Big data et traitement de données parallèles

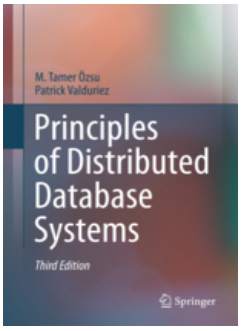
Patrick Valduriez
INRIA, Montpellier

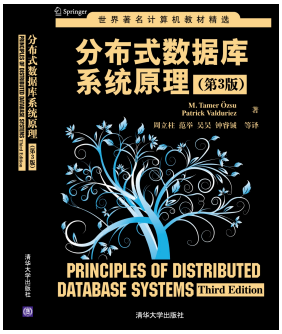
Plan du cours

1. Le big data
2. Objectifs des systèmes parallèles
3. Architectures parallèles
4. Principales techniques
5. Etude de cas: Google Search
6. Machines bases de données

Bibliographie

Principles of Distributed Database Systems
Tamer Özsu & Patrick Valduriez
Springer, 850 pages, 2011.





1. Big data: qu'est ce que c'est?

- Un buzz word!
  - Avec différentes interprétations en fonction de notre perspective
    - Ex. 10 teraoctets est bcp pour un système transactionnel, mais peu pour un moteur de recherche du web
- Une définition (Wikipedia)
  - Ensembles de données qui deviennent si grands qu'il devient difficile de les gérer avec les outils de gestion de données classiques
    - Difficultés: capture, stockage, recherche, partage, analyse, visualisation
  - *Mais la taille est une des dimensions du problème*
- Que veut dire *big*?
  - Cible mouvante: teraoctet (10^{12} octets), petaoctet (10^{15}), exaoctet (10^{18}), zetaoctet (10^{21})
  - Points de repère SGBD
    - 1980: Teradata database machine
    - 2010: Oracle Exadata database machine

Pourquoi le big data aujourd'hui?

- Très grandes quantités de données produites par toutes sortes d'appareils, réseaux et programmes
  - Ex. Capteurs, appareils mobiles, internet, réseaux sociaux, simulations, satellites, radiotélescopes, etc.
- Capacité de stockage
  - A doublé tous les 3 ans depuis 1980 avec les prix en baisse forte et constante:
    - 1 Gigaoctet pour: 1M$ en 1982, 1K$ en 1995, 0.12$ en 2011
- Très utile dans un monde numérique!
  - Peut produire information et connaissance à forte valeur ajoutée
  - Crtitique pour l'analyse, l'aide à la décision, la prévision, le BI, la recherche, la science, etc.

Quelques chiffres

- Big market
  - $18 milliard en 2013, $24 milliard en 2016
  - Source: International Data Corp. (IDC)
- Estimations*
  - 1,8 zetaoctets: la taille des données stockée par l'humanité en 2011
  - 40 zetaoctets en 2020
  - Mais
    - Moins de 1% est analysée
    - Moins de 20% est protégée

* Source: Digital Universe study of IDC, 2012

Dimensions: les cinq *big V's*

1. Volume
   - Grandes quantités de données
   - Rend difficile le stockage et la gestion, mais aussi l'analyse(big analytics)
2. Vélocité
   - Flux continus de données provenant de capteurs, ou d'appareils mobiles
   - Rend difficile l'analyse en ligne
3. Variété
   - Différents formats de données (séquences, graphes, tableaux, …), différentes sémantiques, données incertaines, données multi-échelles (temporelle, spatiale, …)
   - Rend difficile l'intégration et l'analyse
4. Validité
   - Est ce que les données sont correctes et précises
5. Véracité
   - Est ce que les résultats sont significatifs?

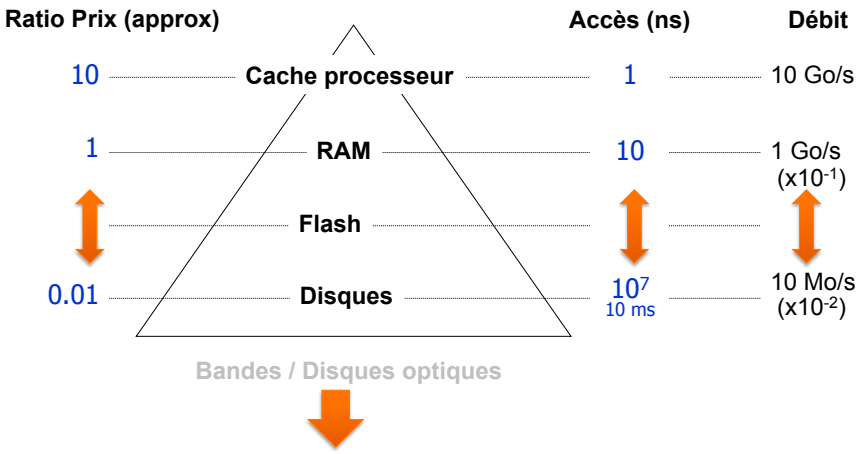
Enjeux pour l'entreprise

- Production d'informations en temps réel à partir de données distribuées
- Croisement de données publiques et privées
- Visualisation des données croisées: Dataviz
- Analyses complexes sur big data: Big Analytics
- Transactions sur des données en réseau
- Réactivité : traitement de flux de données en temps réel, Complex Event Processing (CEP)

Impact des progrès matériels

- Très grandes mémoires
  - Vers 1 Teraoctet de RAM sur 1 chip (Crossbar)
- Stockage sur disque
  - Solid State Disk (SSD), basé sur mémoire flash
    - Plus rapide (facteur 4-10) et moins consommateur en énergie que le disque HDD
- Multiprocessing
  - Processeurs multi-cœurs
  - Combinaison CPU/GPU
- Réseaux haut-débit, extensibles
  - Architectures arborescentes à base de switches
    - Ex. Infiniband jusqu'à 100 megabits/s (Mellanox)
- Virtualisation
  - Serveurs
    - Support des VM par hardware
  - Stockage
    - Storage Area Network (SAN)

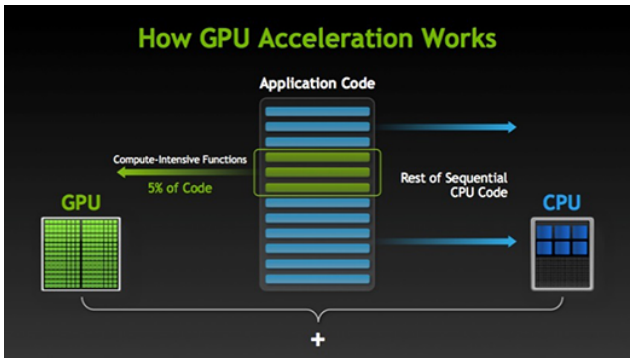
Nouvelle hiérarchie de mémoire



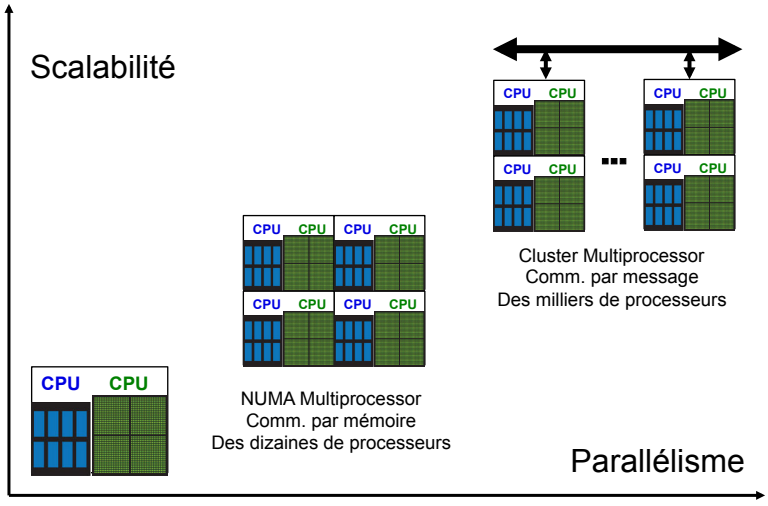
Tape is Dead, Disk is Tape, Flash is Disk, RAM Locality is King
Jim Gray (ACM Turing Award 1998)

Accélération avec CPU/GPU

- CPU
  - Des dizaine de cœurs, optimisé pour le traitement séquentiel
- GPU
  - Des milliers de cœurs (plus simples), optimisé pour le multi-tâche et les gros calculs



Nouvelle hiérarchie de calcul



Opportunités

- Exploitation du big data
  - Production de nouvelles informations et connaissances
- Scalabilité
  - Architectures de BD qui passent à très grande échelle, grâce au parallélisme massif
- Performances
  - Traitement de données in-memory
    - RAM, flash
  - CPU/GPU
- Cloud
  - Possibilité de faire du big data sans grand investissement d'infrastructure

Enjeux

- Complexité et coût des data centers
  - Cablage, refroidissement à eau, etc.
  - Consommation électrique
- Tolérance aux fautes
  - Avec des milliers de nœuds de calcul, la panne d'un nœud devient normale
- Confidentialité des données
  - A cause de l'intégration de sources multiples
- Sureté et confiance
  - Sous-traitance en cascade des data centers à des tiers (ex. fournisseurs de cloud)

Illustration – anecdote 1

TECH 10/07/2013 @ 9:31PM | 40 481 views

The NSA's Hugely Expensive Utah Data Center Has Major Electrical Problems And Basically Isn't Working

21 comments, 10 called-out + Comment Now + Follow Comments

Well, this is good news for those with privacy concerns about the NSA and terrible news for those concerned about government spending.

Illustration – anecdote 1

- The NSA's Hugely Expensive Utah Data Center Has Major Electrical Problems And Basically Isn't Working. *Forbes*, 2013.
- Extraits:

  Well, this is good news for those with privacy concerns about the NSA and terrible news for those concerned about government spending. The National Security Agency's new billion-dollar-plus data center in Bluffdale, Utah was supposed to go online in September, but the Wall Street Journal's Siobhan Gorman reports that it has major electrical problems and that the facility known as "the country's biggest spy center" is presently nearly unusable.

  …..

  "The problem, and we all know it, is that they put the appliances too close together," a person familar with the database construction told FORBES, describing the arcs as creating "kill zones." "They used wiring that's not adequate to the task. We all talked about the fact that it wasn't going to work."

Illustration – anecdote 2

ACTUALITES

Arrêt de Chorus : les raisons de la panne du datacenter de Bull

Reynald Fléchaux
Publié: 27 juin 2013

Share

Vendredi dernier, LeMagIT révélait la panne du progiciel comptable de l'Etat Chorus, suite à un incident d'exploitation chez l'hébergeur de l'application, Bull. Des faits confirmés depuis par l'AIFE, l'agence de l'Etat qui pilote les développements et la maintenance de Chorus. Nous revenons ici sur le déroulement de cet incident, que nous avons pu recouper via différentes sources.

Que s'est-il passé dans la salle du datacenter de Bull à Trélazé mercredi 19 juin ?

Selon nos sources, ce que Bull qualifie d'incident d'exploitation trouve son origine dans le

Illustration – anecdote 2

- Arrêt de Chorus (progiciel comptable de l'Etat) : les raisons de la panne du datacenter de Bull. *LeMagIT*, juin 2013.
- Extraits

> Cet incident trouve son origine dans le déclenchement intempestif des systèmes anti-incendie d'une des salles du datacenter, suite à une erreur dans l'intervention d'un sous-traitant qui a provoqué une réaction en chaîne. Dans cette salle, ces systèmes sont basés sur l'envoi de gaz à haute pression censé étouffer les flammes. En sortant des buses de diffusion à ces pressions, le gaz crée un bruit très fort, entraînant des vibrations. Ce sont ces dernières qui sont préjudiciables aux disques durs que renferment les baies de stockage, les équipements affectés par l'incident de mercredi dernier. Les vibrations provoquées par l'onde sonore peuvent en effet entraîner l'arrêt du disque, la destruction des têtes de lecture, voire la destruction totale du support. Ici, la panne de Chorus s'explique par la défaillance de 3 disques dans une baie de stockage

Solutions pour le big data

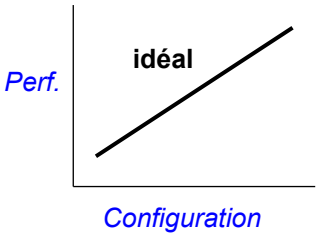
- Principe: exploiter le parallélisme des multiprocesseurs
- Approches
  - Machine base de données
    - Pour les données très structurées
  - Framework de programmation parallèle
    - MapReduce (Google, Hadoop), PigLatin (Yahoo)
  - SGBD NoSQL
    - Pour les données non structurées
    - Attention: MapReduce n'est pas un SGBD NoSQL

2. Objectifs des systèmes parallèles

- Performances grâce au parallélisme
  - Haut débit transactionnel (OLTP)
  - Bon temps de réponse des requêtes décisionnelles (OLAP)
- Haute disponibilité et fiabilité grâce à la réplication et le failover
- Extensibilité et scalabilité grâce à l'ajout de machines et ressources matérielles
  - Processeurs, mémoire, disque, réseau

Extensibilité

- Idéal: speed-up linéaire
  - Augmentation linéaire des performances en augmentant la configuration
    - Pour une charge et volumétrie des données fixes



Limites du speed-up

- Matériel/logiciel
  - Plus on ajoute de ressources, plus les conflits d'arbitrage augmentent
    - Ex. Accès au bus par les processeurs
- Application
  - Seule une partie d'un programme peut être parallélisée
  - Rappel: loi de Amdahl donnant le speed-up maximum
    - *Seq* = fraction de la partie de code non parallélisable

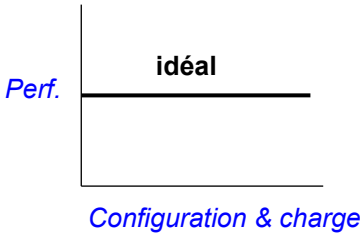
$$\frac{1}{Seq + \frac{1 - Seq}{NbProc}}$$

Exemples

- Seq=0, NbProc=4 => speed-up= 4
- Seq=30%, NbProc=4 => speed-up= 2,1
- Seq=30%, NbProc=8 => speed-up= 2,5

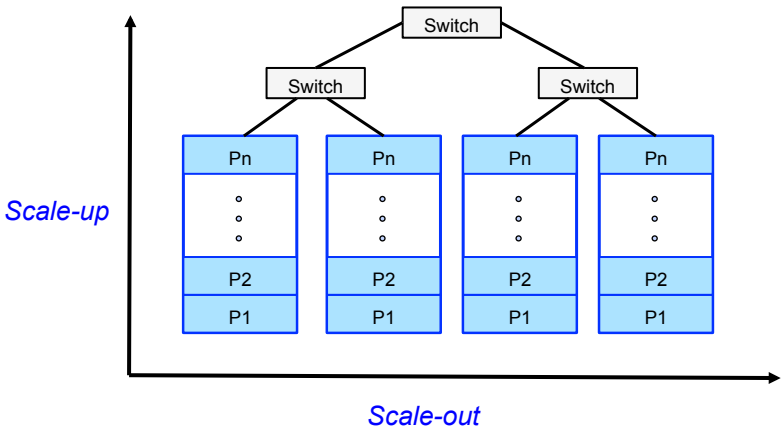
Scalabilité

- Idéal: scale-up linéaire
  - Maintien du niveau de performance face à la montée en charge (ou volumétrie des données) par augmentation proportionnelle de la configuration



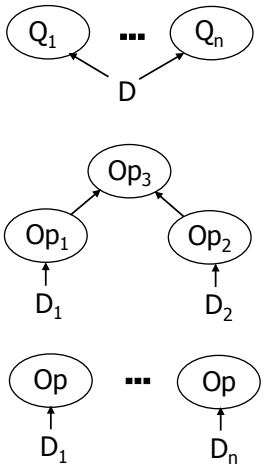
Scalabilité verticale vs horizontale

- Typiquement dans un cluster de machines



Parallélisme de données

- Inter-requête
  - Différentes requêtes sur la même donnée
  - Pour requêtes concurrentes
- Inter-opération
  - Différentes opérations de la même requête sur différentes données
  - Pour requêtes complexes
- Intra-opération
  - La même opération sur des données différentes
  - Pour requêtes lourdes

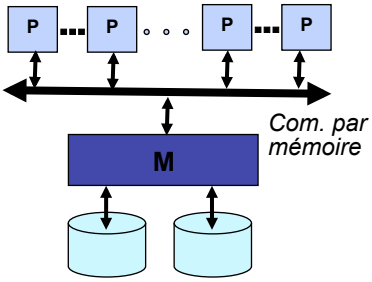


3. Architectures parallèles

- Trois alternatives, selon la façon dont processeurs, mémoire (RAM) et disque sont interconnectés
  - Calculateur à mémoire partagée
  - Cluster à disque partagé
  - Cluster shared-nothing

Calculateur à mémoire partagée

- Disque et mémoire sont partagés
  - Symmetric Multiprocessor (SMP)
  - Non Uniform Memory Architecture (NUMA)
    - Exemples: IBM Numascale, HP Proliant, Data General NUMALiiNE, Bull Novascale



\+ Simple pour les apps

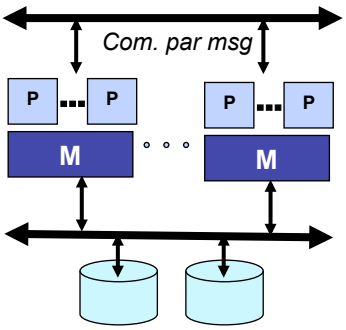
\+ Communications rapides

\- Extensibilité limitée, coût

Pour écritures intensives, cher pour OLAP et big data

Cluster à disque partagé (DP)

- Disque partagé, mémoire privée
  - Bus haut-débit pour interconnecter mémoire et disque (niveau bloc)
    - Infiniband, Fibre Channel
  - Besoin d'un distributed lock manager (DLM) pour la cohérence des caches
  - Exemples
    - Oracle RAC et Exadata
    - IBM PowerHA



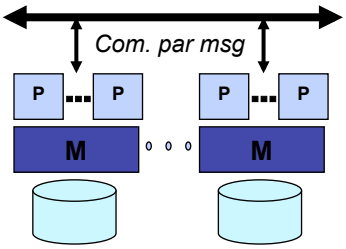
\+ Simple, extensibilité

\- DLM complexe, coût du bus

Pour écritures intensives ou OLAP / big data

Cluster shared-nothing (SN)

- Pas de partage (mémoire ou disque)
  - Pas besoin de DLM
  - Mais besoin de partitionner les données
  - Exemples
    - DB2 DPF, SQL Server Parallel DW, Teradata, MySQLcluster
    - Google search, SGBD NoSQL



\+ Extensibilité, coût réduit

\- Réglage complexe

\- Mises à jour distribuées

Pour OLAP et big data (lectures intensives)

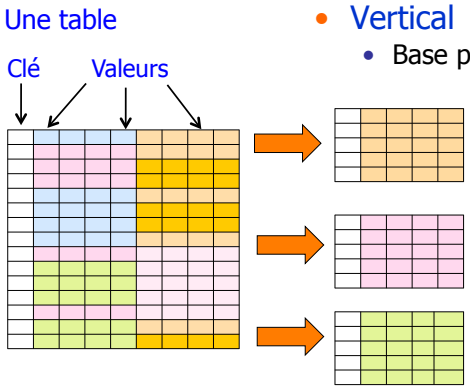
DP versus SN

- DP
  - Simple à administrer (ajout de disques)
  - Bus haut-débit cher
  - Scalabilité limitée
    - Mais on peut repousser les limites
      - ex. Exadata database machine
  - Bien adapté OLTP (mises-à-jour simples)
- SN
  - Plus complexe (partitionnement, réglage)
  - Excellent rapport performance/coût
  - Très grande scalabilité (scale out)
  - Bien adapté OLAP et big data (lectures)

4. Principales techniques

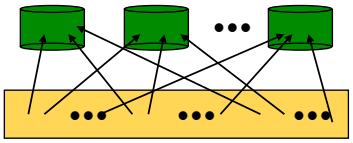
- Partitionnement et indexation des données
  - Problème avec les distributions non uniformes
- In memory
  - Le disque est très lent (environ *100K* fois + lent que RAM)
    - Exploiter les structures en mémoire et la compression
- Parallélisation et optimisation de requêtes
  - Automatique avec un langage déclaratif (ex. SQL)
  - Assisté par le programmeur sinon (ex. MapReduce)
- Transactions
  - Difficile car transactions distribuées (2PC)
    - Les SGBD NoSQL systems ne fournissent pas les transactions
- Haute disponibilité
  - Avec bcp de nœuds (ex. des milliers), la panne d'un nœud devient la norme, pas l'exception
    - Exploiter la réplication et le failover

Partitionnement des données



- Vertical
  - Base pour les column stores
- Horizontal (sharding)
  - Les shards peuvent être stockés et répliqués sur différents nœuds

Différents schémas de sharding

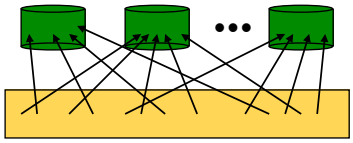


Round-Robin

$i^{ème}$ élément au nœud (i *mod* n)

\+ équilibrage de charge

\- requêtes de scan

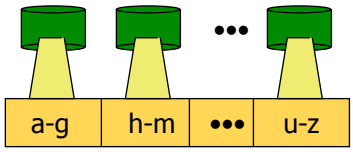


Hashing

(k,v) au nœud h(k)

\+ requêtes « K=v »

\- pb avec le biais



Range

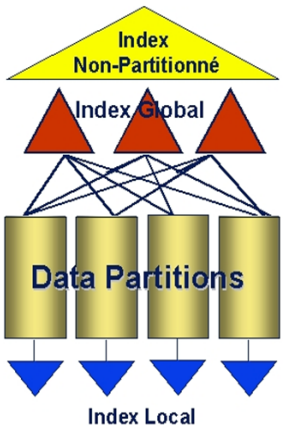
(k,v) au nœud qui gère l'intervalle de k

\+ requêtes « K=v » et « K in [v_1, v_2] »

\- gestion d'index

Indexation

- Fonctionnalités
  - Index secondaire ou fichier inverse
- Deux niveaux
  - Index global
    - Index *(attribut, liste de (n° shard, clés)*
  - Index local
    - Index *(clé, valeur)*



Réplication

- Disque miroir
  - Améliore disponibilité et performances
  - Pb d'équilibrage de charge en cas de panne d'un nœud

Noeud	1	2	3	4
Table	R_1	R_2	R_3	R_4
R_1		R_1	R_1	
R_2			R_2	R_2
R_3	R_3			R_3
R_4	R_4	R_4		

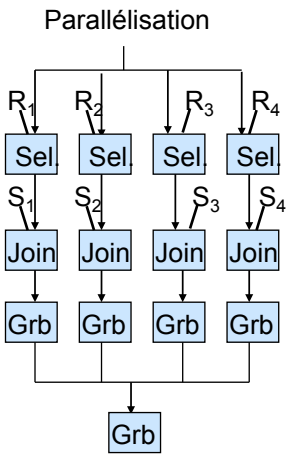
- Partitionnement chaîné (Teradata)
  - Equilibrage de charge
  - Plus complexe (avec *n* copies)

Noeud	1	2	3	4
Table	R_1	R_2	R_3	R_4
R_1		r_{12}	r_{13}	r_{14}
R_2	r_{21}		r_{23}	r_{24}
R_3	r_{31}	r_{32}		r_{34}
R_4	r_{41}	r_{42}	r_{43}	

Traitement de requêtes parallèles

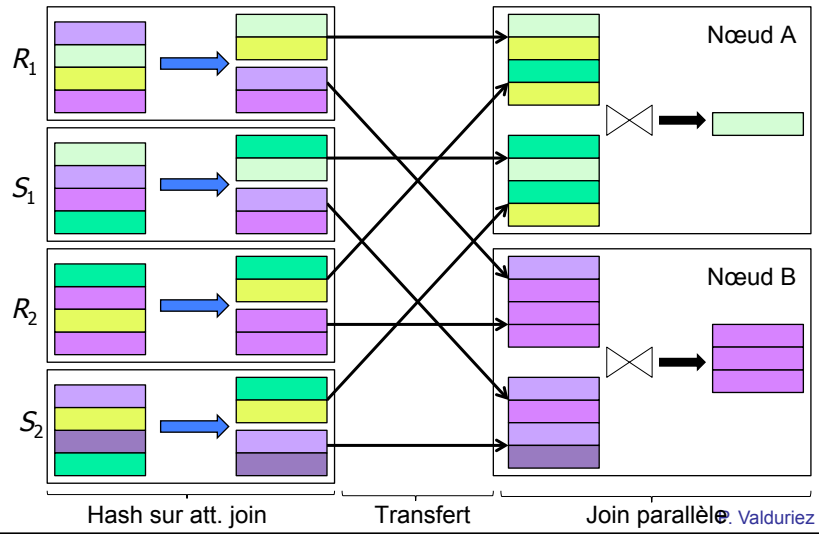
1. Parallélisation des requêtes lourdes
   1. traduction en plans d'exécution parallèles
2. Exécution des opérations parallèles
   - algorithmes parallèles pour les opérateurs relationnels
   - adaptation du degré de parallélisme pour équilibrer la charge

Select … from R,S
where …group by…



Jointure parallèle par hachage

Objectif: calculer $R \bowtie S = \bigcup_{i=1}^{n} R_i \bowtie S_i$ avec *n* noeuds

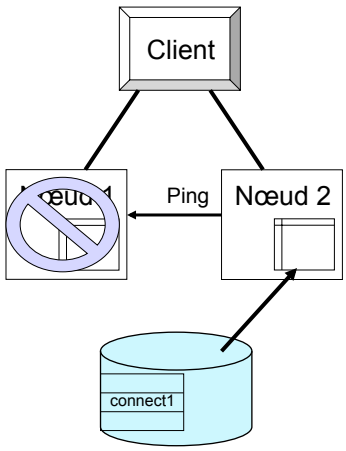


Le Failover

- En cas de panne d'un nœud
  - Détection par un autre noeud
  - Reprise de la connexion et des données
  - Besoin de recalculer les données perdues

    =>

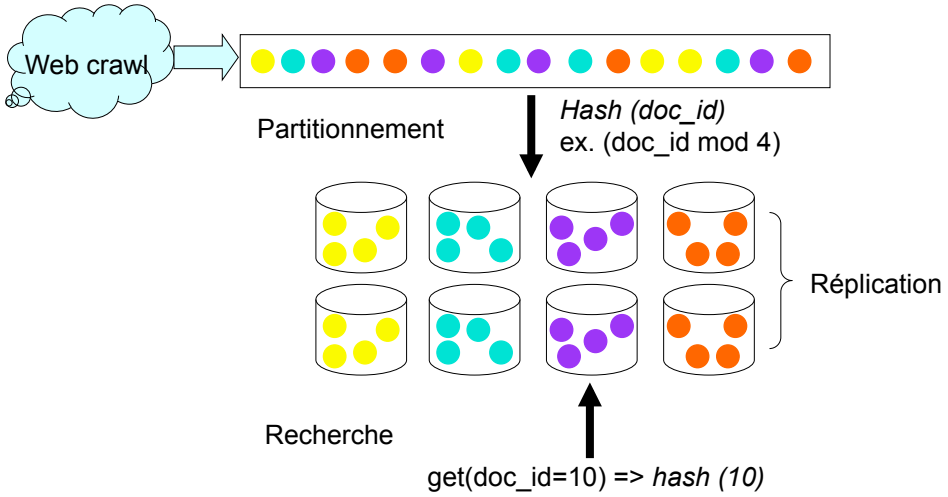
    Importance des savepoints réguliers pour les grandes requêtes



5. Etude de cas: Google Search

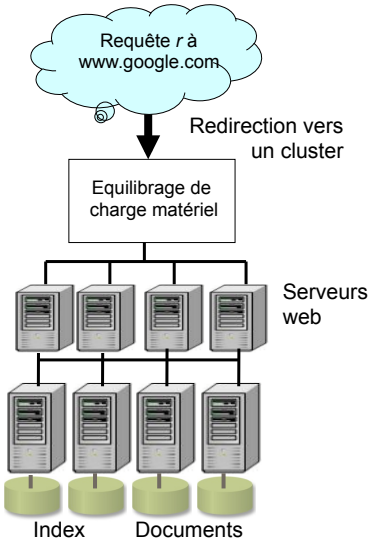
- Distribution et réplication massive des données dans des clusters
  - Exploitation massive du parallélisme
  - Index de documents: <*mot-clé: liste de doc_ids*>
  - Application intensive en lecture
- Le moteur de recherche en chiffres (estimation)
  - Des milliards de requêtes / jour
  - Des dizaines de data centers dans le monde, chacun ayant
    - Un cluster SN hébergeant une copie du web
      - Environ plusieurs pétaoctets (plusieurs milliards de pages)
    - Total estimé à plusieurs millions de serveurs

Algorithme de partitionnement



Traitement d'une requête Google

1. Allocation de r à un serveur web
   - Contrôle l'execution parallèle et formatte le résultat en HTML
2. Accès aux index à partir des mot-clés de r
   - Produit une liste de *doc_ids* triée par pertinence (algorithme PageRank)
3. Accès aux docs de la liste
   - Produit un résumé par doc.



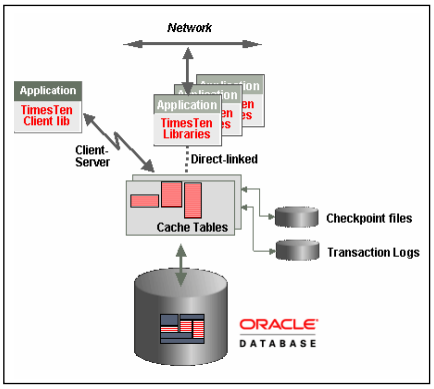
6. Machines bases de données

- SGBD sur machine multiprocesseur ou SGBD parallèle
  - Combinaison de matériel/logiciel dédié à la gestion de données
    - Réseau d'interconnexion à haut débit
      - Infiniband, Fibre channel
    - Grande mémoire RAM et in memory
    - Mémoires flash comme cache
    - Disques SSD (Solid State Disk) à base de flash
    - CPU multi-cœurs et GPU

Principaux produits

Editeur	Produit	Archi.	Remarques
EMC	GreenPlum	SN	Hybride SQL/MapReduce, basé sur PostgreSQL
HP	Vertica	SN	Orienté colonne
IBM	DB2 Pure Scale DB2 Database Partitioning Feature PureData System for Analytics	DP SN	Scalable POWERparallel (SP) Linux sur cluster Acquisition de Netezza
Microsoft	SQL Server SQL Server PDW	DP SN	Windows only
Oracle	Real Application Cluster Exadata Database machine MySQL	DP DP SN	Portabilité OSS sur cluster Linux
ParAccel	ParAccel Analytic Database	SN	Orienté colonne
SAP	High-Performance Analytic Appliance (HANA)	SN	In memory, colonne
Teradata	Teradata Database Aster	SN SN	Unix et Windows Hybride SQL/MapReduce

L'approche d'Oracle

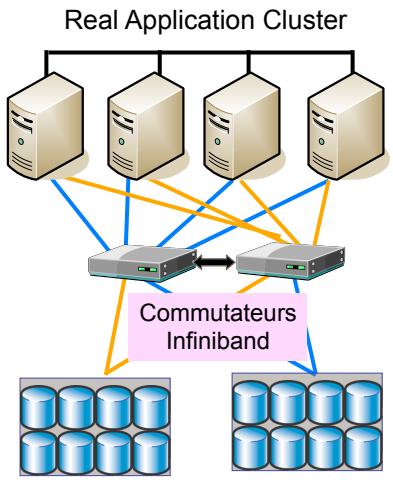


Architecture Shared-disk

- Options
  - In-Memory Database Cache
    - Utilisation du moteur TimesTen
      - 10 fois + rapide qu'avec format disque
  - Real Application Cluster (RAC)
    - *n* instances avec gestion automatique de la charge

Machine base de données Exadata

- Oracle + Sun
- Objectifs
  - OLTP et/ou OLAP
- Real Application Cluster
- Serveur de stockage = cache intelligent
  - + 14 cellules, chacune avec
    - Processeurs, avec RAM
    - Mémoire Flash
    - Disques



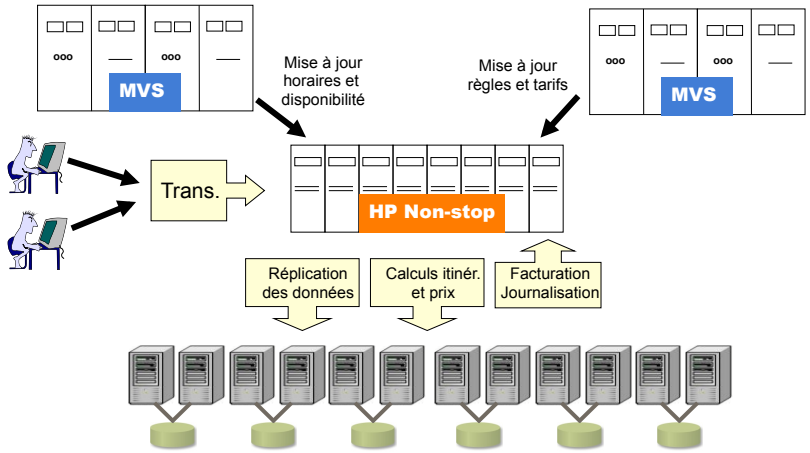
Etude de cas: Sabre ATSE

- Sabre (www.sabre.com)
  - 1er système de réservation informatisé (mainframe IBM) dans les années 1960
  - Utilisé dans le monde entier par les agences de voyage
- Air Travel Shopping Engine (ATSE)
  - Application la plus critique de Sabre : calcul d'itinéraire et de prix
- Problèmes avec ATSE/mainframe:
  - Avec l'achat en ligne/internet, augmentation constante du nombre de requêtes (des millions par jour) et de demandes de nouveaux services
  - Augmentation importante des coûts et des délais avec les mainframes

Solution de migration

- Architecture distribuée
  - Serveurs HP Non Stop (dispo. = 99,999%)
    - Pour les transactions et les mises à jour
  - Cluster de 45 HP rx5670 (4 procs.) sous MySQL/Linux
    - Pour les requêtes de lecture
  - Réplication synchrone des données avec l'outil Goldengate
    - 24h/24, 100 tables MySQL, 50GB/serveur
- Projet (réalisé avec EDS)
  - Accès C++ à MySQL
  - Choix de MySQL/Linux après benchmarks sur SGBD concurrents

Architecture



Cluster de serveurs MySQL/Linux

Résultats

- Plusieurs million $ d'économie
  - Matériel et logiciel
- Protection des éléments stratégiques de l'appli.
  - Licence commerciale MySQL
- Gains
  - Amélioration des performances
  - Extensibilité grâce au cluster
  - Ajout facile de nouveaux services

Exercise 1: Parallel Algorithm Design

- Objective
  - Design an efficient version of the parallel hash-based join algorithm
- Assumptions
  - A parallel shared-nothing cluster
  - Two tables *R* and *S*, partitioned on a number of nodes
    - $R_1, R_2, \ldots R_m$ and $S_1, S_2, \ldots, S_n$
  - Two kinds of tasks that can run at any node
    - Master task: has global information (partitioning, nodes's load, etc) and controls all the workers
    - Worker task: obeys the master
- Interfaces
  - Master-Worker
    - Start a task, with one input buffer and one or more output buffers (for storing partitions)
  - Worker-master
    - Notify master of end of work
- Data transfer between workers (like remote pipes)
  - Write to a distant buffer (at a different worker)
  - Read from a distant buffer
  - D-read and D-write are blocking operations
- Work to do
  - Write pseudo code for Master and Worker's tasks
  - Illustrate with a figure