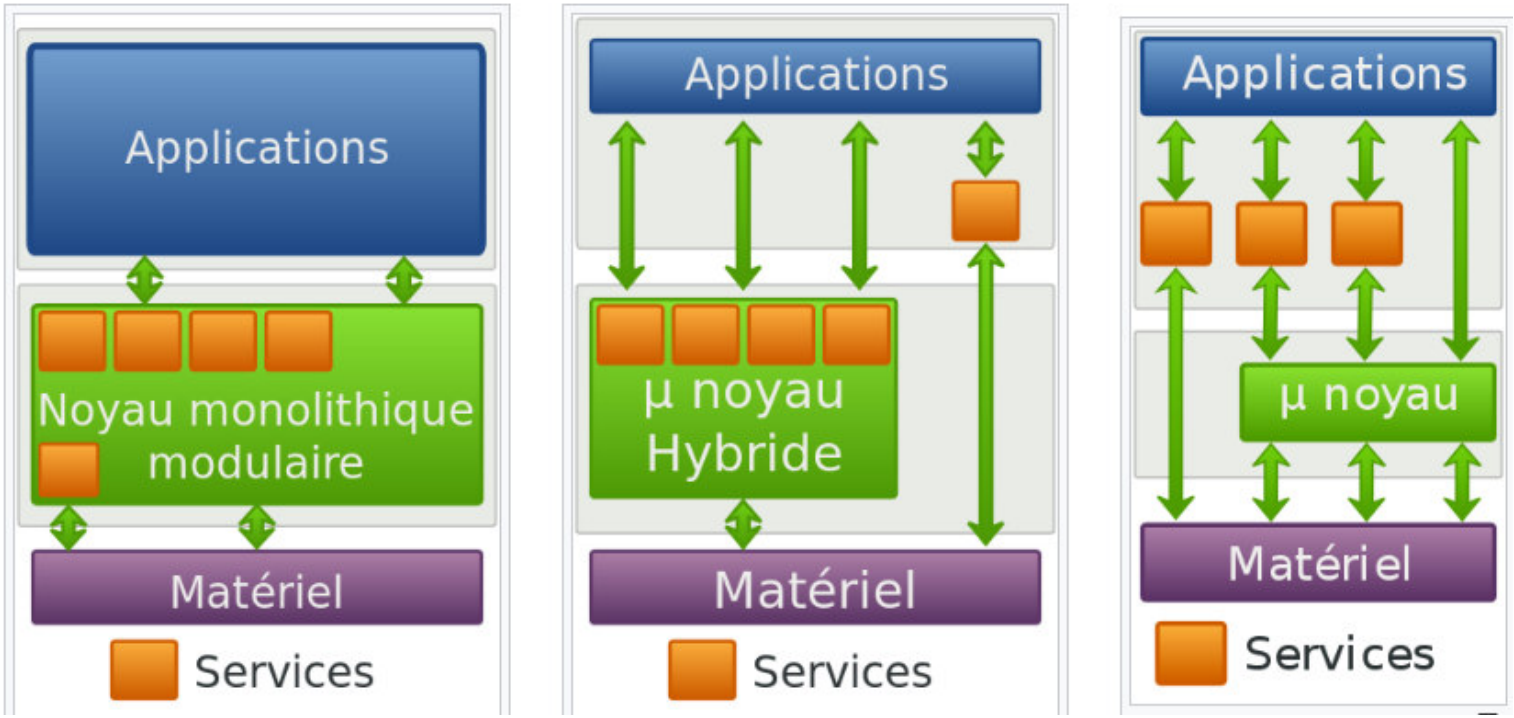


Noyau (kernel) : partie de l'OS toujours présente en mémoire après la séquence de boot (en mémoire protégée) VFS : Virtual File System / IPC : InterProcessus Comm.

Evolution multitâche

- Mono tache
- Traitement par lots (batch) – 1950
  - Chacun son tour
- Multiprogrammation -1960
  - Les attentes de résultats (d'un périphérique lent) libère le processeur pour une autres aplication
- Temps partagé (Time-slicing) – 1970
  - swap
- Evolutions : Temps-réel, systèmes distribués, Cloud

Efficacité/sécurité ?



Monolithique : rapide mais erreurs « mortelle »

Micro Noyau : plus lent, erreur service « non mortelle »

Hybride : performance selon distribution des serices

Répartition appels système