SAM

Transactions déterministes

Etude du système Calvin

Hubert Naacke

MAI 2020

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 : on traite tout ou rien
    - **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 entièrement connues à l'avance

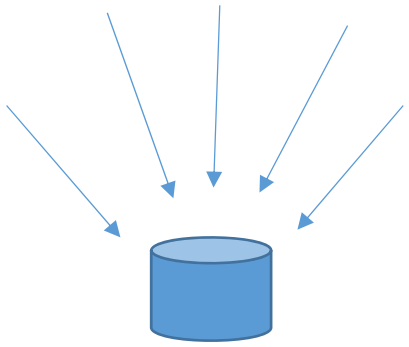
    ≠ Traitement offline d'un lot de transactions prédéfinies

Problème : 1 million de transactions par minutes ?

Table : Zone (nom, effectif)

Transaction : déplace(zoneDépart, zoneArrivée, quantité) :

Update Zone set effectif = effectif - quantité where nom = zoneDépart;
Update Zone set effectif = effectif + quantité where nom = zoneArrivée;

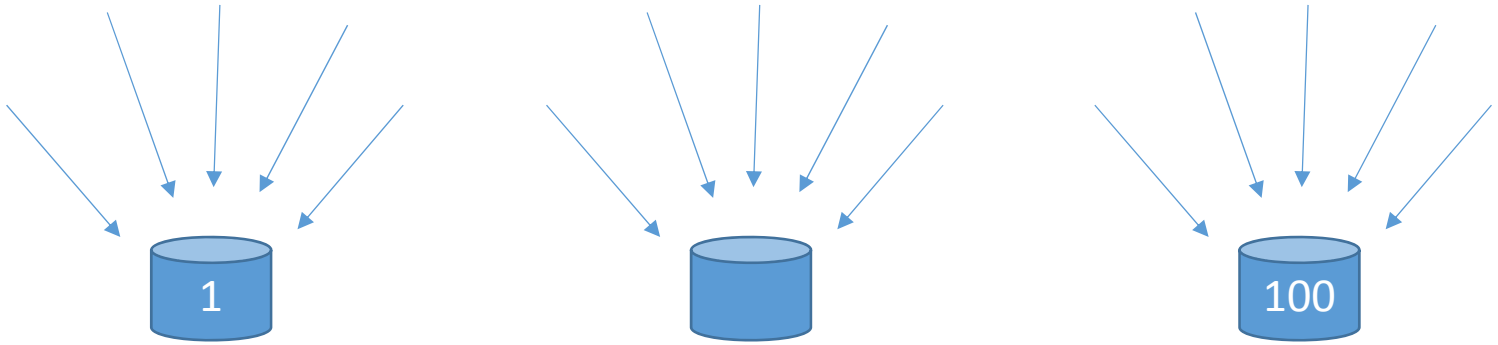


Zone (nom, effectif)

Limiter l'effectif : moins de 10 personnes par zone !

1 millions de transactions par minutes (tpm)?

- 1 super PC est trop faible, seulement 10 000 tpm
- Il faut 100 PC



Salle de sport

Nom	Effectif
Salle B	7
Gymnase	2

Nom	Effectif

Nom	Effectif
Salle A	9

Si 2 personnes vont de la salle A à la salle B, on a une transaction répartie…

Quelques cas d'usage

- Vente en ligne
  - Stock(produit, quantité), Panier(user, produit, quantité)
  - Garantir que tous les produits du panier sont en stock

- Telecom
  - Antenne(numAntenne, mobile)

  Pas plus de 100 mobiles par antenne

- Gaming
  - Echange d'objets entre joueurs

- Micro paiement

Calvin : Bibliographie

- Fast Distributed Transactions and Strongly Consistent Replication for OLTP Database Systems
  - TOSD 2014
    - http://cs-www.cs.yale.edu/homes/dna/papers/calvin-tods14.pdf
  - SIGMOD 2012
    - http://cs-www.cs.yale.edu/homes/dna/papers/calvin-sigmod12.pdf
    - Lire en particulier la section: Scheduler and concurrency control

- Extensions
  - **Q-Store**: Distributed, Multi-partition Transactions via Queue-oriented Execution and Communication
    - EDBT 2020
      - https://openproceedings.org/2020/conf/edbt/paper_39.pdf

Calvin : principes

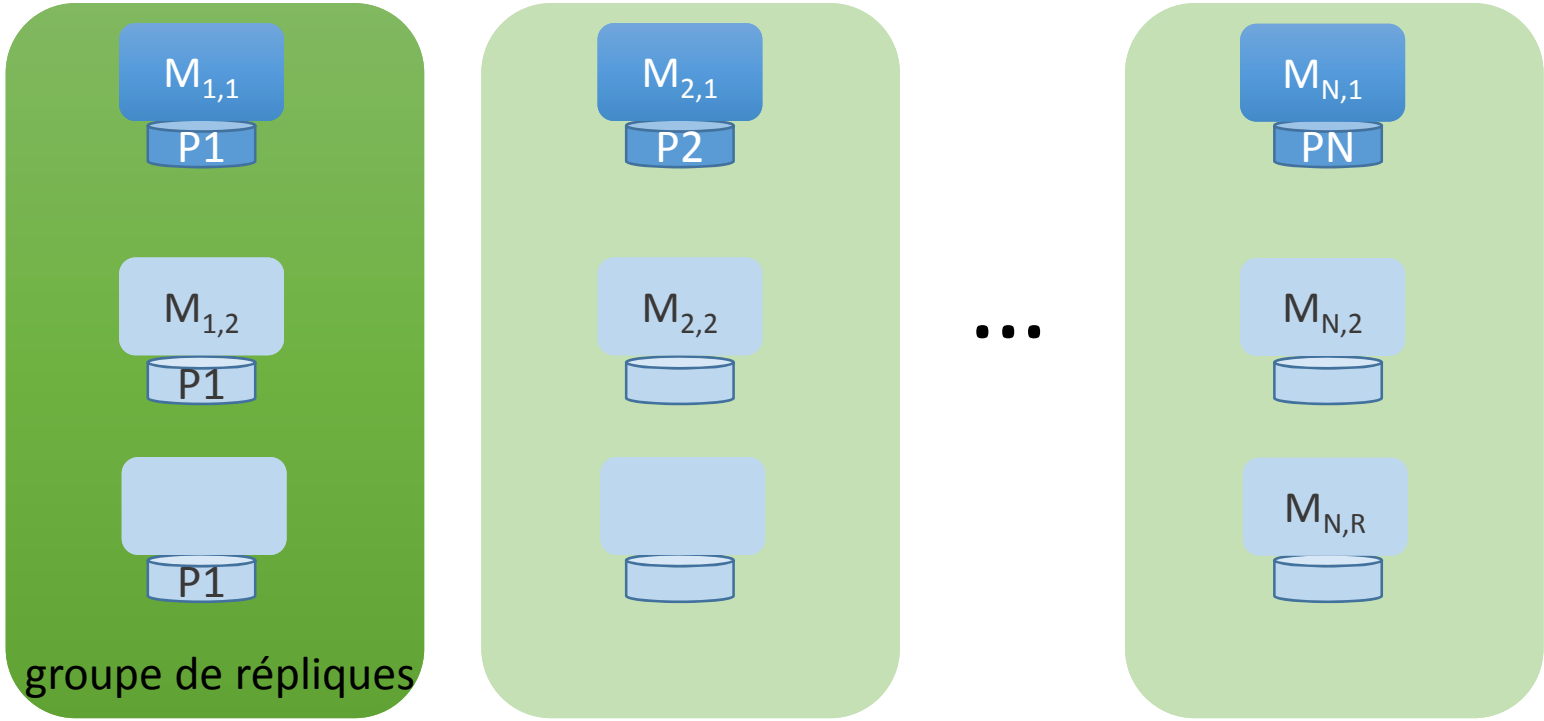
- Données SQL
- Transactions SQL courtes
  - durée bornée: nécessaire pour ordonner les transactions
- Système déterministe
  - Détermine l'ordre global des transactions **avant** de les traiter.
  - Traite les transactions dans l'ordre préalablement déterminé

- 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

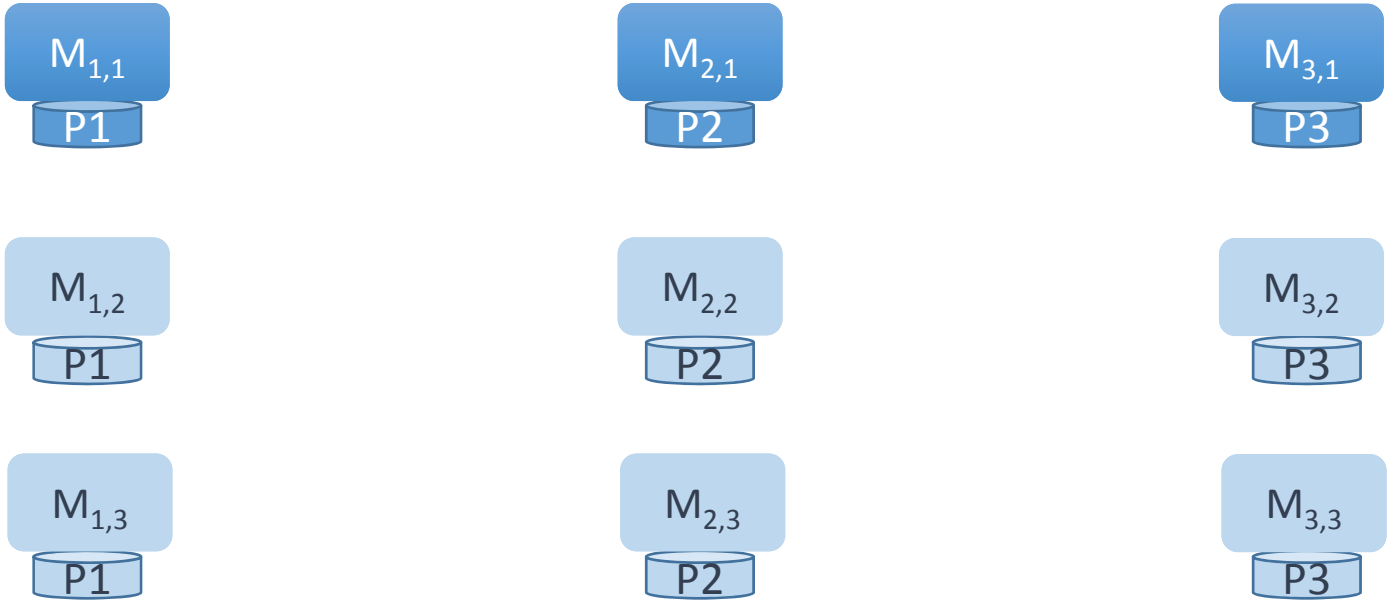
- Données partitionnées et répliquées sur un cluster de machines
  - N partitions: P_1 , ... P_i , ..., P_N
  - R répliques par partition

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



Architecture de Calvin

Exemple avec 3 partitions (N=3) et 3 répliques par partition (R = 3)



Cluster de 9 machines Mi,j

Traitement des transactions réparties

- Transaction multi-partitions
  - Accède à plusieurs partitions sur plusieurs machines
- Besoins
  - Protocole pour coordonner les machines
  - Traitement décentralisé
- Pour réaliser cette coordination chaque machine $M_{i,j}$ a
  - un **séquenceur** : ordonner les transactions
  - un **scheduler :** traiter les transactions

Rôle du Séquenceur

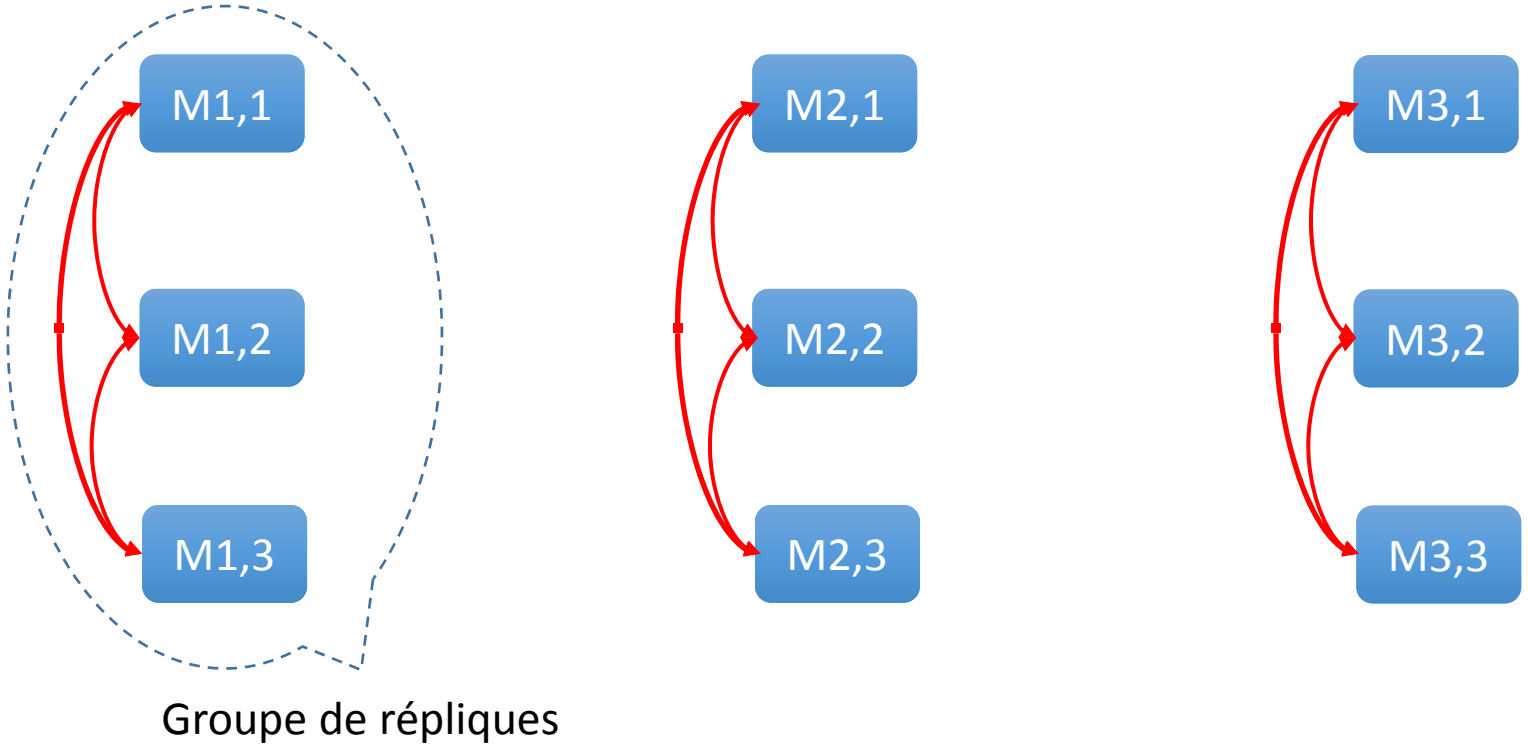
- Ordonner les transactions
- Objectif : Sur chaque partition, connaitre la liste des transactions qui accèdent à la partition

Séquenceur

- Découpe le temps en fenêtres de période fixe
- Chaque machine peut recevoir des transactions
- En fin de fenêtre courante, **propager** 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.

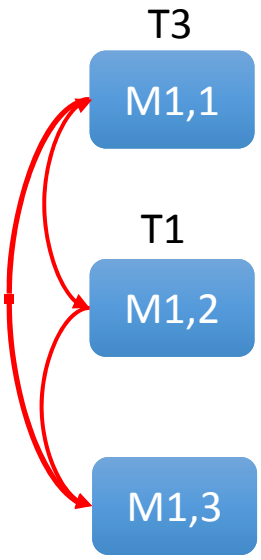
Séquenceur: exemple

- Séquenceur : coordination "verticale"

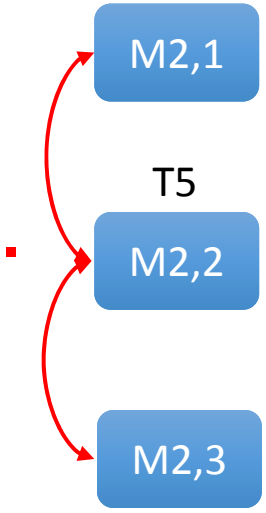


Séquenceur : exemple 1

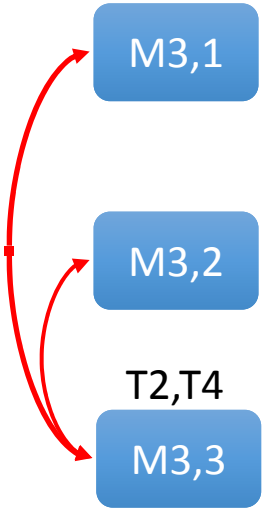
- Les transactions T1 à T5 arrivent pendant la fenêtre courante
- Puis coordination verticale en fin de fenêtre



T3 → M1,2 et M1,3
T1 → M1,1 et M1,3

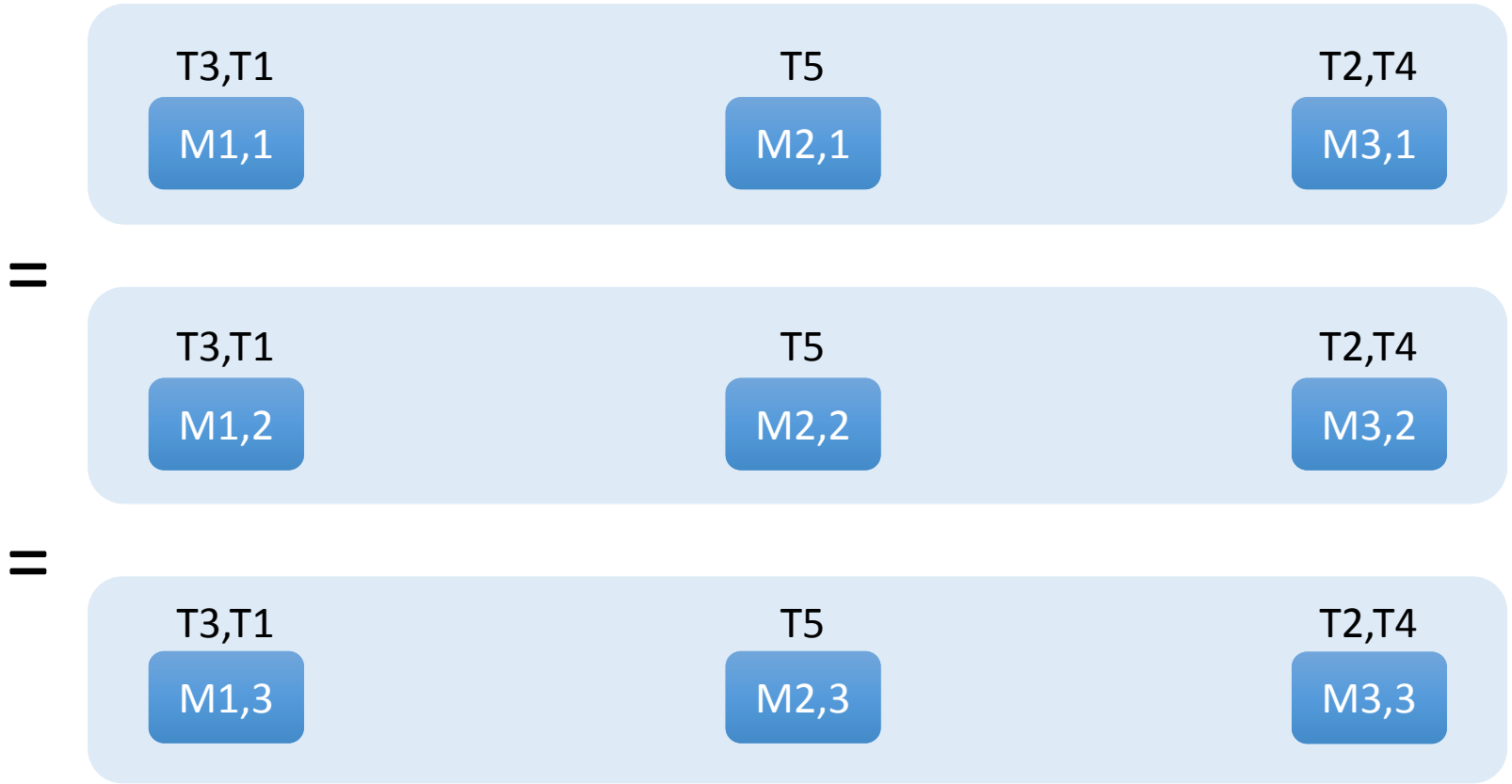


T5 → M2,1 et M2,3



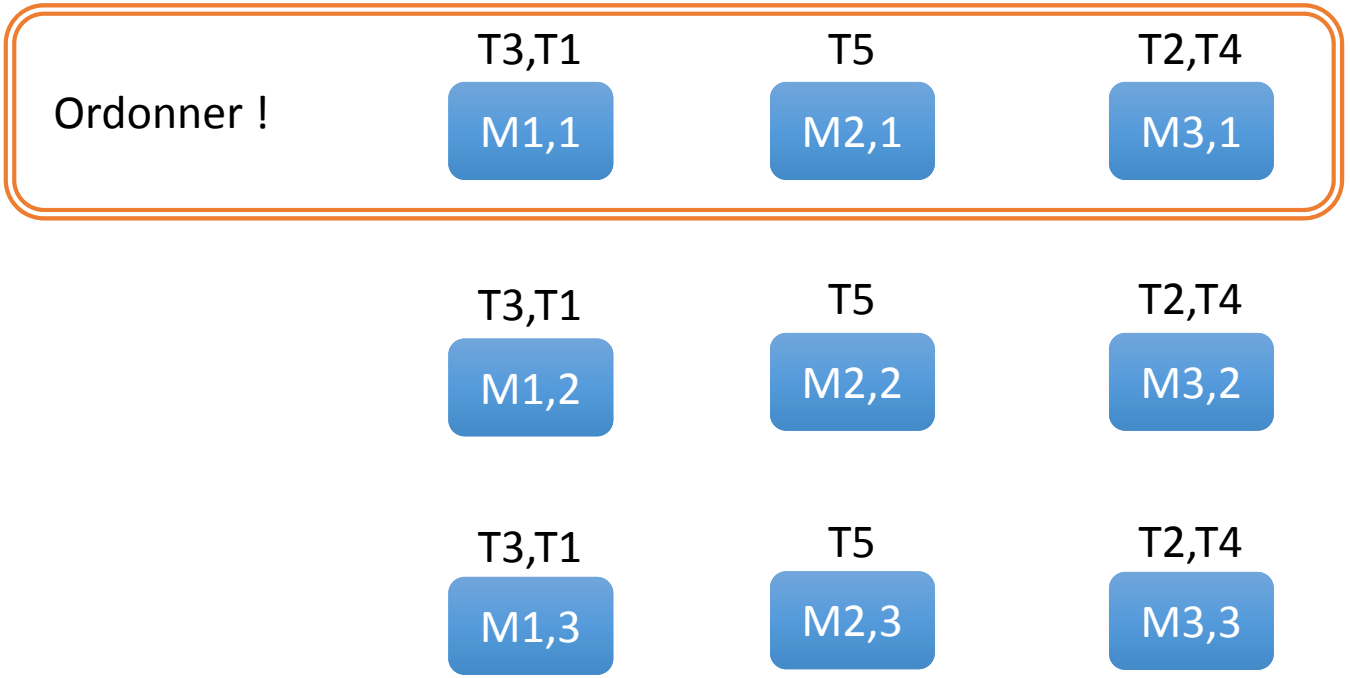
(T2,T4) → M3,1 et M3,2

Séquenceur : exemple 1 après coordination



Ordonner les transactions

- A « l'intérieur » d'un ensemble de partitions
- Communication horizontale sur une même "ligne"
- **Pas** de communication entre les répliques



Ordre global des transactions

- L'ordre global déterminé de manière **décentralisée**

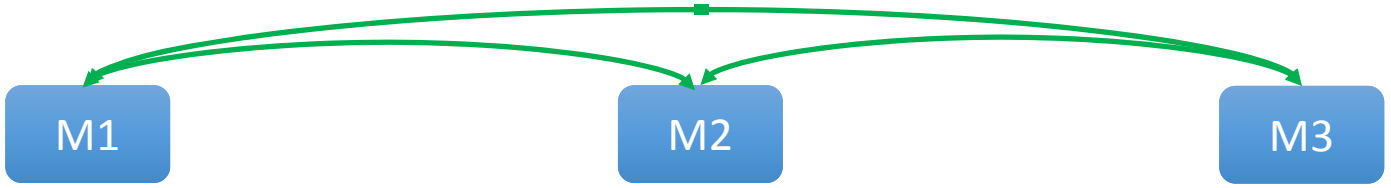
- Ordre global des séquenceurs = numéro de machine
  - M1 < M2 < … Mi < Mj …. < M_N
  - Pour tout i < j : une transaction posée sur Mi **précède** une transaction posée sur Mj

- Ordre des transactions arrivées sur la même machine
  - Si i = j alors Ta précède Tb si a < b

Messages pour déterminer l'ordre

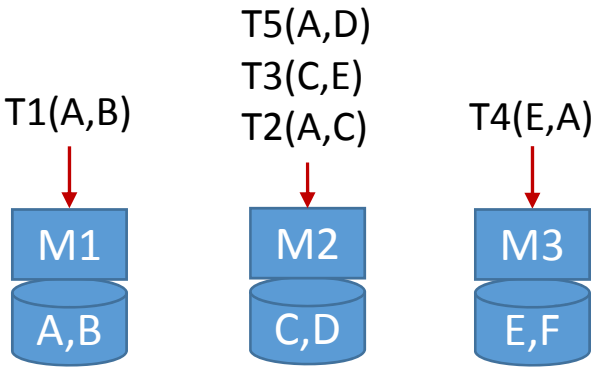
- Compléter la liste des transactions, connue sur chaque machine, pour tenir compte des transactions **multi-partitions**
- Transmettre les transactions aux seules machines concernées
- Chaque transaction est transmise

  sur chaque machine stockant au moins une donnée de la transaction.



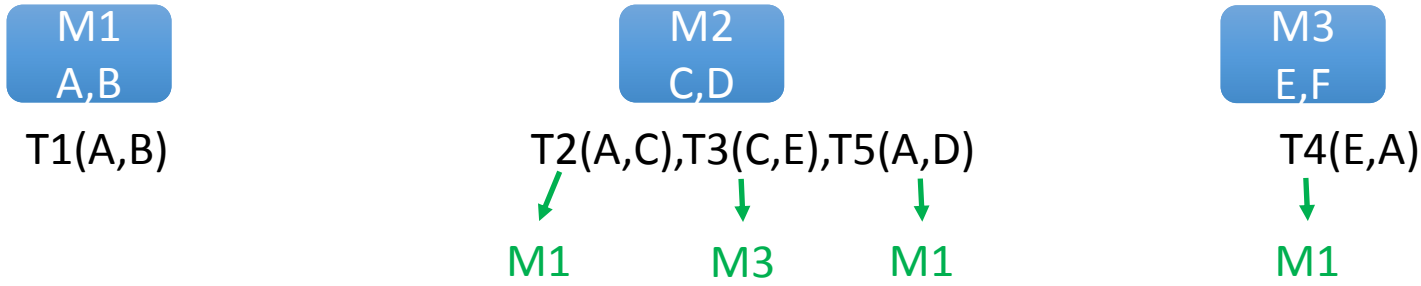
Messages entre séquenceurs : Exemple 2

- 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)

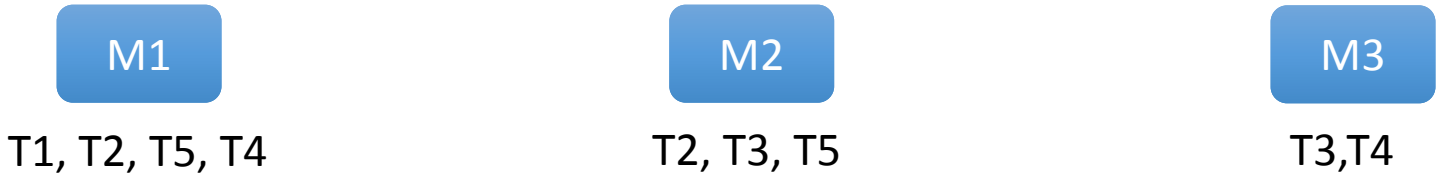


Exemple 2 (suite)

- Avant transmission

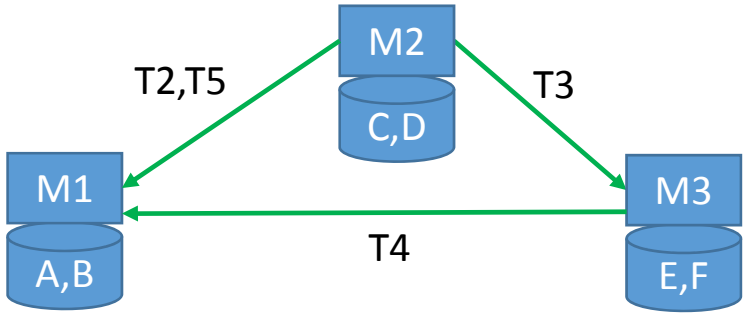


- **Après** transmission



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 machines ordonnent seulement les transactions qu'elles reçoivent
  - M_1 : T1, T2, T5, T4
  - M_2 : T2, T3, T5
  - M_3 : T3, T4



Rôle du scheduleur

=

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

Scheduler : bilan

- Ne pas traiter une transaction dont les données à écrire ne sont pas locales (ex: T2 sur M1)
- Ne pas envoyer une donnée à un site qui ne fait que lire les données de la transaction sans la traiter
  - 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

Conclusion

- Traitement décentralisé des transactions réparties
- Tolérant aux pannes
- Très rapide
  - Transactions = procédure stockée
  - Ralentissement si taux de transactions réparties > 10%
- A inspiré plusieurs systèmes NewSQL

Annexe

Architecture de Calvin

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