SAM – cours 9a

Transactions dans les systèmes New SQL

HUBERT NAACKE

2023

Intro : Transaction

- Transaction SQL
  - Séquence d'instructions SQL manipulant des données
    - insert, select, update, delete
  - Propriétés des transactions
    - **A**tomicité d'une séquence d'opérations
    - **C**ohérence : données intègres
    - **I**solation : chaque utilisateur est isolé des autres
    - **D**urabilité : ne jamais perdre des données
- Application transactionnelle
  - Traitement de transactions en ligne : OnLine Transaction Processing (OLTP)
    - Les transactions à traiter ne sont pas connues à l'avance
    - ≠ Traitement offline d'un lot de transactions prédéfinies

Intro : Large échelle

- Dynamique
  - De + en + d'utilisateurs
  - Possible fluctuation du nombre d'utilisateurs : ↗ ou ↘
- Volume des données
  - Augmente avec les nouveaux utilisateurs, et les nouvelles fonctionnalités
- Intensité du workload
  - Augmente avec le nombre d'utilisateurs
- Besoins
  - Performance élevée: 1K à 1M de transactions par minute (tpm)
  - Ceci à faible coût matériel et logiciel : rapport perf/prix élevé

Calvin

- Calvin : un système newSQL
  - Système réparti, basé sur une architecture matérielle de type cluster
  - Conçu pour être scalable
  - premier prototype en 2013 puis transféré dans la solution Fauna

Calvin : principes

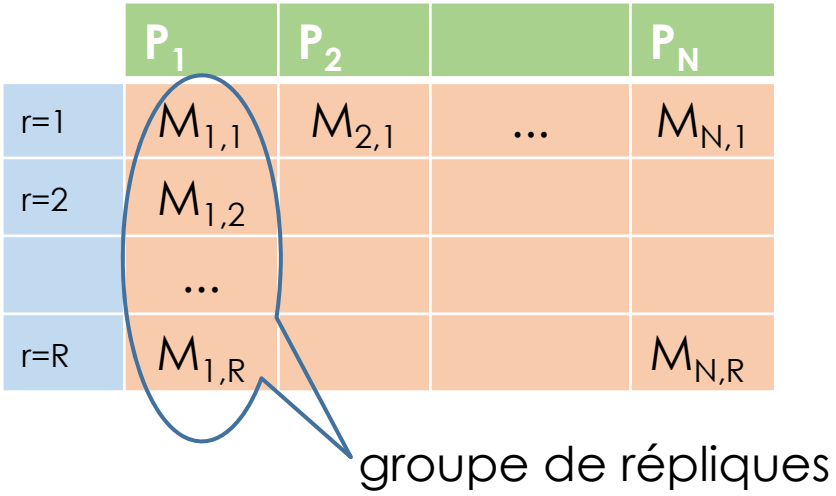
- Données SQL
- Transactions SQL courtes
  - durée **bornée**: nécessaire pour ordonner les transactions
- Système déterministe
  - Détermine l'ordre global des transactions **avant** de les traiter.
  - Traite les transactions dans l'ordre préalablement déterminé

- Nouveauté
  - Supporte les transaction globales (**multi-partitions**) sur des données réparties
    - Traitée par plusieurs transactions **locales** indépendantes
    - Sans nécessiter de validation globale

Architecture de Calvin

- Données partitionnées et répliquées sur un cluster de machines
  - N partitions: P_1 , ... P_i , ..., P_N
  - R répliques par partition

  → donc N*R machines
    - dénotées $M_{1,1}$ à $M_{N,R}$



Architecture de Calvin

Exemple avec 3 partitions (N=3) et un degré de réplication = 3

Cluster de 9 machines Mi,j

Coordination des transactions

- **Ordonner** les transactions en 3 étapes décentralisées
  1. Laisser des transaction arriver sur des machines
  2. Former des groupes de transactions (un groupe par partition)
  3. Compléter les groupes pour tenir compte des transactions **multi-partitions**
- Puis traitement décentralisé des transactions

- Pour **réaliser** cette coordination chaque machine $M_{i,j}$ a
  - un **séquenceur** : ordonner les transactions
  - un **scheduler :** traiter les transactions

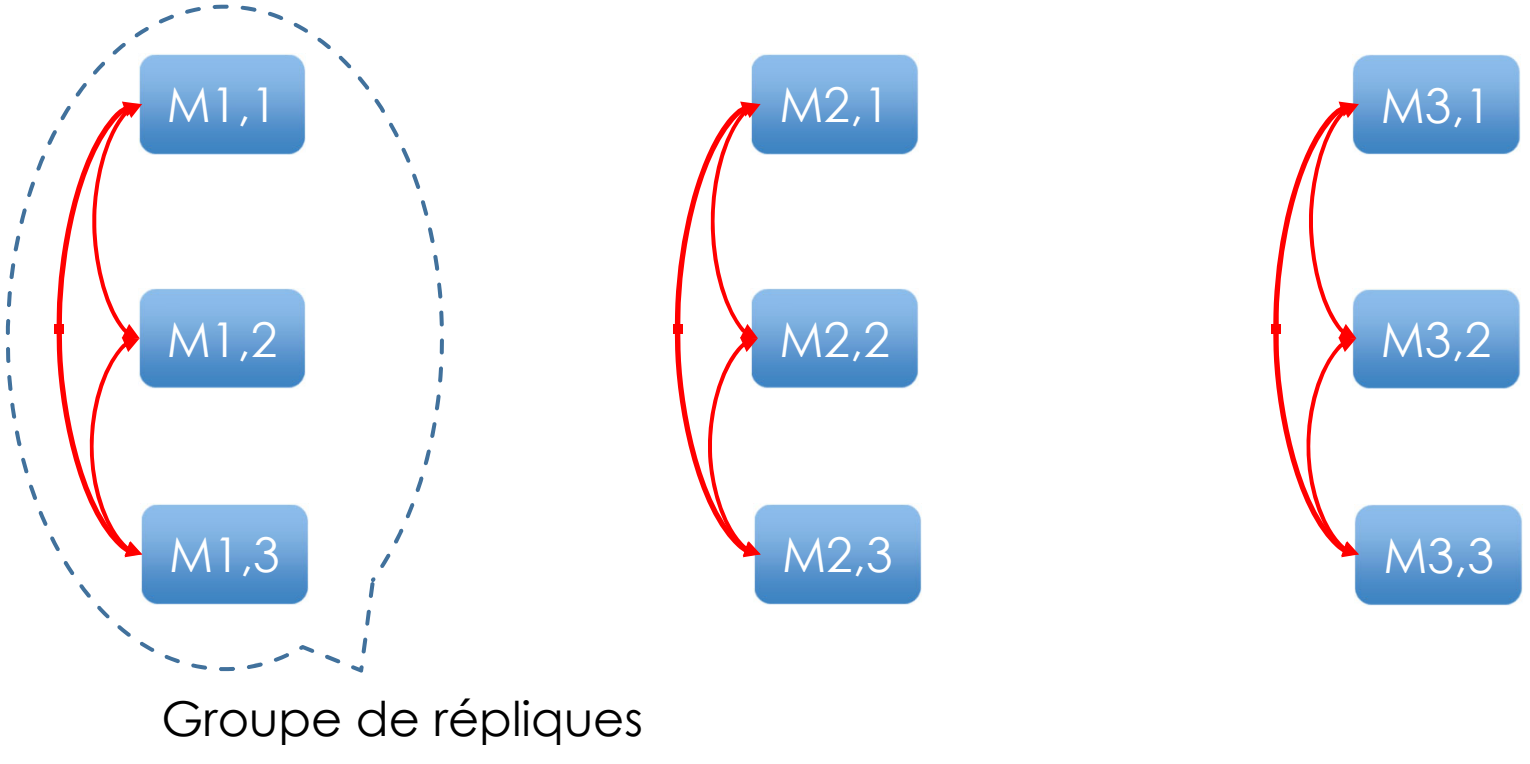
Ordonner les transactions

Séquenceur

- Découpe le temps en fenêtres de période fixe
- Reçoit des transactions
- En fin de période courante, **propage** les demandes aux autres séquenceurs du même groupe de répliques
  - Ainsi chaque séquenceur d'un groupe connait la liste des demandes du groupe
  - Avantages :
    - Plusieurs points d'entrée dans un groupe : disponibilité
    - Propagation interne à un groupe : plus rapide qu'une propagation globale entre toutes les machines.

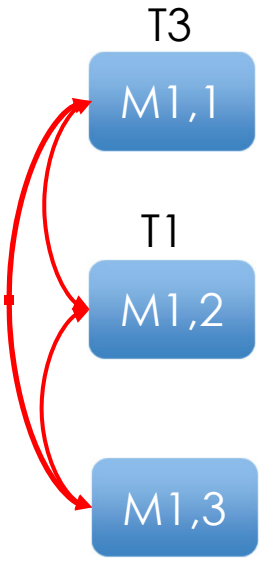
Séquenceur: exemple

- Séquenceur : coordination "verticale"

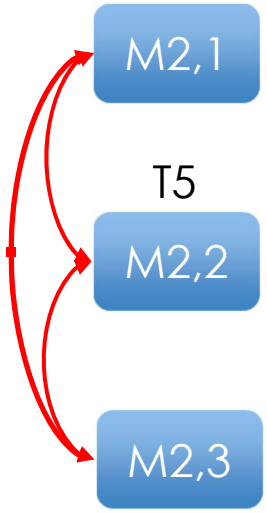


Séquenceur : exemple 1

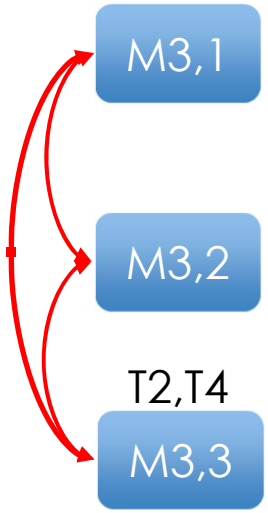
- Les transactions T1 à T5 arrivent pendant la période courante
- Puis coordination verticale en fin de période



T3 → M1,2 et M1,3
T1 → M1,1 et M1,3



T5 → M2,1 et M2,3



(T2,T4) → M3,1 et M3,2

Séquenceur : exemple 1 (suite)

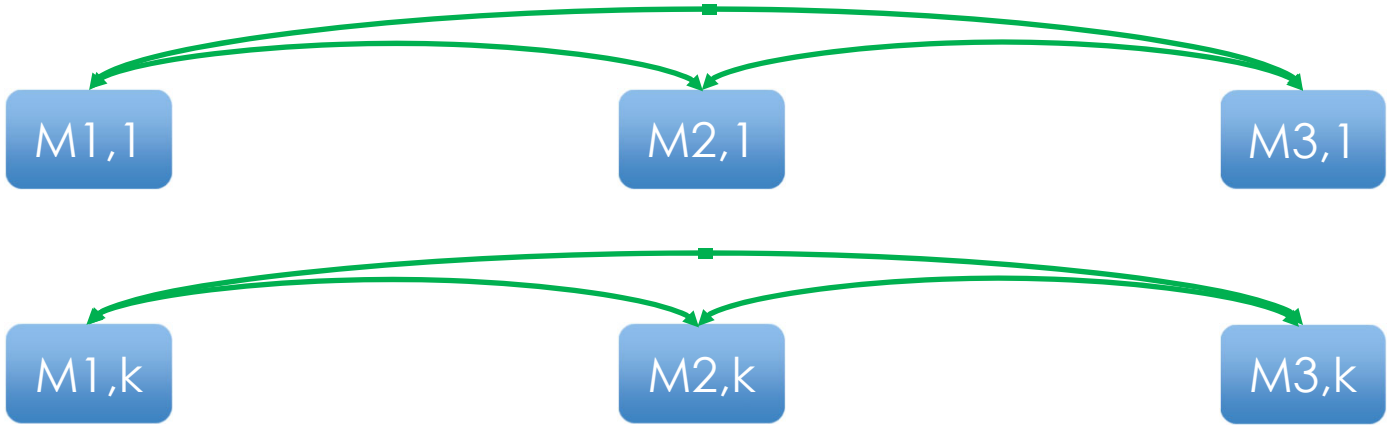
T3,T1	T5	T2,T4
M1,1	M2,1	M3,1
T3,T1	T5	T2,T4
M1,2	M2,2	M3,2
T3,T1	T5	T2,T4
M1,3	M2,3	M3,3

Transmission d'un séquenceur vers les schedulers

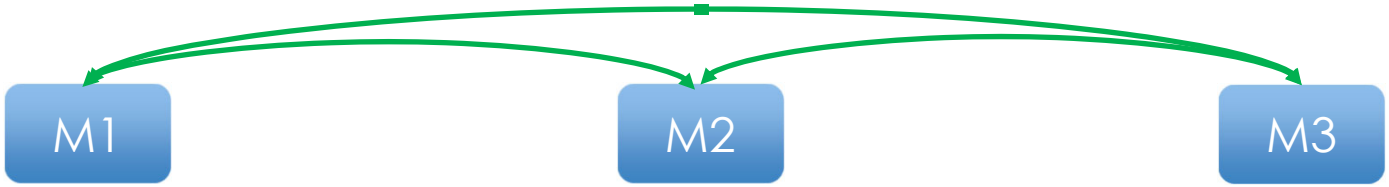
- Un séquenceur communique seulement avec les schedulers ayant le même indice r
  - Communication horizontale sur une même "ligne"
  - Du séquenceur $M_{i,r}$ vers les schedulers parmi $M_{1,r}$ à $M_{n,r}$
- Transmet les transactions aux seuls schedulers concernés
  - Chaque transaction est transmise sur chaque machine stockant au moins une donnée de la transaction.
- La réplication est "orthogonale"
  - Comportement identique sur chaque "ligne" de machines

Séquenceur → Scheduler

- Coordination "horizontale"



- Représentation simplifiée (quel que soit k)

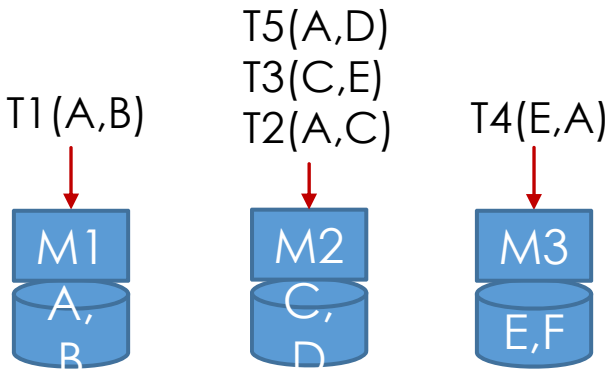


Ordre global des transactions

- L'ordre global des transactions est déterminé de manière **décentralisée**
  - En fixant un ordre global entre les séquenceurs à partir du numéro de machine : M1 < M2 < ... Mi < Mj .... < M_N
  - Pour tout i < j: une transaction posée sur Mi **précède** une transaction posée sur Mj
  - Si i = j, alors Ta précède Tb si a<b

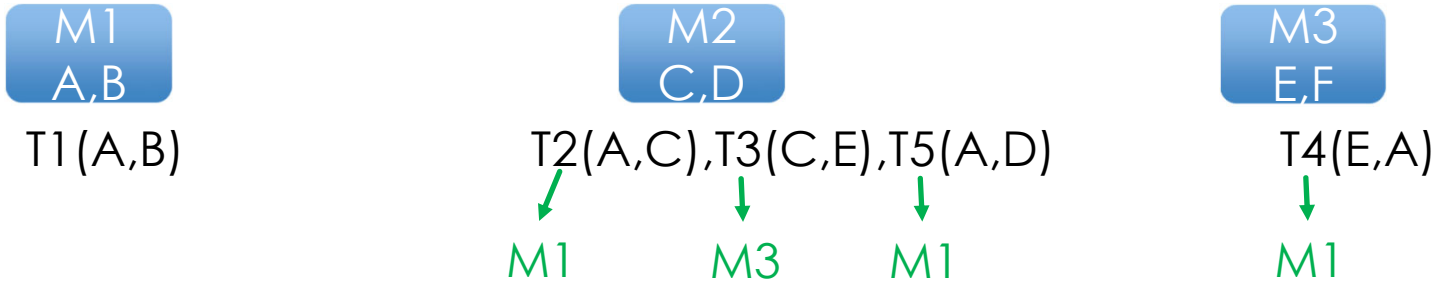
Séquenceur → Scheduler Exemple 2

- Exemple pour les machines M1 à M3
- Les données A à F sont réparties :
  - M_1 (A,B) M_2(C,D) M_3(E,F)
- T1 manipule (A,B) T2(A,C) T3(C,E) T4(E,A) T5(A,D)
- Les transactions arrivent sur :
  - M_1 : T1(A,B)
  - M_2 : T2(A,C), T3(C,E), T5(A,D)
  - M_3 : T4(E,A)

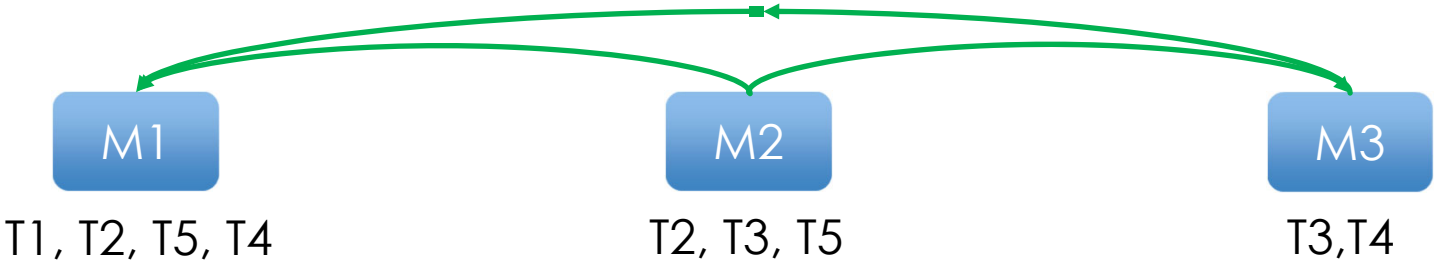


Séquenceur → Scheduler Exemple 2 (suite)

- Avant transmission

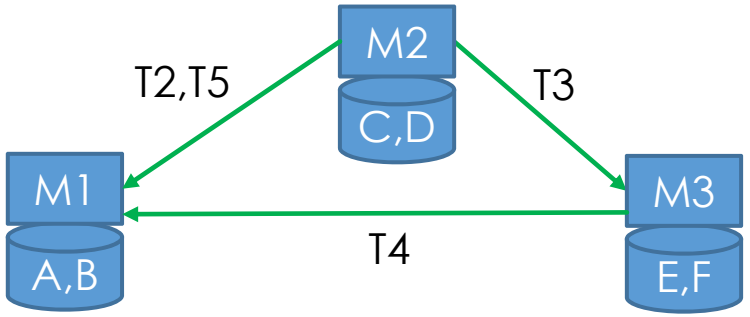


- **Après** transmission



Séquenceur → Scheduler Exemple 2 (suite)

- Transmission
  - M_1 : T1 reste locale pas d'envoi
  - M_2 : T2 et T5 vers M1, T3 vers M3
  - M_3 : T4 vers M1
- Les schedulers ordonnent seulement les transactions qu'ils reçoivent
  - M_1 : T1, T2, T5, T4
  - M_2 : T2, T3, T5
  - M_3 : T3, T4



Traiter les transactions

Scheduler : Traitement des transactions

- Planifie le traitement local des transactions
  - Ordonne les transactions reçues
- Détermine les données qu'une transaction lit ou écrit
  - Données lues = **R**ead set (R)
  - Données écrites = **W**rite set (W)
- Traitement local d'une transaction globale
  - Lit les données ($\in$ R) et les **envoie** aux machines concernées
  - **Reçoit** les données ($\in$ R) venant des autres machines
  - Traitement local
    - si la machine contient des données à écrire ($\in$ W)

Scheduler : Exemple

- Transactions : données lues (R) et écrites (W) pour T1(A,B) T2(A,C) T3(C,E) T4(E,A) T5(A,D) :
  - T1 : R = A, B W = A, B
  - T2 : R = A W = C
  - T3 : R = C, E W = E
  - T4 : R = E W = E, A
  - T5 : R = A, D W = A, D
- M1(A,B) : (T1, T2, T5, T4)
  - traiter T1
  - T2 : envoyer A vers M2
  - T5 : (A déjà envoyé à M2), recevoir D de M2, traiter T5
  - T4 : recevoir E de M3, traiter T4
- M2(C,D) : (T2, T3, T5)
  - T2: recevoir A de M1, traiter T2
  - T3: envoyer C vers M3
  - T5: envoyer D vers M1, recevoir A de M1, traiter T5
- M3(E,F) : (T3, T4)
  - T3: recevoir C de M2, traiter T3
  - Envoyer E vers M1, traiter T4 (inutile de recevoir A pour traiter T4 sur M3 car A n'est pas lu)

Scheduler : bilan

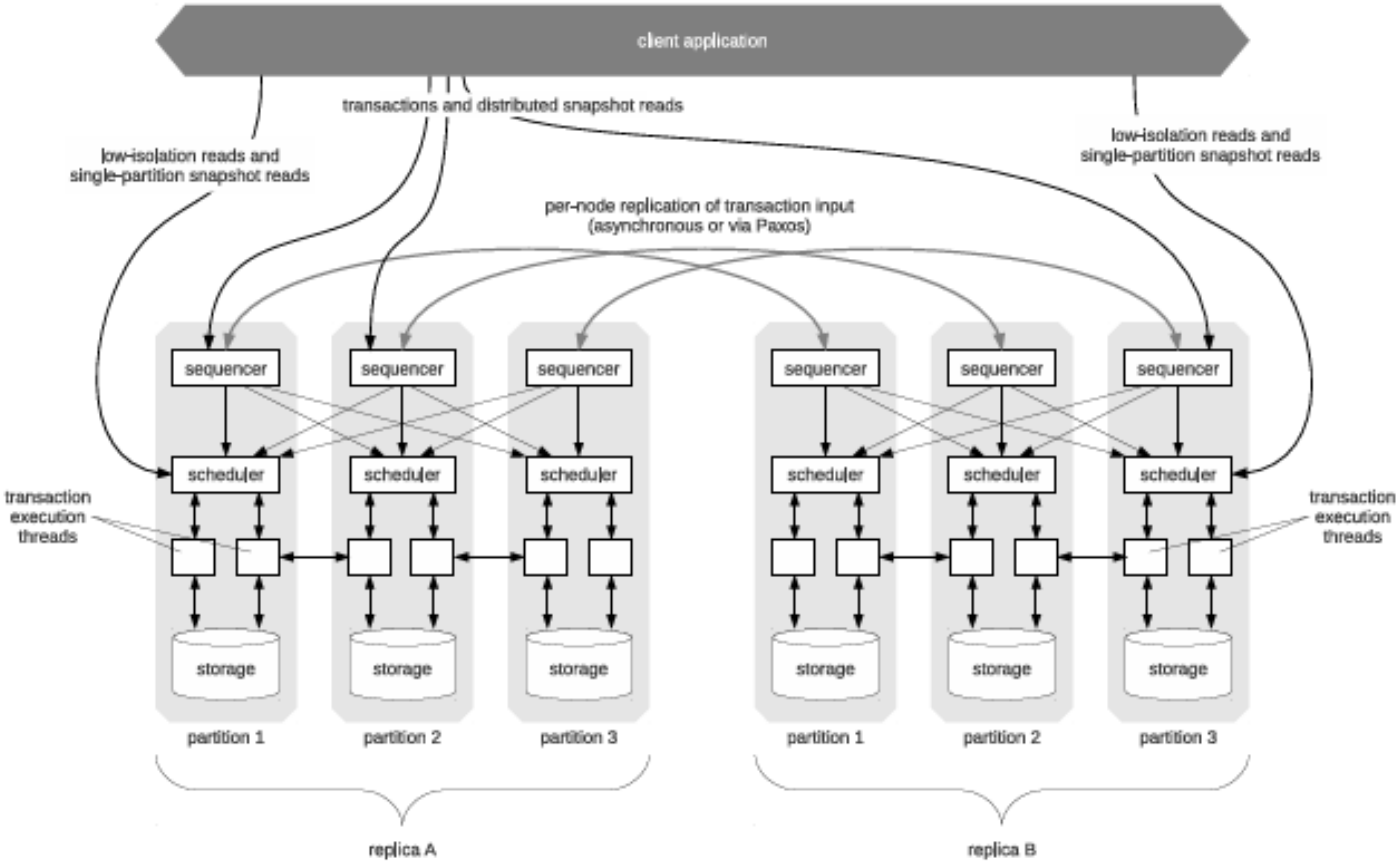
- Traiter les transactions (et seulement celles-ci) ayant au moins une donnée à écrire localement.
  - Exple: ne pas traiter T2 sur M1
- Envoyer les données d'une transaction à un site si et seulement si le site traite la transaction
  - Ne pas envoyer C à M1 car M1 ne traite pas T2
- Ne pas renvoyer plusieurs fois une donnée lue si elle n'a pas été modifiée entre temps (ex: A sur M1)
- Efficacité
  - Dépend du nombre de transactions multi-partitions
  - Dépend de la taille des données lues par les transactions multi-partitions

Ref bibliographique

- Calvin
  - Fast Distributed Transactions and Strongly Consistent Replication for OLTP Database Systems
  - http://cs-www.cs.yale.edu/homes/dna/papers/calvin-tods14.pdf
  - http://cs-www.cs.yale.edu/homes/dna/papers/calvin-sigmod12.pdf
    - Lire la section: Scheduler and concurrency control

Ref bibliographique

Figure extraite de l'article Sigmod 2012
Scalable transaction layer over shared nothing storage system



Références connexes

- Google Spanner
  - Spanner Becoming a SQL System (SIGMOD 2017)
    - https://ai.google/research/pubs/pub46103
  - TrueTime and External Consistency
    - https://cloud.google.com/spanner/docs/true-time-external-consistency
- Cornell Univ
  - LightWeight Multi-Key Transactions for Key-Value Stores (2015)

- Blog : **Spanner vs. Calvin**: Distributed Consistency at Scale
  - D. Abadi 2017
  - https://fauna.com/blog/distributed-consistency-at-scale-spanner-vs-calvin