BUL : Block device in User Land
Un terrain d'expérimentation pour des stratégies de stockage distribué

- Problématique du stockage distribué
  - Différents niveaux de stockage
    $\rightarrow$ Le niveau *block device*
- Principe de BUL
  - Concept
  - Implémentation
- Et les graphes dans tout ça ?
  Quelques algorithmes innovants pour le stockage distribué

Niveau utilisateur

Ensembles de fichiers, “collections”

- Données non mutables :
  répliquées et distribuées en pair-à-pair
- Données mutables :
  la résolution de conflits est souvent manuelle
- Sémantiques d'accès et de recherche très floues
  (utilisation de moteurs de recherches généraux ou spécialisés)

Niveau application

Fichiers, “documents”, ressources (URL...)

- Transmission et réplication par des moyens très variés (réseau, support amovible, moult protocoles...)
- Réplication possible au niveau applicatif, selon le type de données (exemples : NNTP, proxy HTTP, AFS, Coda...)
- Sémantiques d’accès dépendant du protocole / de l’application
- Cas “Maildir” : adaptation d’un format à l’accès concurrent

Niveau fichier

Fichier = suite d'octets non structurée

- Réplication facile sur des données non mutables (encore !)
- Accès concurrent et répliqué possible avec un couplage fort
  (par utilisation de jeton, par exemple)
  (pas d'implémentation satisfaisante à ce jour - Intermezzo)
- Sémantiques d'accès très bien définies (POSIX)
  Mais parfois insuffisantes !
  → développement de NTFS, ReiserFS

Niveau bloc

Périphérique bloc = suite d'octets (encore !)
(exemple : disque dur, disquette, CD-ROM...)

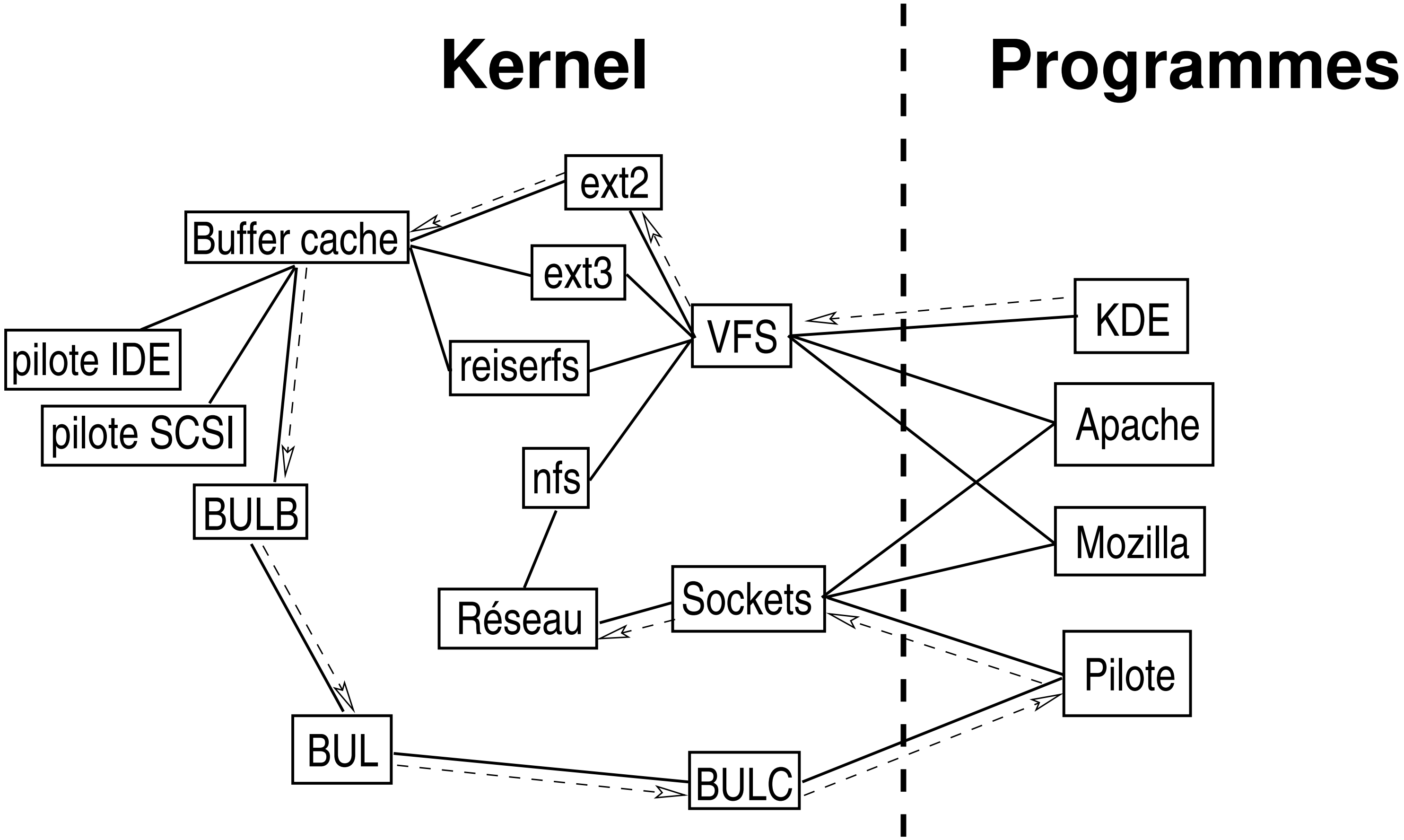
- Sémantiques encore plus simples
  (quasiment pas de création ou changement de taille)
- Accès concurrent impossible
  (sauf si la couche supérieure y est adaptée : GFS...)
- Réplication et distribution : matériels et protocoles dédiés
  (Fibre Channel, iSCSI...)
- Solutions logicielles non satisfaisantes
  (NBD, ENBD)
  $\rightarrow$ ouverture potentielle !

BUL : principe “théorique”

- Intérêt d’un protocole similaire à SCSI, mais :
  sur un réseau plus répandu, moins coûteux, plus performant ?
  → Utilisation d’Ethernet et IP
- Développement système (kernel) difficile
  → Système permettant de déférer les requêtes
  à un programme “normal” en “userland”
- Performances moins bonnes, mais plus grande souplesse
- Utilisation d’autres langages que le C
- Nouvelle classe d’algorithmes dans le domaine du stockage

BUL : implémentation

- Développement sous Linux (choix logique)
- Écriture d'un **petit** *driver*
- Deux facettes : le *block device*
  - Se comporte exactement comme un disque “normal”
  - Les requêtes sont placées en attente dans des “cookies”, et transmises au ...
- ... *control device*
  - Périphérique en mode “caractère” (canal bidirectionnel, orienté “message”)
  - Le “pilote” (programme en “userland”) lit et écrit dessus (transmission des requêtes et de leurs réponses)

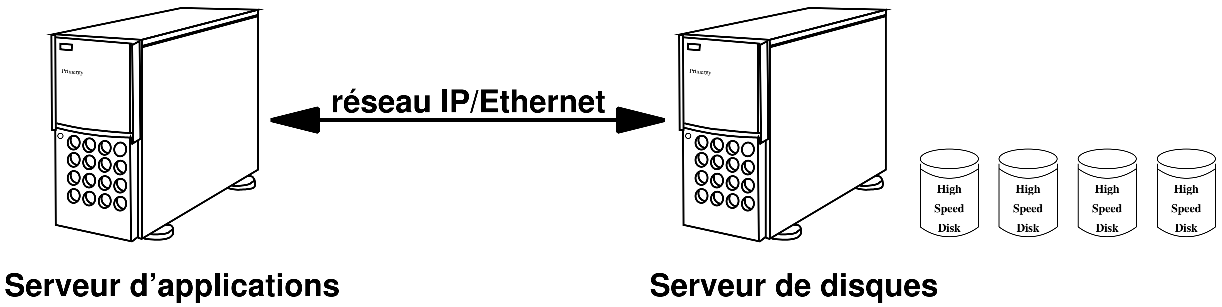

Kernel
Programmes
ext2
Buffer cache
ext3
VFS
KDE
pilote IDE
reiserfs
Apache
pilote SCSI
nfs
BULB
Mozilla
Sockets
Réseau
Pilote
BUL
BULC

Algorithme RAID5 de base

- Redondant avec le RAID5 logiciel de Linux
- Peut servir de base pour des tests de performance
- Un intérêt : possibilité d'agrandir "en ligne" le volume
- Autre intérêt : utilisation d'un "journal" pour les transactions au niveau bloc
  (pour éviter les délais de reconstruction en cas de crash)

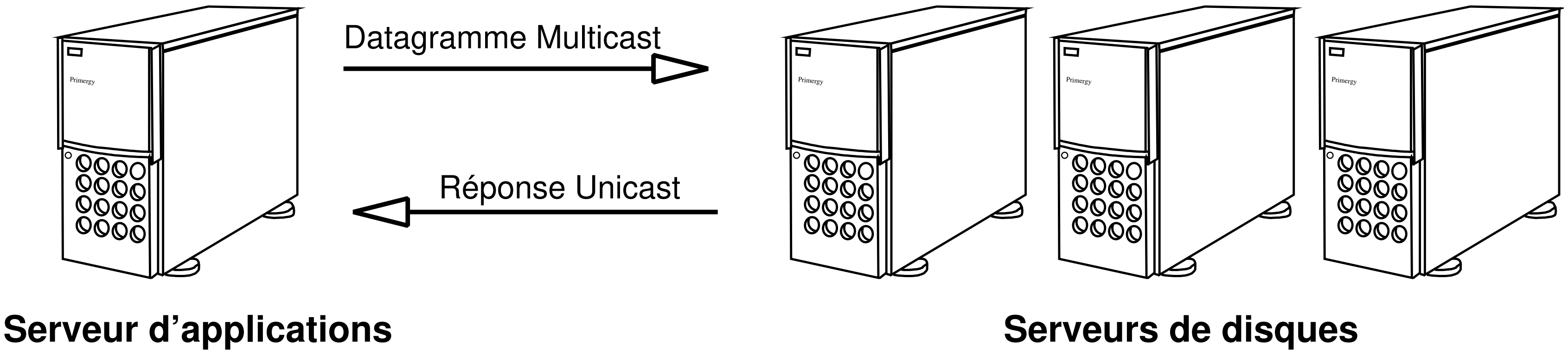
Un peu de réseau (enfin !)

- Déporter un disque vers une autre machine (similaire à NBD, ENBD, iSCSI, FC)
- Protocole non connecté (UDP), stateless (bonne résistance aux pertes)
- Les calculs liés au RAID5 (par exemple) sont déportés (sommes de contrôle, reconstruction de la parité...)



Plus intéressant : Multicast

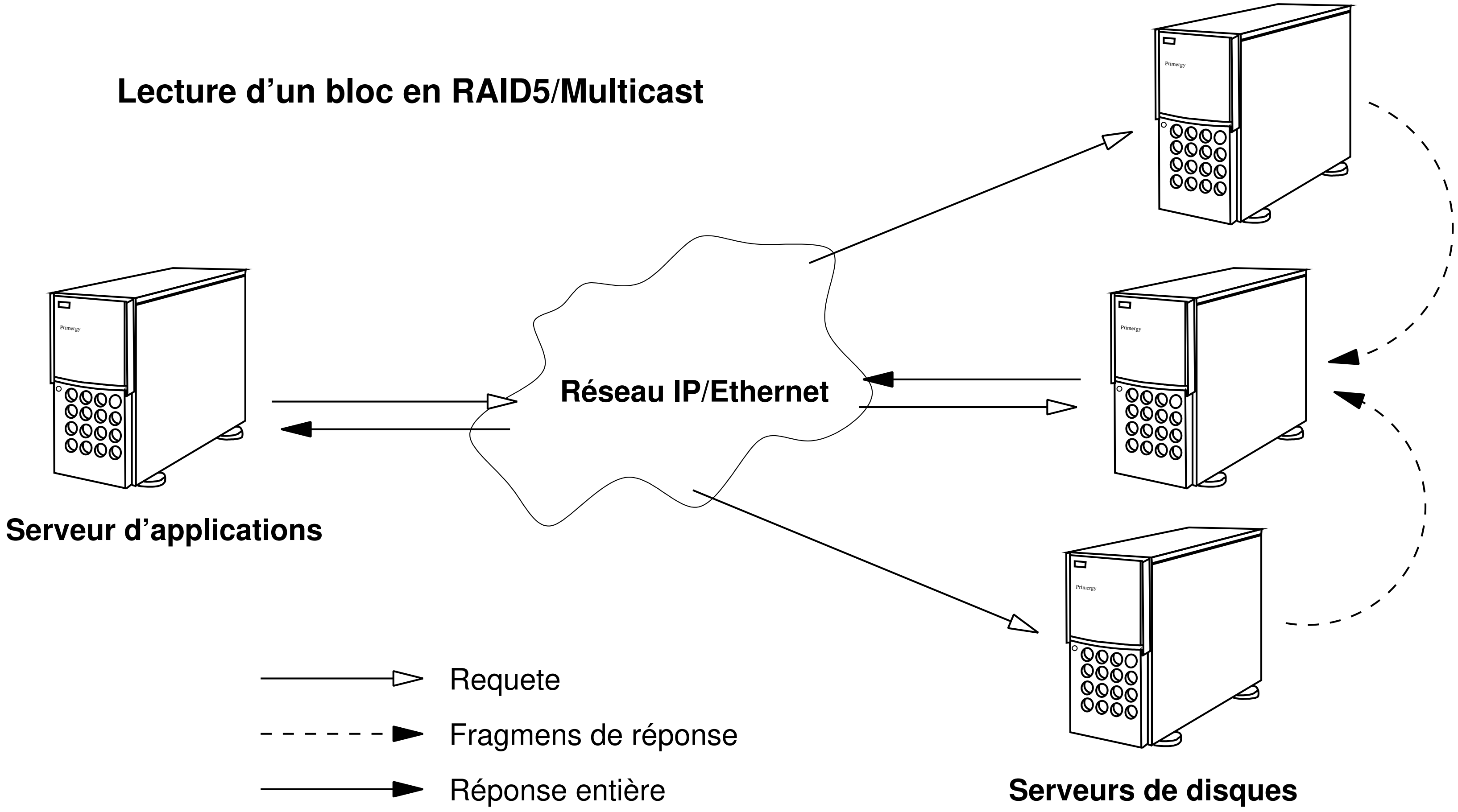
- Répartition de données entre plusieurs machines
- La machine “utilisatrice” n’a pas besoin de savoir où se trouvent les données
- Amélioration des performances, de la fiabilité ... sans augmenter la complexité du système



RAID5 distribué

- Combinaison du multicast et du RAID5
- le "pilote" envoie une demande multicast pour un bloc
- Toutes les machines de stockage reçoivent la requête,
  et envoient leur fragment vers une machine "reconstructrice"
  (déterminée par modulo)
- La machine "reconstructrice" rétablit le bloc de données
  et l'envoie au "pilote"
- En cas de défaillance, la régénération d'un disque
  peut se faire de façon indépendante
  $\rightarrow$ sans impacter la machine "pilote"

Lecture d'un bloc en RAID5/Multicast



Redimensionnement des volumes

- Problématique classique dans un volume RAID5 :
  que fait-on quand c'est plein ?
  - Solution classique : on ajoute un deuxième volume
  - Autre "solution" : on casse tout, et on recommence
    (mais avec plus de disques, et/ou plus gros !)
    → pas satisfaisant !
- Redimensionnement à la volée de façon transparente
- Ajout de N nouveaux disques (facile)
- Ajout de 1 nouveau disque
  → reconstruction progressive délicate, mais possible

Analyse de la charge d'un disque réseau

- Utilisation du multicast
  → on dispose d'une “trace” de tous les accès
- Possibilité de détecter les blocs les plus souvent accédés
- Migration (toujours transparente) de ces blocs
  vers des disques rapides
- Marquage de données “importantes” (métadonnées...)
  → affectation d'un meilleur niveau de redondance
- Rendre systématique l'analyse du profil des accès
  → meilleures optimisations

Répartition “souple” des données

- Utilisation d’une temporisation :
  le serveur de disque le plus rapide répond à la requête
- Plus un serveur est chargé, moins il est rapide
  $\rightarrow$ les données se répartissent automatiquement
- Intérêts :
  - Hétérogénéité possible des ressources
  - Intégration d’une nouvelle ressource sans “tout casser”
- Problème : tenir à jour un “suivi” de redondance
- Ne pas prendre ce système pour un système pair-à-pair !

Conclusion

- Principe très simple
  $\rightarrow$ Espoir raisonnable d'aboutir à un "pilote" propre, de qualité "industrielle" (intégrable dans Linux)
- Terrain d'expérimentation et de prototypage
  - Codes correcteurs : Hamming, Reed Solomon...
  - Répartition optimale des données
  - Passage à l'échelle
- Intérêt théorique discutable, certes :-)