



Cassandra, le noSql en quasi Sql





Historique de la solution

Une histoire classique Amazon, Facebook, Apache...

Historique de la solution



Un besoin, une grosse boîte, une nouvelle technologie

L'histoire de Cassandra commence chez facebook en 2007

Facebook souhaite implémenter la recherche inbox sur tous les messages postés par l'utilisateur. La quantité de données induites est énorme et nécessite de remettre à plat la méthode de requêtage de la base.

C'est Avinash Lakshman et Prashant Malik qui se colle au développement en java chez Facebook.

On n'a pas évoqué Amazon, c'était sans compter sur Amazon's Dynamo.



Historique de la solution

Amazon's Dynamo, la base théorique



Dynamo est un ensemble de technique qui ensemble peuvent fournir :

- Une structure de stockage clef-valeur
- Un stockage distribué

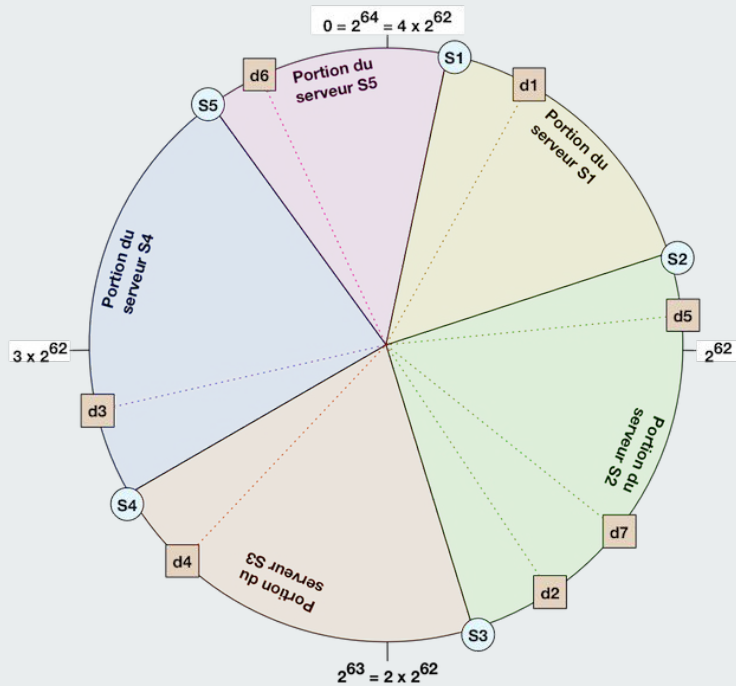
Il est à la fois une base de données et une table de hachages distribués.

Ses principes :

- Incremental scalability : On peut ajouter un par un des noeuds de stockage avec un minimum d'impact
- Symmetry : Tous les noeuds ont la même responsabilité (pas de chef d'orchestre)
- Decentralization : Les synchros sont peer-to-peer
- Heterogeneity : Les noeuds peuvent être de taille ou puissance différentes.

Historique de la solution

Amazon's Dynamo, la base théorique



| Chunk S1 | |
|----------|-------------|
| resp | d1 |
| réplicas | d5 d2 d4 d7 |

| Chunk S2 | |
|----------|----------|
| resp | d5 d2 d7 |
| réplicas | d4 d3 |

| Chunk S3 | |
|----------|-------|
| resp | d4 |
| réplicas | d3 d6 |

| Chunk S4 | |
|----------|-------|
| resp | d3 |
| réplicas | d6 d1 |

| Chunk S5 | |
|----------|-------------|
| resp | d6 |
| réplicas | d1 d5 d2 d7 |

Historique de la solution

Amazon's Dynamo, la base théorique



Jusqu'en 2012, sortie de DynamoDB, Amazon n'a pas exploité en propre Dynamo.

Mais Avinash Lakshman participe à l'étude Dynamo et compte bien exploiter le modèle chez Facebook (on reboucle).

Il n'est d'ailleurs pas le seul, d'autres projets emboîte le pas (avec moins de succès) :

- Aerospike
- Project Voldemort
- Riak

Historique de la solution



Sortie chez Facebook

<https://www.facebook.com/notes/facebook/inbox-search/20387467130/>

Le 20 Juin 2008, Facebook annonce la mise à disposition de la fonctionnalité inbox Search.

Dès Juillet 2008, le projet sort en open source sur Google Code sous le nom Cassandra.

A sa sortie le CTO de Facebook la décrit comme un model BigTable sur une infrastructure Dynamo-like.

<http://perspectives.mvdirona.com/2008/07/facebook-releases-cassandra-as-open-source/>



Historique de la solution

Encore la fondation Apache...

Le projet cassandra intègre l'incubateur de projet de la fondation Apache en Mars 2009 et devient top project en 2010.

Selon db-engines.com, en 2015, Cassandra est le 8ème SGBD le plus populaire et le 2ème dans la catégorie des NoSQL.

le premier est MongoDB, encore quelques jours et on en reparle.



Historique de la solution

Les releases de Cassandra par la fondation

| Version | Original release date | Latest version | Release date | Status ^[15] |
|---|-----------------------|----------------|--------------|--------------------------------------|
| 0.6 | 2010-04-12 | 0.6.13 | 2011-04-18 | No longer supported |
| 0.7 | 2011-01-10 | 0.7.10 | 2011-10-31 | No longer supported |
| 0.8 | 2011-06-03 | 0.8.10 | 2012-02-13 | No longer supported |
| 1.0 | 2011-10-18 | 1.0.12 | 2012-10-04 | No longer supported |
| 1.1 | 2012-04-24 | 1.1.12 | 2013-05-27 | No longer supported |
| 1.2 | 2013-01-02 | 1.2.19 | 2014-09-18 | No longer supported |
| 2.0 | 2013-09-03 | 2.0.17 | 2015-09-21 | No longer supported |
| 2.1 | 2014-09-16 | 2.1.19 | 2017-10-05 | Still supported, critical fixes only |
| 2.2 | 2015-07-20 | 2.2.11 | 2017-10-05 | Still supported |
| 3.0 | 2015-11-09 | 3.0.15 | 2017-10-10 | Still supported |
| 3.11 | 2017-06-23 | 3.11.1 | 2017-10-10 | Latest release |
| Legend: ■ Old version ■ Older version, still supported ■ Latest version ■ Latest preview version | | | | |



Intérêt de la solution

Apache Cassandra est un système permettant de gérer une grande quantité de données de manière distribuée. Ces dernières peuvent être structurées, semi-structurées ou pas structurées du tout.

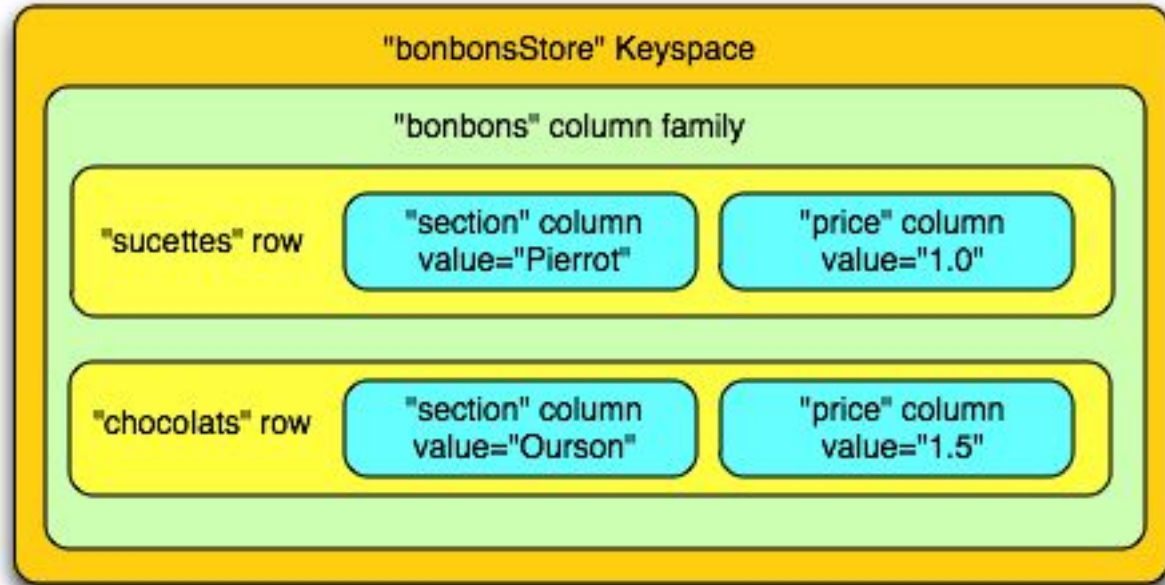


Intérêt de la solution

Cassandra a été conçu pour être hautement scalable sur un grand nombre de serveurs tout en ne présentant pas de Single Point Of Failure (SPOF).

Cassandra fournit un schéma de données dynamique afin d'offrir un maximum de flexibilité et de performance.

Mais pour bien comprendre cet outil, il faut tout d'abord bien assimiler le vocabulaire de base.



- **Keyspace** : c'est l'équivalent d'une database dans le monde des bases de données relationnelles. À noter qu'il est possible d'avoir plusieurs « Keyspaces » sur un même serveur.
- **Colonne (Column)** : une colonne est composée d'un nom, d'une valeur et d'un timestamp (instant de création).
- **Ligne (Row)** : les colonnes sont regroupées en Rows. Une Row est représentée par une clé et une valeur. Il existe deux types de rows : les « **wide row** » permettant de stocker énormément de données, avec beaucoup de colonnes et les « **skinny row** » permettant de stocker peu de données.
- **Une famille de colonnes (Column family)** : c'est l'objet principal de données et peut être assimilé à une table dans le monde des bases de données relationnelles. Toutes les rows sont regroupées dans les column family.



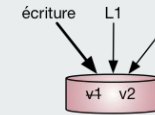
Intérêt de la solution

Pour quels besoins ?

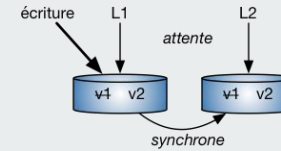
Intérêt de la solution / Pour quels besoins ?



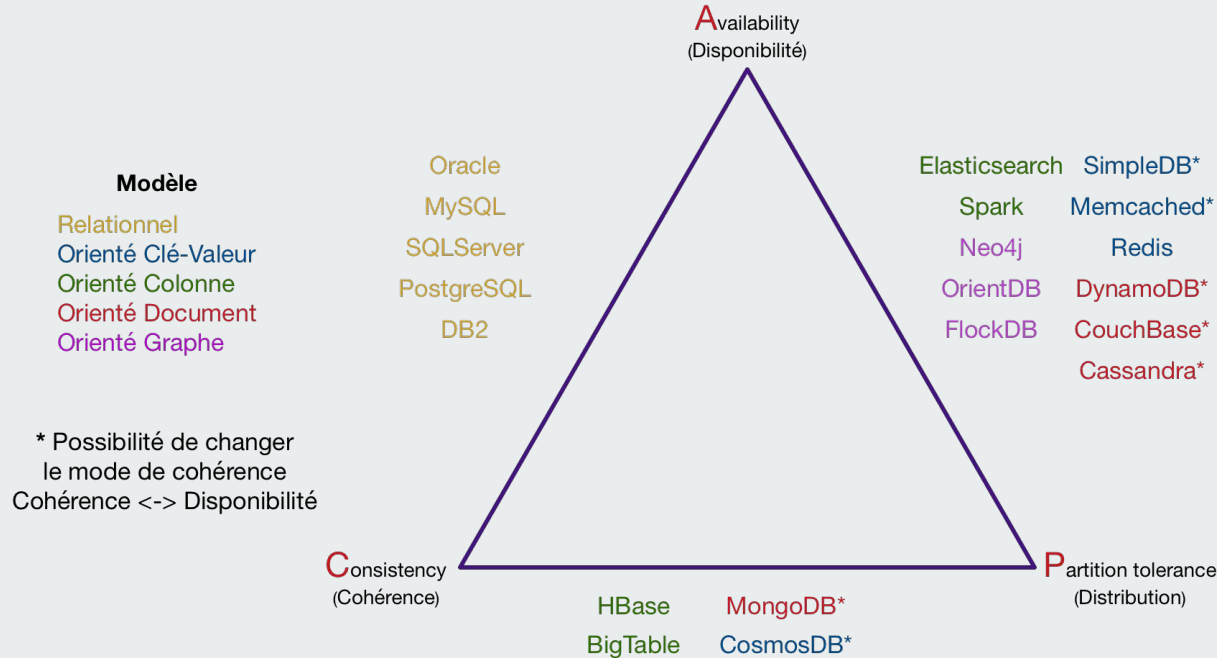
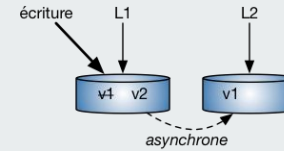
CA
Cohérence + Disponibilité



CP
Cohérence + Distribution



AP
Disponibilité + Distribution



* Possibilité de changer
le mode de cohérence
Cohérence <-> Disponibilité

Intérêt de la solution / Pour quels besoins ?

Cassandra a une bonne courbe de montée en charge par noeud

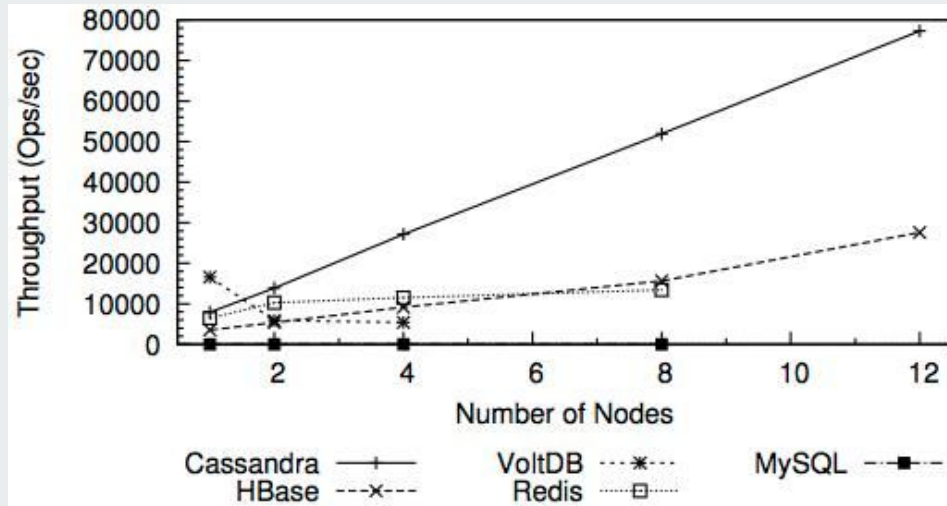


Figure 14: Throughput for Workload RSW



Intérêt de la solution

Quels sont les équivalents ?



Intérêt de la solution / Quels sont les équivalents ?

Orienté colonne

Cassandra est une base orienté colonne dans son système de stockage mais se rapproche d'un modèle orienté document sur son utilisation finale...
Son côté orienté colonne, et sa proximité en font une alternative possible à HBASE.



Intérêt de la solution / Quels sont les équivalents ?

Orienté document

Dans son fonctionnement plus structuré que HBase, Cassandra se rapproche d'une base de données orientée document. Dans ces cas d'usages, elle peut passer la main à :

- CouchBase : avec un requêtage SQL like
- CouchDB : base documentaire XML
- MongoDB : base documentaire JSon



Couchbase



mongoDB

Intérêt de la solution / Quels sont les équivalents ?

Couchbase != CouchDB

| | Couchbase Server | Apache CouchDB |
|----------------------|----------------------------------|-----------------------|
| Data models | Document, Key-Value | Document |
| Storage | Append-only B-Tree | Append-only B-Tree |
| Consistency | Strong | Eventual |
| Topology | Distributed | Replicated |
| Replication | Master-Master | Master-Master |
| Automatic failover | Yes | No |
| Integrated cache | Yes | No |
| Memcached compatible | Yes | No |
| Locking | Optimistic & Pessimistic | Optimistic with MVCC |
| MapReduce (Views) | Yes | Yes |
| Query language | Yes, N1QL (SQL for JSON) | No |
| Secondary indexes | Yes | Yes |
| Notifications | Yes, Database Change Protocol | Yes, Changes Feeds |



Intérêt de la solution

Quand l'utiliser et quand préférer autre chose ?

Intérêt de la solution / Quand l'utiliser et quand préférer autre chose ?



Beaucoup de flux, pas de jointures...

Lorsque la base de données peut être dénormalisée.
Lorsque il est prévu une montée en charge importante.

Jamais dans le cas d'une structure complexe

En effet, Cassandra ne peut

- ni faire de jointures,
- ni sous-requêtes

De plus (mais c'est un paramètre réglable), la base de données Cassandra est "éventuellement consistante"

Intérêt de la solution / Quand l'utiliser et quand préférer autre chose ?


Beaucoup de flux, pas de jointures...

Quelques exemples d'utilisation :

- gestion de contenu (bibliothèques numériques, collections de produits, dépôts de logiciels, collections multimédia, etc.),
- framework stockant des objets,
- collection d'événements complexes,
- gestion des historiques d'utilisateurs sur réseaux sociaux.

Qui l'utilise ?





Analyse d'un cas client



NETFLIX

Analyse d'un cas client




Plusieurs NoSQL chez Netflix, pourquoi?

Dans le cas d'un monstre comme Netflix, le moindre gain est considérable et chaque système a ses avantages et inconvénient, pour Netflix :

- Hbase a l'intérêt de son intégration à Hadoop
- Cassandra a pour elle son élasticité et sa facilité pour le scaling sans single point of failure

Analyse d'un cas client



Historique chez Netflix

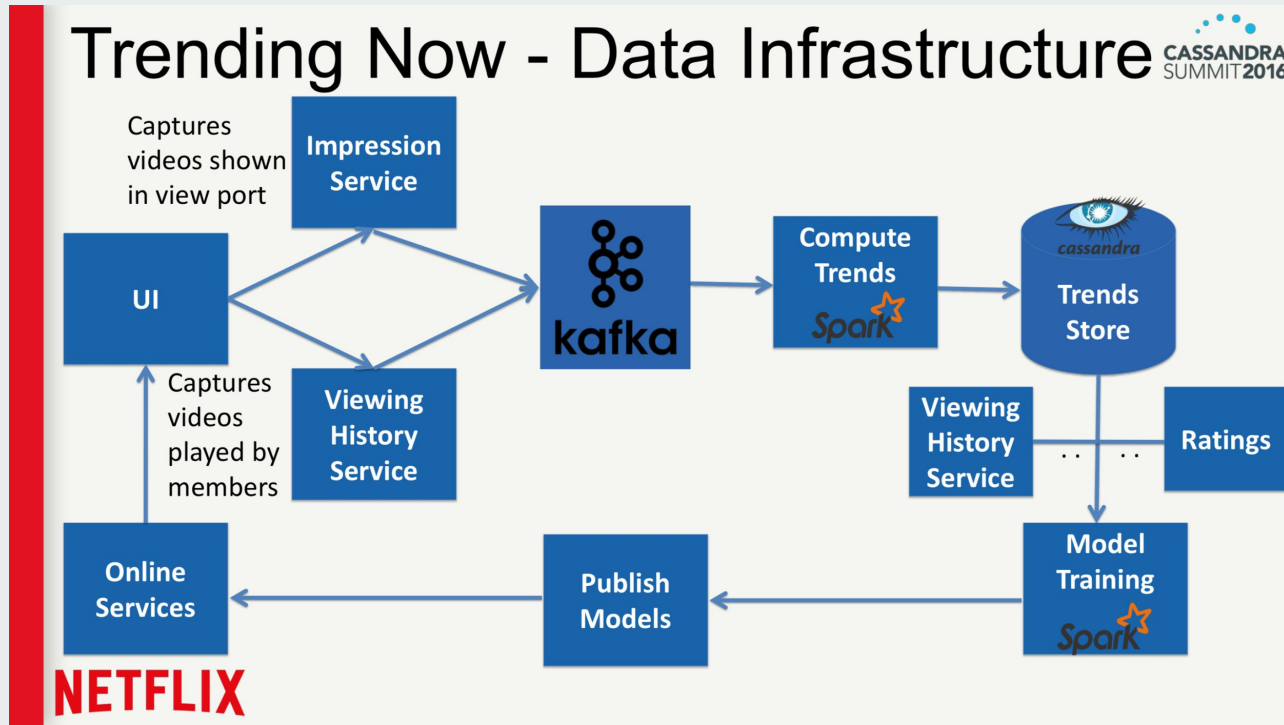
Comme beaucoup de grosses structures, Netflix a commencé son histoire sur le SGBD relationnel Oracle (en 2007).

En 2010, Netflix commence sa révolution en migrant ses datas sur Amazon Web Services mais conserve sa base Oracle comme backend. Dans le cas de Netflix, les problèmes commencent à se multiplier :

- Downtime sur changement de modèle
- Coût d'utilisation qui s'envole
- Base de données centrales et monolithiques qui devient le point of failure de Netflix
- Difficulté à scaler le modèle pour accompagner l'extension internationale

Analyse d'un cas client

Un exemple d'intégration pour les trendings



Analyse d'un cas client

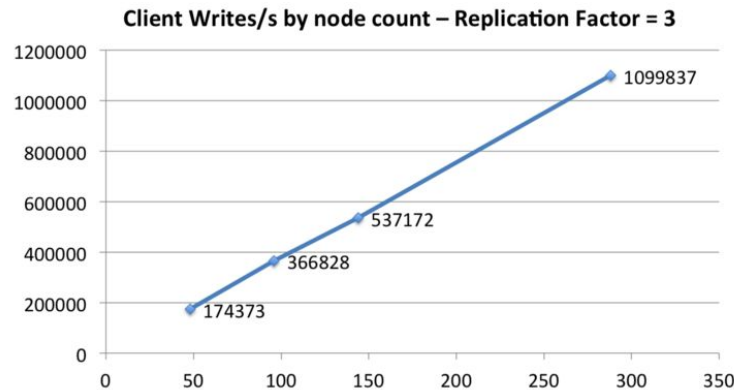
Que représente Cassandra maintenant chez Netflix

Cassandra stocke maintenant 95% des données (hors vidéos) de Netflix :

- Compte client
- Note des films
- Favori
- Logs
- Historique de visualisation

Le cluster Cassandra chez Netflix peut encaisser 10 millions de transactions par seconde
Il gère en moyenne 2,1 milliards de lecture par jour et 4,1 milliards d'écritures.

Scale-Up Linearity



Analyse d'un cas client

Etude des coûts sur un benchmark interne , document Netflix

Per Node Activity

| Per Node | 48 Nodes | 96 Nodes | 144 Nodes | 288 Nodes |
|---------------------|-------------|-------------|-------------|-------------|
| Per Server Writes/s | 10,900 w/s | 11,460 w/s | 11,900 w/s | 11,456 w/s |
| Mean Server Latency | 0.0117 ms | 0.0134 ms | 0.0148 ms | 0.0139 ms |
| Mean CPU %Busy | 74.4 % | 75.4 % | 72.5 % | 81.5 % |
| Disk Read | 5,600 KB/s | 4,590 KB/s | 4,060 KB/s | 4,280 KB/s |
| Disk Write | 12,800 KB/s | 11,590 KB/s | 10,380 KB/s | 10,080 KB/s |
| Network Read | 22,460 KB/s | 23,610 KB/s | 21,390 KB/s | 23,640 KB/s |
| Network Write | 18,600 KB/s | 19,600 KB/s | 17,810 KB/s | 19,770 KB/s |

Node specification – Xen Virtual Images, AWS US East, three zones

- Cassandra 0.8.6, CentOS, SunJDK6
- AWS EC2 m1 Extra Large – Standard price \$ 0.68/Hour
- 15 GB RAM, 4 Cores, 1Gbit network
- 4 internal disks (total 1.6TB, striped together, md, XFS)



Time is Money

| | 48 nodes | 96 nodes | 144 nodes | 288 nodes |
|-----------------------|------------|------------|------------|-------------------------|
| Writes Capacity | 174373 w/s | 366828 w/s | 537172 w/s | 1,099,837 w/s |
| Storage Capacity | 12.8 TB | 25.6 TB | 38.4 TB | 76.8 TB |
| Nodes Cost/hr | \$32.64 | \$65.28 | \$97.92 | \$195.84 |
| Test Driver Instances | 10 | 20 | 30 | 60 |
| Test Driver Cost/hr | \$20.00 | \$40.00 | \$60.00 | \$120.00 |
| Cross AZ Traffic | 5 TB/hr | 10 TB/hr | 15 TB/hr | 30 ¹ TB/hr |
| Traffic Cost/10min | \$8.33 | \$16.66 | \$25.00 | \$50.00 |
| Setup Duration | 15 minutes | 22 minutes | 31 minutes | 66 ² minutes |
| AWS Billed Duration | 1hr | 1hr | 1 hr | 2 hr |
| Total Test Cost | \$60.97 | \$121.94 | \$182.92 | \$561.68 |

¹ Estimate two thirds of total network traffic

² Workaround for a tooling bug slowed setup



Point sur les offres cloud



Point sur les offres cloud





Bon appétit