Éléments techniques sur Azure

Matthieu Durut

September 20, 2012

Qu'est-ce que le Cloud ?

- HPC et Commodity Hardware
- Grid puis P2P
- Cloud

Différences Grid/Cloud

- Unicité du domaine administratif
- Credentials + contrôle d'identité/ d'activité
- Shift du modèle économique de la possession du bien de production à la location de service
- MultiTenancy et Elasticité => mutualisation comme en assurance...
- Pas de maintenance, machines remplacées instantanément grâce à la virtualisation

XaaS

- SaaS : GMail, Google Search, Facebook, Picasa, ...
- PaaS : Amazon MapReduce avec Simple Storage Service (S3), Microsoft Azure, BungeeLabs, ...
- IaaS : Amazon Elastic Cloud Compute (EC2)
- HaaS : RackSpace, ...

XaaS

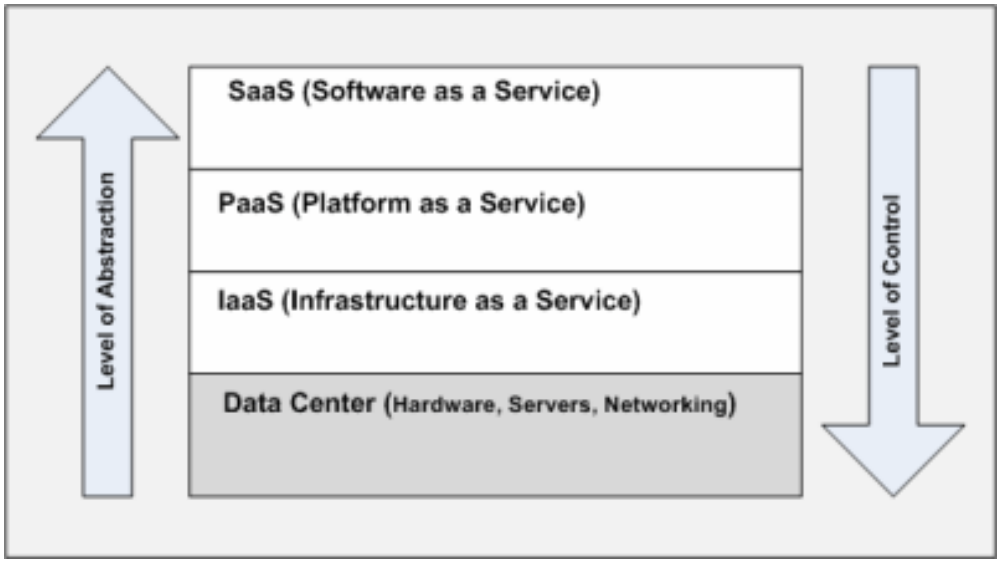

Figure : Paas, Iaas, SaaS

Compromis

- compromis abstraction/contrôle
- compromis scalability/coût de développement

Software Stack dans le cas PaaS

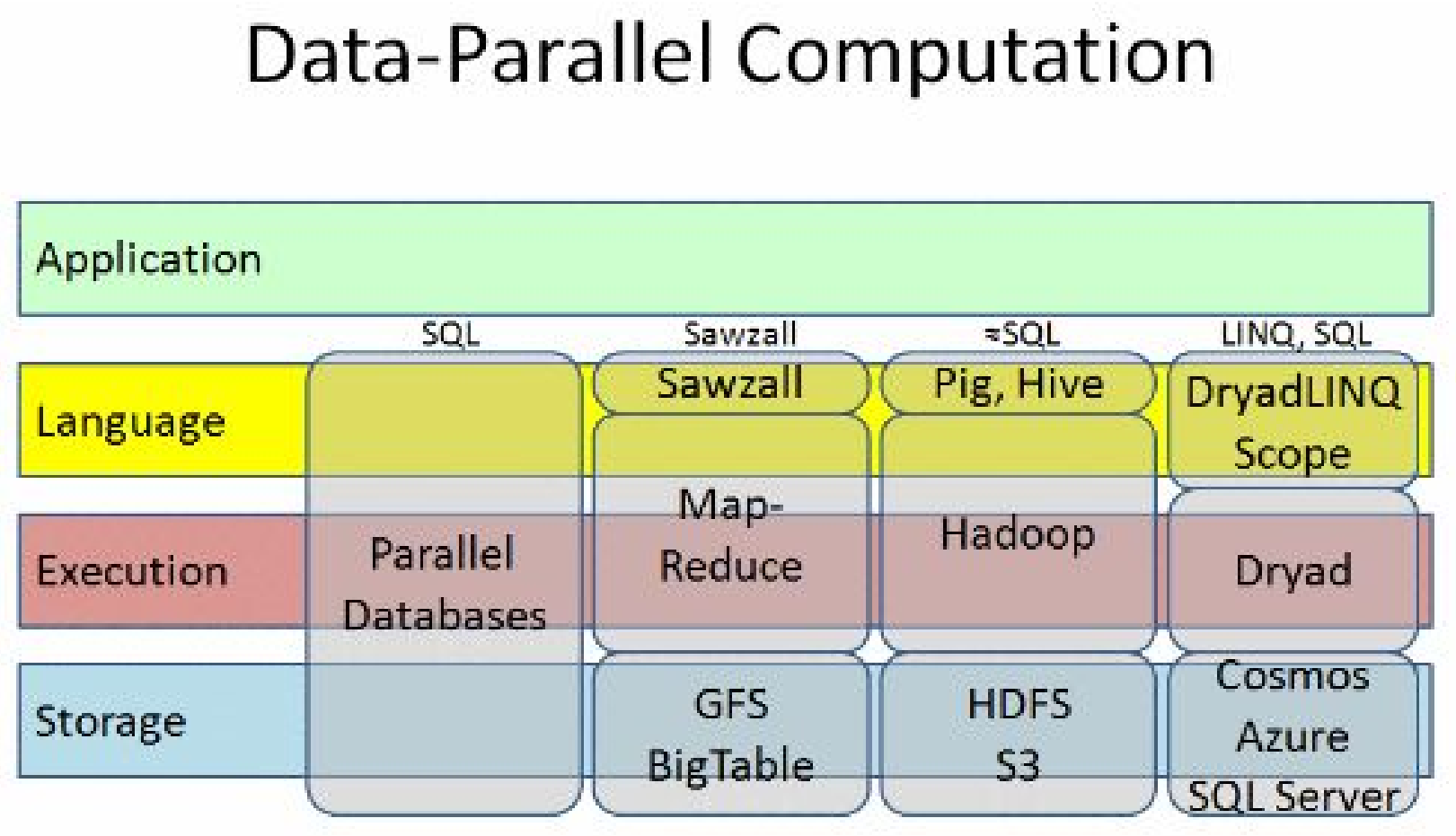

Figure : Les différentes stacks les plus utilisées/abouties

SQL

- Système très orienté applicatif : Relational DataBase Managemement System (RDBMS)
- Données structurées
- Invariants logiques
- Requêtage et index secondaires
- Quantum de modification : la transaction
- Possibilités de rollback

Propriétés ACID des bases SQL

- Atomicity
- Consistency
- Isolation
- Durability

Les différentes notions de la Consistency

- Unicité de la mémoire : consistency = invariants logiques ne sont pas cassés
- Storage réparti : Strong Consistency = toutes les updates appliquées sur tous les noeuds au même temps logique
- Storage réparti : Eventual Consistency = si le système est laissé au repos pendant un temps suffisant, il fournit une version consistente des données
- Difficulté et coût de la consistency dans le cas réparti. Double-commit protocol, ...

Stockage réparti : les propriétés principales

- Strong Consistency
- Partition tolerance
- Availability
- Faible latence
- CAP theorem et ses limites

Mouvement NoSQL

- Remise en cause des features : Facebook, Twitter, Youtube, ... Consistency encore primordiale ?
- Evolution de la contrainte d'échelle : cf slide suivant
- Evolution sur les contraintes métiers : latence rédhibitoire : Google Search. Très faible contrainte de latence : photos Facebook
- => design de nouveaux systèmes

Un exemple par les chiffres, Facebook

- Facebook en 2010, c'était :
- 500 millions d'utilisateurs actifs
- 690 milliards de pages vues par mois
- 6 milliards d'éléments de contenu partagés par semaine
- entre 60 000 et 100 000 serveurs
- 300 To de données en RAM avec Memcached
- Un ingénieur Facebook pour 1.1 millions d'utilisateurs

- Quelles features relâcher ?
- La latence comme paramètre d'ajustement
- Les différents choix de différents systèmes : Cassandra, Azure, ...

Taxonomie des storages cloud

- Document oriented DB : CouchDB, MongoDB. SimpleDB
- Graph DB :Horton, FlockDB, ...
- Key-value pairs DB: Cassandra, Dynamo, Velocity, BigTable, Azure BlobStorage, ...

Windows Azure Storage system (WAS)

- Usage interne : Bing, XBox Gaming Network, ...
- Usage externe : Storage as a Service.

Les composants du storage d'Azure

- Azure QueueStorage
- Azure BlobStorage
- Azure TableStorage
- Azure SQL

Cas d'usage fréquent du storage pour une appli cloud

- I/O de l'application avec l'extérieur par le BlobStorage
- Instructions générales du flow d'exécution par des messages dans le QueueStorage
- Résultats intermédiaires dans le TableStorage ou le BlobStorage

Azure QueueStorage

- Mécanisme de communication asynchrone faiblement couplé (loosely coupled)
- 500 messages/queue/seconde (étonnamment faible)
- Pas de garantie FIFO, mais comportement FIFO dans la pratique si non stressée.
- Garantie que chaque message est retourné au moins une fois.
- Nécessité d'idempotence des messages
- Message limité à 8 Ko => en pratique même moins

Azure BlobStorage

- Stocke des fichiers ou des instances de classes sérialisées.
- Chaque élément est stocké dans un "blob" : Binary Large OBject
- Chaque blob a une taille comprise entre quelques Ko et 50 Go
- Strong Consistency garantie ! (cf infra)

Azure BlobStorage 2

- Key: ContainerName/BlobName
- Contraintes sur le nommage de la Key et erreurs inattendues
- 3 à 63 caractères
- Seulement des lettres, des chiffres, et le symbole "-"
- Pas de majuscule

Azure BlobStorage 2

- Opérations disponibles : Get/Put/List
- mécanisme de "Optimistic nonblocking atomic read-modify-write" via storage stamp appelé Etag
- Pas d'atomicité multi-blobs, même dans le même container
- Notion de "logique atomique" intra-blob

Azure BlobStorage 3

- Logique de réplication inter datacenter
- Logique de réplication intra datacenter (3 replicas min)
- Notion de storage stamp (cf infra)
- Compromis latence/consistence. Quel choix pour Azure ?

Azure TableStorage

- Comparable au BlobStorage
- Key/Value pair Storage
- clé constituée de deux parties : PartitionKey/RowKey
- Une Entity possède une liste de paires de (Name/Typed Value) appelées Properties
- Exemple

Azure TableStorage 2

- Le schéma est flexible pour chaque Entity
- => pas d'index secondaire, toute requête entraine un scan linéaire des données...
- Notion d'atomicité au sein d'une même PartitionKey par paquet de 100 Entity max
- Restrictions : que des Insert, des Delete ou des Update.

Partitionning et Load Balancing

- Partitionnement manuel et automatique
- Frontière : la PartitionKey
- Un Partition Server adresse les requêtes de plusieurs Partitions (même si n serveurs de stockage derrière)
- Le WAS monitore l'utilisation et peut dynamiquement reconfigurer le map entre partitions et partition servers.

Partitionning et Load Balancing

- Compromis Atomicité/Scalabilité dans le cas du TableStorage
- Comment choisir ? Domain Driven Design ? ...

Mécanique interne du WAS

- Tous les détails dans [?]

BlobStorage ou TableStorage ?

- Atomicité : propriété nécessaire pour le projet ?
- Supériorité du BlobStorage pour les gros objets
- Aspect financier en faveur du TableStorage
- BlobStorage plus simple d'utilisation
- Le TableStorage est moins abouti pour le moment niveau stabilité

Considérations sur le développement d'applis sur Azure

- Idempotence (cf scénarii du slide suivant)
- Pas d'affinité CPU/Disque-dur, au contraire même
- Communication inter-machines

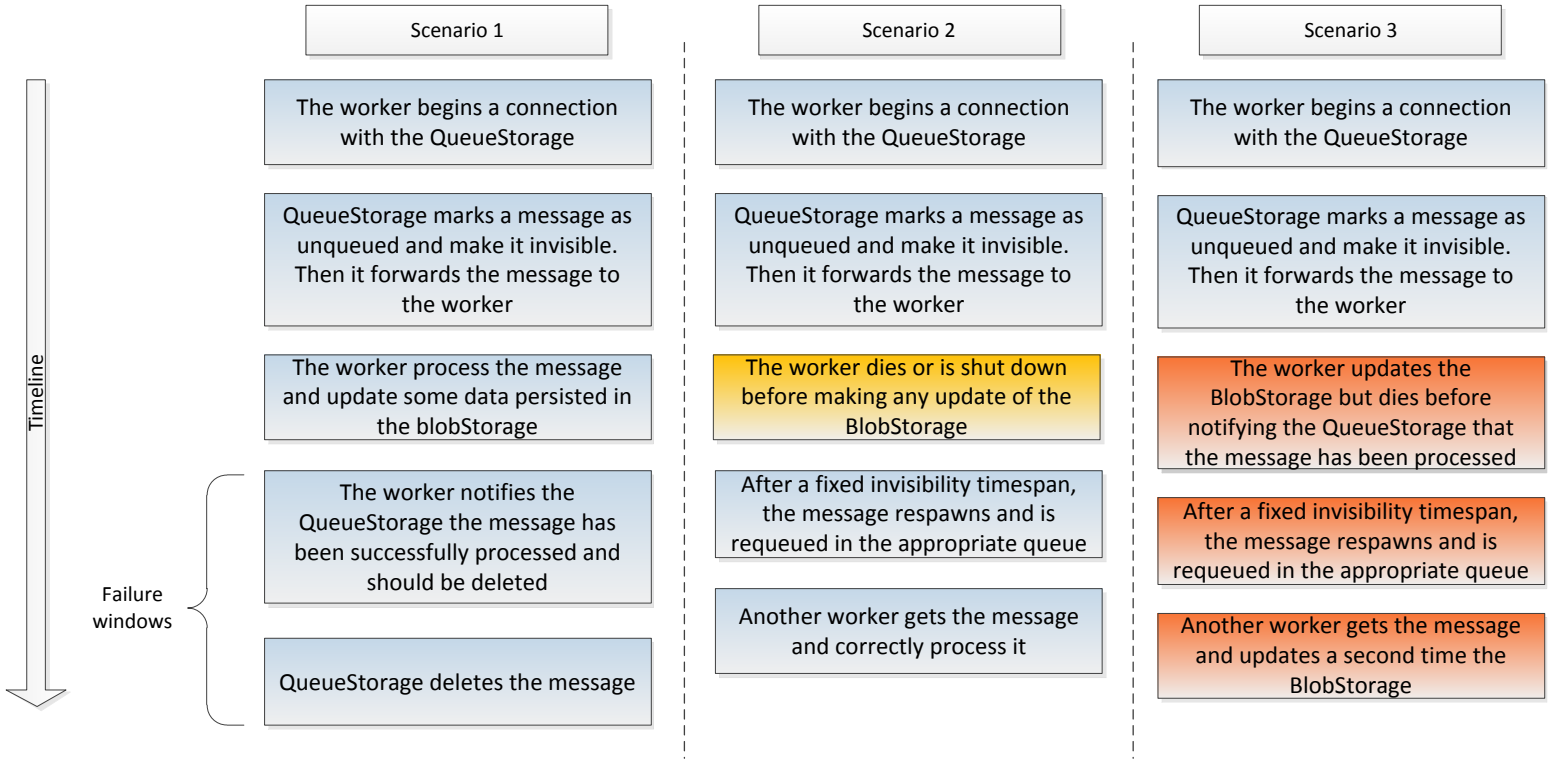

Figure : Idempotency

Communication inter-machines

- Pas de shared-memory efficace
- Pas de MPI
- Pas de framework P2P
- MapReduce trop peu expressif
- Dryad abandonné
- Utilisation du WAS: Queue et Blob
- Eviter les écritures concurrentes
- Lokad.Cloud, Lokad.CQRS

Primitives de synchro : les compteurs

- Compteurs idempotents, scalable
- Cas d'utilisation : égalité de lecture et d'écriture.
- Sharded counters à la Facebook ne marchera pas
- Des compteurs custom

Dimensionnement d'Azure aujourd'hui

- Taille des VM: Extra small, small, medium, large, extra-large
- Débit en écriture/lecture dans le storage. Cas idéal et le vôtre
- Puissance des CPU. Cas idéal et le vôtre
- Contrainte de l' "aggregated bandwidth"

Les prix

- 1 millions de transactions storage/workers = 1 dollar
- 1 heure de calcul = 10 cents
- 1 Go stocké par mois = 1 dollar