A Database State Machine Approach

Luís Silva – m2086

Fábio Beirão – m2199

Introdução

- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer eget lacus vitae lorem varius pretium quis nec diam. Nulla nec viverra tellus. Nullam sodales venenatis quam eget convallis. Curabitur id justo nisl, eget varius nulla. Aenean massa mauris, egestas vel varius sit amet, aliquet et urna. Nullam a felis ligula. Maecenas adipiscing enim in est posuere vulputate ac id sem. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus fermentum, nibh ut facilisis suscipit, mauris augue feugiat felis, at blandit diam neque sit amet nibh. Duis condimentum enim eu turpis congue pharetra. In hac habitasse platea dictumst. Fusce ultrices ullamcorper aliquam. Donec eget nulla risus. Aliquam erat volutpat. Pellentesque elementum tempus sem vitae ultrices. Curabitur placerat erat quis enim scelerisque non volutpat neque egestas. Aenean egestas vestibulum laoreet.
- Sed mollis arcu non enim volutpat rutrum. Nulla nunc turpis, feugiat non pulvinar ac, venenatis sit amet est. Fusce posuere dui non neque tempor luctus. Pellentesque sed ipsum ac sapien dignissim mattis interdum quis massa. Ut porttitor enim vel augue iaculis sodales. Curabitur volutpat, ipsum non congue molestie, diam ante aliquam leo, ut iaculis nunc lacus eget lacus. Mauris venenatis lacus quis arcu convallis sed rutrum massa interdum. Duis non tortor non mauris laoreet blandit. Mauris ultricies hendrerit mauris ut consequat. Morbi commodo enim urna. Pellentesque rhoncus feugiat felis consequat aliquet. Nam tempor tortor ac lacus blandit quis dignissim est elementum. Quisque non laoreet nibh. Fusce ligula dolor, vehicula eu scelerisque ac, fermentum at metus. Vestibulum ante mauris, cursus et tincidunt et, ornare non justo. In iaculis arcu lobortis lectus tincidunt vel molestie tellus venenatis. Aenean porta augue non leo posuere porttitor. Integer interdum tortor quis felis suscipit vitae cursus lectus lobortis. Vestibulum at ligula sed erat blandit hendrerit sit amet vel lacus.
- Etiam gravida leo eget sem consectetur nec vestibulum tellus pretium. Nulla ultrices, purus a rhoncus scelerisque, lorem orci dignissim ipsum, sit amet tincidunt sem elit non turpis. Ut enim metus, auctor ut aliquet eget, lobortis et eros. Ut tellus tortor, commodo sed rhoncus ut, pretium quis ligula. Sed eleifend facilisis elit, nec egestas felis tristique vitae. Cras quis lorem nec nibh vehicula rutrum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam vitae urna in nibh sagittis egestas sit amet non risus. Proin pellentesque ante sagittis magna semper bibendum. In eleifend augue quis ante pretium tempus. Nam elementum dictum purus, sed sodales arcu sagittis et.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean at quam elit. In vel justo eu nunc hendrerit semper quis a dui. Aliquam erat volutpat. Sed quis risus felis. Sed vestibulum consequat tincidunt. Fusce sem odio, fringilla non aliquam sed, ultrices et mi. Praesent lectus ante, feugiat sed malesuada ac, eleifend ac sem. Cras suscipit nisi orci, quis interdum lacus. Aliquam ipsum lorem, laoreet ut tincidunt eget, aliquet non purus. Maecenas ullamcorper, nunc quis eleifend bibendum, lorem orci pretium nibh, in mollis nulla orci id felis. Morbi tortor mi, placerat eget scelerisque vitae, iaculis a nunc. Aenean eu ante felis, quis cursus tellus. Sed scelerisque feugiat sem, vel rutrum nisi varius non.
- Sed mollis arcu non enim volutpat rutrum. Nulla nunc turpis, feugiat non pulvinar ac, venenatis sit amet est. Fusce posuere dui non neque tempor luctus. Pellentesque sed ipsum ac sapien dignissim mattis interdum quis massa. Ut porttitor enim vel augue iaculis sodales. Curabitur volutpat, ipsum non congue molestie, diam ante aliquam leo, ut iaculis nunc lacus eget lacus. Mauris venenatis lacus quis arcu convallis sed rutrum massa interdum. Duis non tortor non mauris laoreet blandit. Mauris ultricies hendrerit mauris ut consequat. Morbi commodo enim urna. Pellentesque rhoncus feugiat felis consequat aliquet. Nam tempor tortor ac lacus blandit quis dignissim est elementum. Quisque non laoreet nibh. Fusce ligula dolor, vehicula eu scelerisque ac, fermentum at metus. Vestibulum ante mauris, cursus et tincidunt et, ornare non justo. In iaculis arcu lobortis lectus tincidunt vel molestie tellus venenatis. Aenean porta augue non leo posuere porttitor. Integer interdum tortor quis felis suscipit vitae cursus lectus lobortis. Vestibulum at ligula sed erat blandit hendrerit sit amet vel lacus.
- Etiam gravida leo eget sem consectetur nec vestibulum tellus pretium. Nulla ultrices, purus a rhoncus scelerisque, lorem orci dignissim ipsum, sit amet tincidunt sem elit non turpis. Ut enim metus, auctor ut aliquet eget, lobortis et eros. Ut tellus tortor, commodo sed rhoncus ut, pretium quis ligula. Sed eleifend facilisis elit, nec egestas felis tristique vitae. Cras quis lorem nec nibh vehicula rutrum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam vitae urna in nibh sagittis egestas sit amet non risus. Proin pellentesque ante sagittis magna semper bibendum. In eleifend augue quis ante pretium tempus. Nam elementum dictum purus, sed sodales arcu sagittis et.

Introdução (agora a sério)

- Replicação por software é uma forma barata de fornecer alta disponibilidade, que de outra forma seria cara com replicação por hardware.
- No entanto, os modelos de replicação síncrona de um sistema com uma boa performance ainda estão a ser desenvolvidos e investigados.
- Neste *paper* é proposta uma nova abordagem para replicação **síncrona** de bases de dados num cluster de servidores.

A abordagem proposta

- Baseada em máquinas de estados, difere dos mecanismos de replicação tradicionais pois não lida com a replicação utilizando mecanismos transaccionais distribuídos (tais como o *commit* atómico).
- Técnica de *actualização diferida*. De acordo com esta técnica, as actualizações são processadas localmente num dos servidores.

Vantagens da replicação diferenciada

- Melhor performance (agrega e propaga múltiplas actualizações em simultâneo, diminuindo assim o número de mensagens na rede).
- Melhor suporte a tolerância a falhas (actualizações em falta podem ser tratadas pelo módulo de comunicação como mensagens perdidas).
- Menor taxa de *deadlocks* (por eliminar os *deadlocks* distribuídos).

Um problema da replicação diferenciada

- Um dos grandes problemas da replicação diferenciada é que a falta de sincronização durante a execução das transacções pode levar a uma elevada taxa de *aborts*.
- Neste *paper* é mostrado como a abordagem baseada em máquinas de estados pode reduzir a taxa de *aborts* de transacções.

Método

O método proposto por este *paper* é composto por um conjunto de elementos, que comunicam entre eles por mensagens, não partilhando memória nem dispondo de um relógio comum.

O conjunto é particionado nos elementos:

- Clientes
- Bases de Dados

Os elementos *Base de Dados* são réplicas completas e actualizadas da BD principal.

Métodos

- State machine approach (SMA)

  vai definir a estratégia de replicação

- Atomic Broadcast

  mecanismo usado para implementar o SMA

State Machine Approach

- Utiliza os princípios da Replicação Activa.
- É uma técnica de coordenação de replicação não centralizada.
- Todas as réplicas recebem e processam a mesma sequência de pedidos.
- Vão também ter um comportamento determinístico, onde recebem e produzem o mesmo I/O.

State Machine Approach

Os pedidos nas réplicas tem de obedecer a dois requisitos:

- Acordo – Todas as réplicas recebem todos os pedidos.
- Ordem – Se é processado o pedido p1 e em seguida p2, nenhuma réplica pode processar p2 e depois p1.

Nota: A ordem pode não ter relevância, se o p1 e p2 não tiverem relação entre si, ou seja, o I/O de cada um não influencia o outro.

Atomic Broadcast

Os envios para as réplicas têm de obedecer a três requisitos:

- Acordo – Se a mensagem M é entregue por um elemento, então todos os elementos entregam a mensagem M.
- Ordem – A entrega é feita na mesma ordem em todos os elementos.
- Terminação – Caso um elemento transmita uma mensagem (sem falhas), todos os elementos recebem a mensagem.

Nota: a **recepção** da mensagem não significa a **entrega** ao elemento, pois pode existir a verificação do atomic broadcast (garantindo a entrega no global), antes da entrega no elemento.

Atomic Broadcast

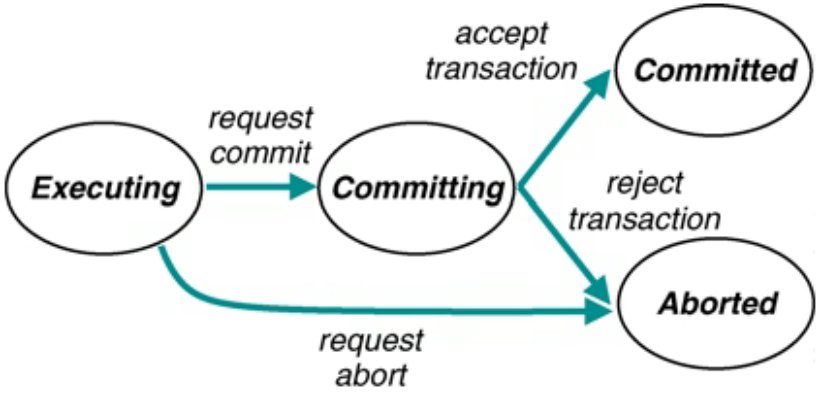
No caso de falha (crash) do elemento, é importante existir um acordo e ordenação que contemple a recuperação desse elemento aquando a sua recuperação.

Depois de recuperar duma falha de tempo T, é necessário fazer a entrega de todas mensagens que surgiram durante esse período T, obedecendo aos requisitos de ordem e terminação.

Transacções

Estados de uma transacção:

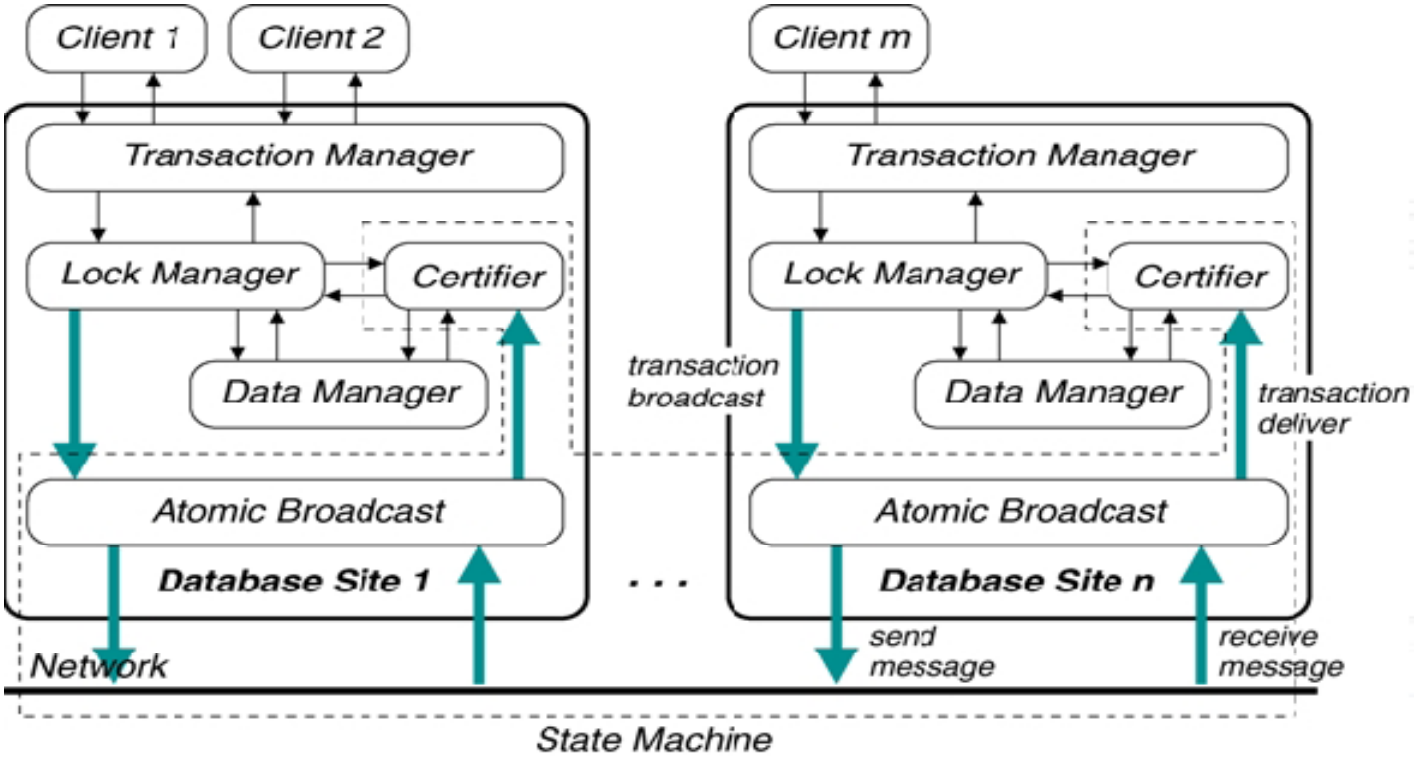
- *Executing*: as operações de leitura e escrita são executadas na BD onde foi iniciada.
- *Commiting*: quando o cliente pede o *commit* (a transacção é enviada para as outras BD's).
- Dependendo se a BD aceita ou aborta a transacção passa para o estado *Commited* ou *Aborted*



Transacções

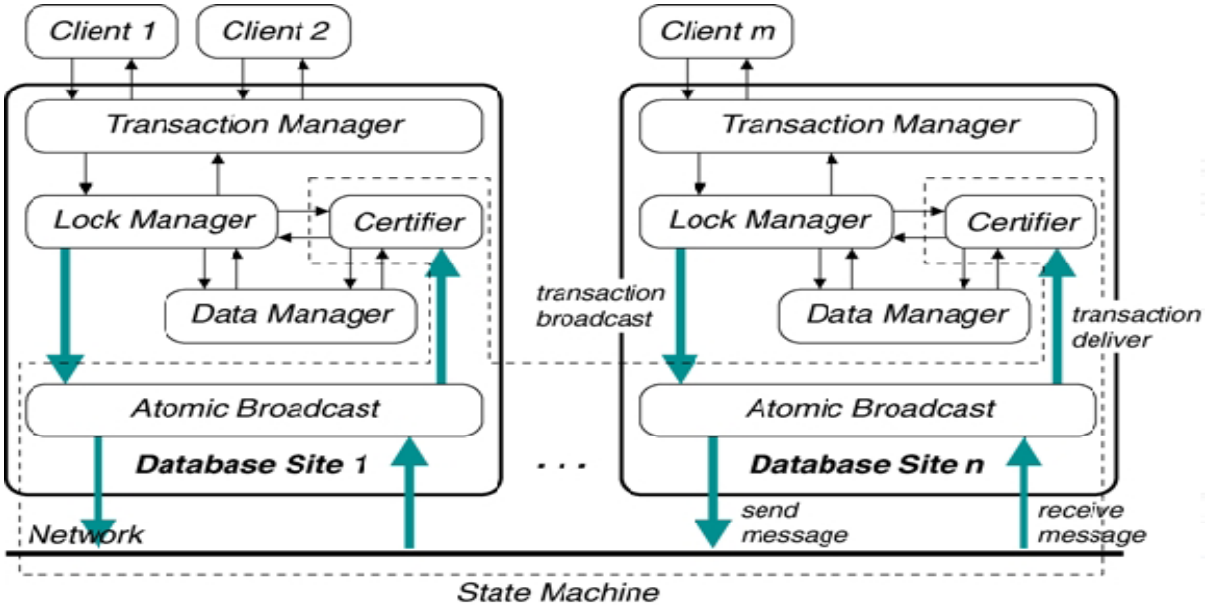
Ter em conta: commit´s, updates, etc… são feitos a todos os elementos BD em broadcast.

As transacções nas BDs são feitas segundo os módulos:



Transacções - 1º passo:

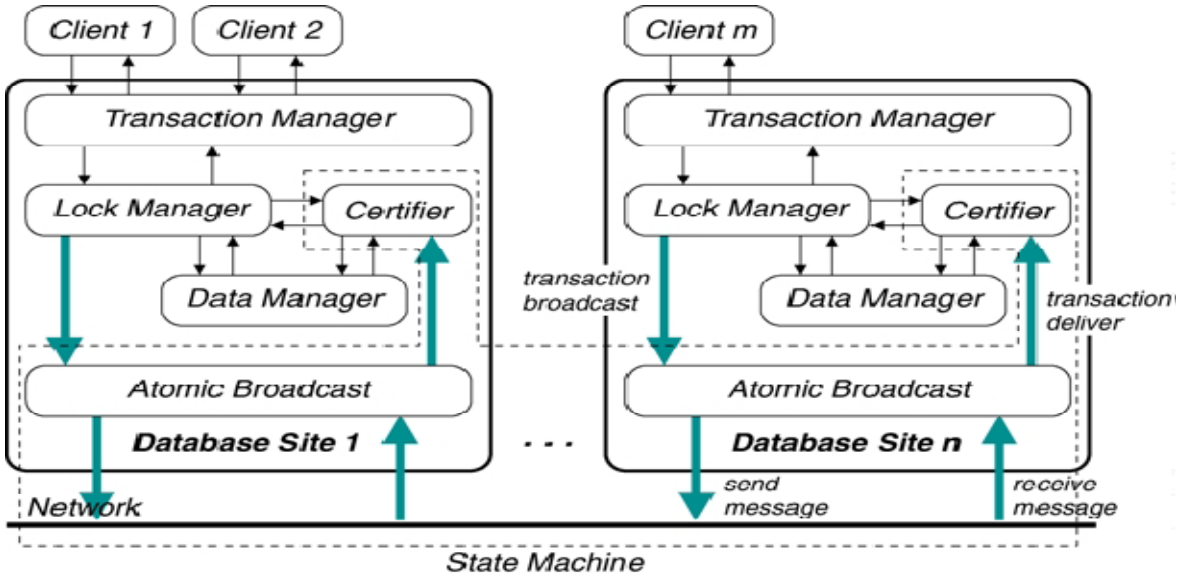
- Os elementos Clientes ligam-se ao Transaction Manager que faz a interacção entre os elementos;
- Lock Manager envia a transacção ao Certifier;
- Certifier verifica/testa a veracidade das transacções na BD.



Transacções - 2º passo:

(Transacção validada)

- É reposta a transacção pelo Lock Manager, onde faz o lock dos recursos na BD (passando para o estado *committing*).
- É transmitida a transacção a todos os elementos BD, através de mensagens do Atomic Broadcast (AB).



Transacções - 3º passo:

- Recepção nos elementos pelo AB, passando ao Certifier e feito o lock dos recursos na BD (passando para o estado *committing*).
- Certifier verifica que todos os elementos BD alcançaram o mesmo estado (*committing*).
- Caso se verifique: É então feita a transacção em todos os elementos (passando para o estado *commited*).
- Certifier verifica que todos os elementos alcançaram o mesmo estado final (*committed*) (Ordem de escrita, todos foram alterados, etc..) .

Posso fazer sempre o *Lock*?

Para uma requisição de recurso (exemplo: para escrita na BD), a confirmação do lock é feita se não existir conflito* com uma transacção:

- em curso nos outros elementos BD;
- em curso no próprio elemento;
- que está no estado *committing* (passou no Certifier e tem o lock já feito, encontrando-se à espera da confirmação).

*requere os mesmos recurso

Nota: Os conflitos de escrita são evitados se a alocação for feita também por ordem em todos. No entanto, isto que leva a uma grande concorrência/indisponibilidade dos Certifiers.

Certifier com espera e ordenação

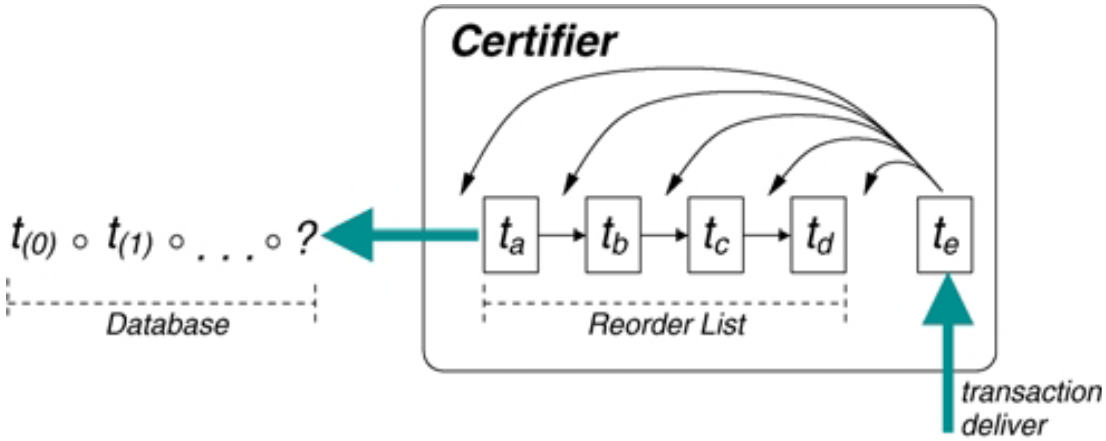
Na execução das transacções nas BD (state committing -> state commited), existem muitos *aborts* devido à inexistência de sincronização dos elementos BD.

A solução está em reordenar as transacções que esperam passar ao estado final commited.

Então a ordem de execução pode ou não ser a ordem de entrega ao Certifier.

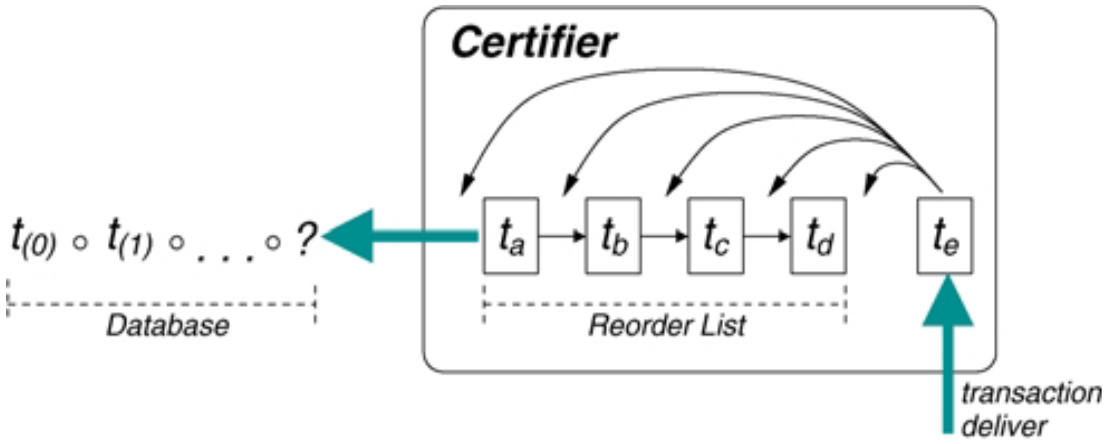
Certifier com espera e ordenação

- Construir uma *Reorder List* – contém as transacções em espera para modificação da BD. Mais as transacções que não foram vistas, pois o elemento já tinha uma transacção em curso.
- Existe um factor para a *Reorder List* – número máximo de transacções em espera no Certifier.



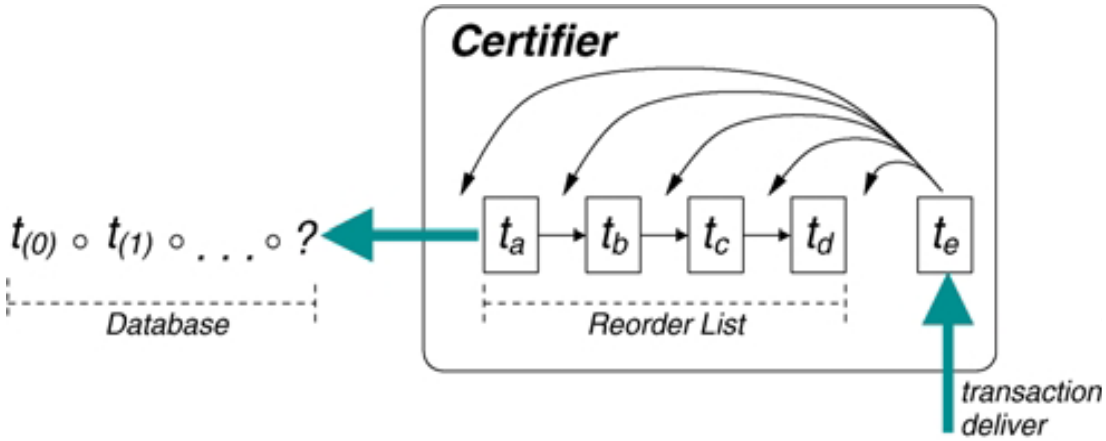
Certifier com espera e ordenação

- A cada nova chegada de transacções, é retirada a transacção mais à esquerda lista (se passar o factor do numero máximo);
- Executada na BD a transacção retirada e libertados os seus recursos;



Certifier com espera e ordenação

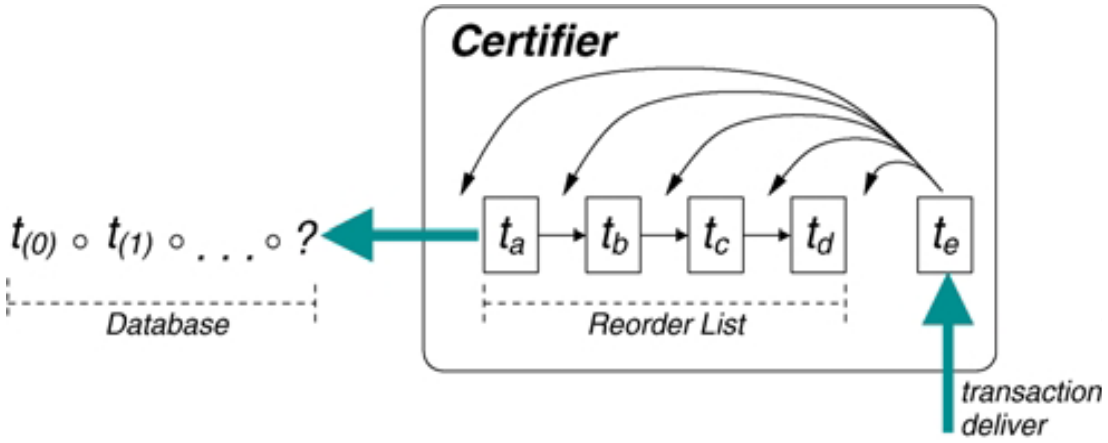
- É verificada a lista caso exista alguma transacção *T* que estava à espera dos recursos, agora libertados:
  - Caso sim, é retirada *T* da lista executada também e assim sucessivamente;
  - Caso não, a transacção entregue entra para o fim da lista.



Certifier com espera e ordenação

Vantagem: as transacções não são abortadas tão facilmente devido à falta de recursos ou sua utilização;

Desvantagem: os recursos podem estar alocados mais tempo (em caso de poucas entregas);



Conclusões

- Os métodos apresentados por este paper servem para aumentar a tolerância a falhas (ao aumentarem a disponibilidade) e a performance (por balancearem a carga pelos servidores).

- Por toda a comunicação ser encapsulada pelo módulo do broadcast atómico, os aborts por falta de sincronização são significativamente reduzidos (nos testes de 20% para <5%).

Bibliografia

Fernado Pedone, Rachid Guerraoui e André Schiper, “ The Database State Machine”, em Kluwer Academic Publishers, Distributed and Parallel Databases, cap.14, pag. 71–98, 2003.

V. Hadzilacos e S. Toueg “Fault-Tolerant Broadcasts and Related Problems”, em Distributed Systems, 2ª ed., cap 3, 1993.

F. Pedone, R. Guerraoui, e A. Schiper, “Transaction reordering in replicated databases,” em 16th IEEE Symposium on Reliable Distributed Systems, Durham, Out. 1997.

F.B. Schneider, “Implementing fault-tolerant services using the state machine approach: A tutorial,” em ACM Computing Surveys, vol. 22, nº 4, pag. 299–319, Des. 1990.