Replicación y consistencia

Tema 4

Modelos de replicación

Índice

Bibliografía

[BMST93] N. Budhiraja, K. Marzullo, F. B. Schneider, S. Toueg: “The Primary-Backup Approach”. En S. J. Mullender, editor, “Distributed Systems”, capítulo 8, págs. 199-216. Addison-Wesley, 2ª edición, 1993.

[DSS98] Xavier Défago, André Schiper, Nicole Sergent: “Semi-passive Replication”, IEEE SRDS, pgs. 43-50, West Lafayette, IN, EE.UU., octubre 1998.

[HT93] Vassos Hadzilacos, Sam Toueg: “Fault-Tolerant Broadcasts and Related Problems”. En S. J. Mullender, editor, “Distributed Systems”, capítulo 5, págs. 97-145. Addison-Wesley, 2ª edición, 1993.

[Lam98] Leslie Lamport: “The Part-Time Parliament”. ACM Trans. Comput. Syst. 16(2): 133-169 (1998)

[Pow94] David Powell: “Distributed Fault Tolerance: Lessons from Delta-4”, IEEE Micro, 14(1), pgs. 36-47, febrero 1994.

[Sch93] Fred B. Schneider: “Replication Management Using the State-Machine Approach”. En S. J. Mullender, editor, “Distributed Systems”, capítulo 7, págs. 166-197. Addison-Wesley, 2ª edición, 1993.

[WS05] M. Wiesmann, A. Schiper: “Comparison of Database Replication Techniques Based on Total Order Broadcast”, IEEE Trans. on Know. and Data Eng., 17(4):551-566, abril 2005.

Índice

1. Concepto y objetivos

- La replicación es una técnica consistente en mantener más de una copia de un mismo componente o dato en una aplicación.
- El uso de cachés es un caso especial de replicación, que tiene como objetivo mejorar la escalabilidad (tanto de servicio, como geográfica).

1. Concepto y objetivos

- En un sistema distribuido se utiliza la replicación para:
  - Evitar que un fallo en una de las réplicas provoque un fallo en el componente replicado.
    - Si las réplicas se han distribuido convenientemente será muy improbable que más de una falle simultáneamente.
    - Con ello, se garantiza que el componente replicado jamás falle.
  - Incrementar la capacidad de servicio.
    - Sólo ocurre con las peticiones que no impliquen una modificación del estado del componente replicado.
    - Hay muchas aplicaciones con alto porcentaje de operaciones de lectura.

1. Concepto y objetivos

- Sin embargo, la replicación tiene un coste generalmente alto:
  - Hay que controlar con cierta frecuencia el estado de cada réplica.
  - Deben soportarse las incorporaciones de nuevas réplicas y el fallo de las existentes.
  - Deben propagarse las actualizaciones a todas las réplicas, ofreciendo un modelo de consistencia adecuado.

1. Concepto y objetivos

- Por todo esto, deben diseñarse protocolos para:
  - Gestionar adecuadamente la consistencia. Esto implica:
    - Propagación de actualizaciones.
    - Filtrado o bloqueo de accesos que sólo impliquen lectura.
  - Añadir réplicas. Puede necesitar el bloqueo de las peticiones en curso.
  - Recuperar réplicas previamente caídas.
    - Depende de la cantidad de estado que tenga el componente replicado. Si hubiera poco, bastaría con descartarlo y propagarlo entero.
    - Los protocolos de recuperación (y/o reconciliación, según el modelo de particionado) no deben implicar una pérdida de disponibilidad.

Índice

2. Modelos de replicación

- Existen diferentes técnicas para replicar un determinado componente o servicio.
- Las más comunes son:
  - Replicación pasiva (o replicación primario-backup).
  - Replicación activa (o máquina de estados).
  - Replicación semipasiva.
  - Replicación semiactiva.

2. Modelos de replicación

- Cada modelo se caracteriza por:
  - El número de réplicas que pueden servir directamente las peticiones de los clientes.
  - La forma de recibir las peticiones.
  - La forma de propagar las actualizaciones.
  - La estrategia para soportar caídas de las réplicas.
  - Su capacidad para servir operaciones cuya ejecución no sea determinista.

Índice

2.1. Replicación pasiva
[BMST93]

- Todos los clientes envían sus peticiones a una misma réplica del objeto: réplica primaria.
- La réplica primaria es la única que sirve directamente la petición y actualiza de inmediato su estado.
- No puede haber más de una réplica primaria en cada momento.
- Las peticiones recibidas (erróneamente) por otras réplicas deben ser:
  - Descartadas, avisando al cliente de ello, o...
  - Reenviadas a la réplica primaria.

2.1. Replicación pasiva

- En algunos sistemas se admite que las réplicas secundarias procesen las peticiones de solo lectura.
- Periódicamente o bien antes de responder al cliente en cada petición que modifique su estado, la réplica primaria propaga las actualizaciones a las demás réplicas.
- El resto de réplicas se llaman réplicas secundarias.
- Cuando hay una caída de la réplica primaria debe elegirse de entre las secundarias una para reemplazarla y tomar el rol de primaria.

2.1. Replicación pasiva

- No será fácil asegurar que no se pierda ninguna petición en caso de fallo.
  - El cliente debe reintentar las peticiones que no sean contestadas por la réplica primaria.
  - Ese reintento se realizará sobre la nueva réplica primaria.
  - Los secundarios deben saber qué peticiones fueron atendidas, para no servirlas múltiples veces.

2.1. Replicación pasiva

- Ventajas:
  - No sobrecarga el sistema. Sólo una réplica procesa las peticiones.
    - Pero podría saturarse con facilidad.
    - Para evitar la saturación, los servicios deben ser ligeros.
      - Múltiples servicios deberían utilizar un mismo conjunto de réplicas.
      - Cada servicio usará un primario diferente.
      - Así se equilibrará la carga.
  - No necesita protocolos de difusión ordenada.
    - Sólo hay un emisor. El orden es fácil de establecer.
    - Basta con numerar consecutivamente los mensajes en que se propague estado a los secundarios.

2.1. Replicación pasiva

- Ventajas (continuación):
  - Soporta sin problemas la ejecución de operaciones no deterministas.
    - Esas operaciones las ejecuta únicamente la réplica primaria. La decisión que tome en una bifurcación no será discutida por otras réplicas.
    - Como propagamos el estado resultante, en lugar de la operación, los secundarios no podrán generar inconsistencias.
  - Mecanismos de control de concurrencia fáciles de implantar, pues son locales y se utilizan en la réplica primaria.

2.1. Replicación pasiva

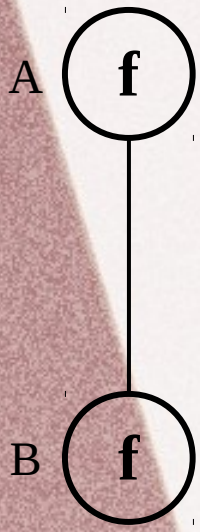
- Inconvenientes:
  - Reconfiguración costosa:
    - Necesidad de elección de un nuevo “líder”.
    - Cambio de rol en al menos una réplica.
    - Podrían perderse peticiones en curso.
      - El cliente reintentará la petición.
      - Se incrementa el tiempo de respuesta si hay fallos.
  - Código diferente para cada rol: primario y secundario se comportan de distinta manera.

2.1. Replicación pasiva

- Inconvenientes (continuación):
  - No se soportan los modelos de fallos más severos [HT93].
    - Los fallos arbitrarios no pueden ser gestionados.
      - Cuando el primario fallara de esta manera, el cliente no podría detectarlo.
    - Los fallos de omisión general no siempre se soportarán adecuadamente.
      - Si ocurren “f” fallos simultáneos de este tipo, se necesitan más de 2f réplicas para soportar esa situación sin promocionar erróneamente a un segundo primario.

2.1. Replicación pasiva

- **Omisión general**: n>2f.



- Asumiremos que existen 2f réplicas que se han agrupado en dos conjuntos disjuntos A, B tal como muestra la figura.
- Si todas las réplicas de A cayeran en cierta ejecución, una réplica de B sería elegida como primario.
- Si todas las réplicas de B cayeran en cierta ejecución, una réplica de A sería elegida como primario.
- Si las réplicas de A cometen fallos de omisión general con las réplicas de B (bien no emiten o no reciben sus mensajes), entonces esta ejecución será indistinguible de la primera para las réplicas de B y de la segunda para las de A, con ellos se elegirían dos primarios, uno en cada grupo.

Índice

2.2. Replicación activa [Sch93]

- El cliente debe hacer llegar la petición a todas las réplicas.
  - En algunos sistemas, la petición llega a una sola réplica servidora, que antes de iniciar su proceso la envía a todas las demás (incluida ella) en orden total.
- Todas las réplicas sirven la petición: son activas.
- Todas las réplicas devuelven una respuesta.
  - Debe efectuarse un filtrado de respuestas en el stub cliente.
- Consistencia: Se necesita un protocolo de difusión fiable de orden total.

2.2. Replicación activa

- Ventajas:
  - Todas las réplicas comparten un mismo programa y un mismo estado.
  - La caída de una o más réplicas no requiere un cambio de rol en las réplicas restantes.
  - La caída de una o más réplicas no retarda la entrega de una respuesta al cliente.
  - Bajo este modelo, los programas servidores pueden escribirse de forma muy similar a la de un servicio no replicado.

2.2. Replicación activa

- Ventajas (continuación):
  - Pueden soportarse los fallos arbitrarios.
    - Si pueden darse “f” fallos simultáneos, se necesitarán al menos “2f+1” réplicas.
    - El cliente necesitará esperar una mayoría de respuestas coherentes.
      - ¿Cómo puede preverse el número de fallos simultáneos?

2.2. Replicación activa

- Inconvenientes:
  - Hay que garantizar que las peticiones lleguen a todas las réplicas: protocolos de difusión fiable.
  - Hay que garantizar que diferentes peticiones con interacción entre sí lleguen en el mismo orden a todas las réplicas: la difusión requiere orden total.
    - Como la difusión atómica se reduce al problema de consenso, éste debe poder resolverse en el modelo de sistema asumido.
    - Paxos [Lam98] es un ejemplo de protocolo para dar soporte al modelo activo, considerando todos los mecanismos necesarios (consenso, comunicación, orden, procesos, persistencia...).

2.2. Replicación activa

- Inconvenientes (continuación):
  - Resulta difícil atender operaciones no deterministas, pues en cada réplica podrían originar resultados diferentes.
    - Esto provoca inconsistencias.
    - Por tanto, el código de este tipo de servicios debe ser determinista.
  - Resulta difícil la gestión de las peticiones iniciadas por un objeto replicado bajo este modelo (múltiples peticiones que deberían ser identificadas como una sola por el servicio invocado).
  - Necesidad de filtrado de peticiones y filtrado de respuestas.

Índice

1. Concepto y objetivos
2. Modelos de replicación
    1. Pasivo
    2. Activo
    3. **Semipasivo**
    4. Semiactivo
3. Protocolos de replicación de BD

2.3. Replicación semipasiva
[DSS98]

- Modelo de replicación similar al pasivo, que trata de evitar alguno de sus problemas:
  - Se elimina la dependencia sobre la viveza de la réplica primaria.
  - Reconfiguración rápida en caso de fallo de dicha réplica.

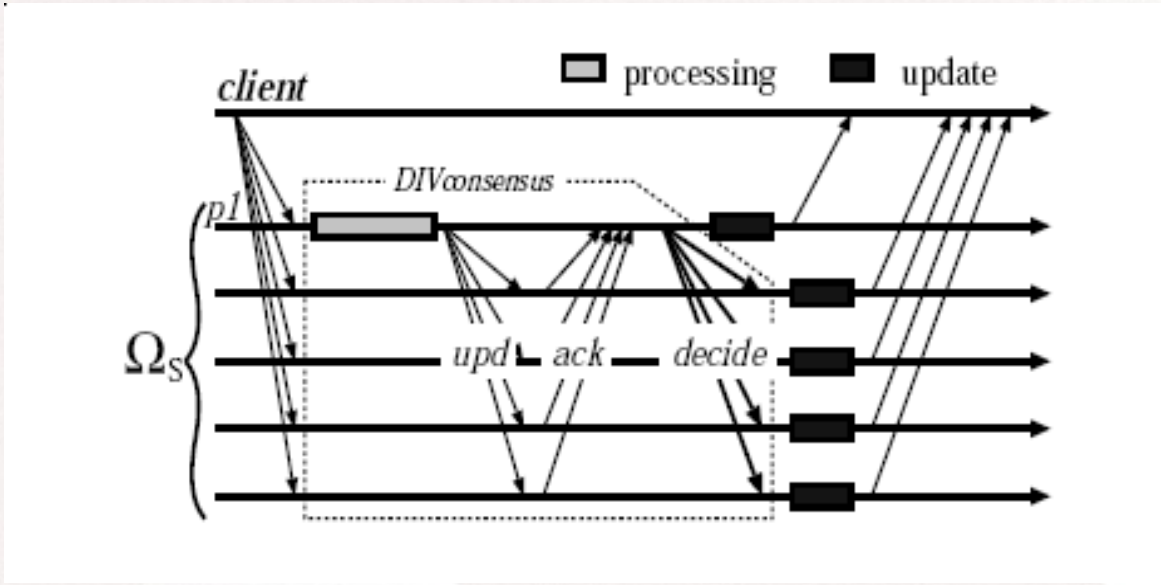
2.3. Replicación semipasiva

- Se introducen algunas variaciones:
  - Las peticiones de los clientes llegan a todas las réplicas.
    - Se necesitan difusiones de orden total.
  - Se introduce una fase de consenso, que en la práctica se emplea para:
    - Decidir el orden de entrega de las peticiones.
    - Propagar las actualizaciones.
    - Confirmar su aplicación.

2.3. Replicación semipasiva

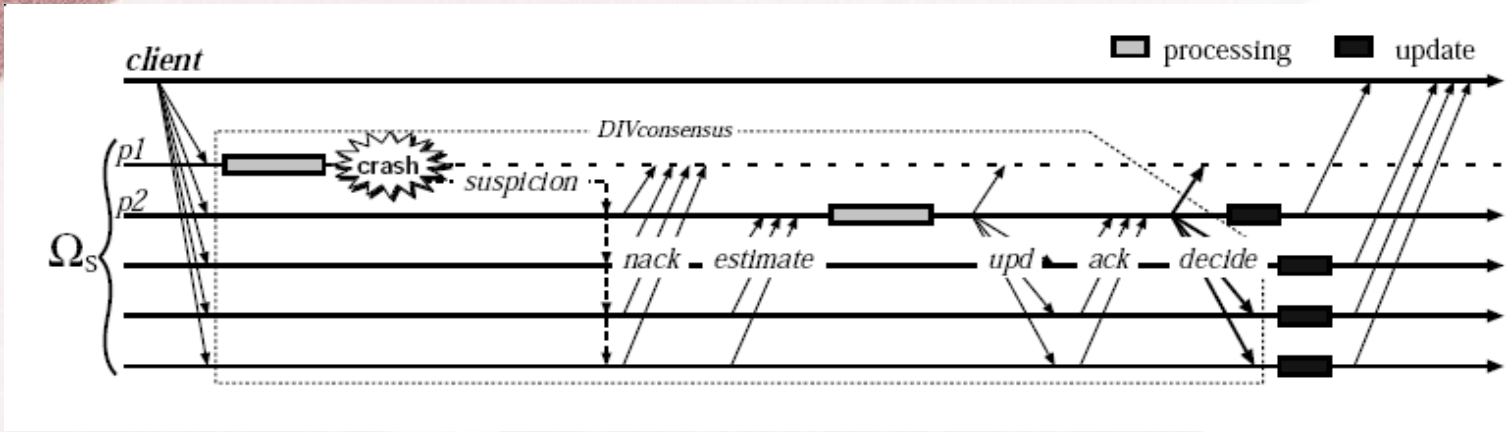
- Se introducen algunas variaciones (ii):
  - Cuando se sospecha la caída del primario, todos envían lo que conocen acerca del procesamiento de la última petición al nuevo primario.
    - Si nadie recibió el resultado de su ejecución, el nuevo primario la reejecuta.
    - Si alguno obtuvo el resultado, el nuevo primario lo difunde a todos los demás empleando el algoritmo “normal”.

2.3. Replicación semipasiva



- Ejecución sin fallos:
  - Las fases "*upd*", "*ack*" y "*decide*" son las necesarias para el protocolo de consenso utilizado para propagar las actualizaciones.

2.3. Replicación semipasiva



- Ejecución en caso de caída del primario:
  - Cuando los detectores de fallos locales así lo indiquen, se envía un mensaje "*estimate*" al nuevo primario, con el resultado de la última actualización conocida.
  - En base a tal mensaje, el nuevo primario continúa con la última petición interrumpida por el fallo.

2.3. Replicación semipasiva

- Ventajas (i):
  - No se necesita un acuerdo sobre la caída del primario.
    - En caso de sospecha incorrecta, lo que haga el “nuevo” primario será descartado.
      - El mensaje “*nack*” de la figura anterior trata de gestionar adecuadamente esa situación.
      - El primario “sospechoso” contestaría a ese “*nack*” indicando que no ha fallado y obligando a que el “nuevo” primario desista.
    - La sobrecarga introducida en ese caso no es excesiva.

2.3. Replicación semipasiva

- Ventajas (ii):
  - El cliente no tiene que reenviar ninguna petición en caso de fallo del primario.
    - Las peticiones se difunden a todos los servidores y todos ellos pueden mantenerlas.
    - Quien pase a ser el nuevo primario ya conoce el orden de entrega de las peticiones y qué petición estaba en curso.
  - El protocolo seguido para reconfigurar el servicio replicado ya incluye la finalización de la operación que estuvo en curso en el momento de la caída.

2.3. Replicación semipasiva

- Inconvenientes:
  - Necesita una difusión fiable en orden total para propagar las peticiones.
    - Pero el consenso necesario para establecer el orden total también se amortiza en otras tareas (propagación de actualizaciones y comprobación del estado del primario).
  - Necesita tres rondas de mensajes para poder aplicar las actualizaciones de estado.
    - Esto hace que el número total de rondas sea igual a cinco (incluyendo el mensaje de petición y el primer mensaje de respuesta).

Índice

2.4. Replicación semiactiva [Pow94]

- Es un modelo de replicación similar al activo cuando todas las operaciones sean deterministas.
- Sin embargo:
  - Selecciona una réplica líder que será la única que gestionará las operaciones no deterministas.
  - El resto de réplicas, aunque reciban todas las peticiones, deben esperar los resultados de la líder cuando una operación invocada no sea determinista.
  - Por tanto, ante operaciones no deterministas se adopta un modelo similar al pasivo.

2.4. Replicación semiactiva

- Las peticiones se envían a todas las réplicas.
  - Se siguen necesitando difusiones fiables. El orden total puede establecerlo el líder.
- Sólo una de las réplicas (la líder) procesa absolutamente todas las peticiones y genera tanto las respuestas como las peticiones sobre otros servidores.
- El resto de réplicas puede procesar directamente aquellas peticiones que sean deterministas, pero...
  - No pueden generar respuestas.
  - No pueden iniciar peticiones.
  - Debe esperar el resultado de la ejecución, por parte del líder, de las operaciones no deterministas.

2.4. Replicación semiactiva

- Ventajas:
  - Compartidas con el modelo activo:
    - Recuperación rápida. No hay riesgo de pérdida de peticiones.
      - Aunque caiga el líder es fácil elegir cualquier otro.
  - Compartidas con el modelo pasivo:
    - Se pueden ejecutar operaciones no deterministas.
    - No se requiere filtrar las peticiones ni las respuestas, pues las réplicas seguidoras (las “no líder”) no generan invocaciones.

2.4. Replicación semiactiva

- Inconvenientes:
  - Resulta necesario un protocolo de difusión fiable para entregar las peticiones.
    - Pero es mucho más “barato” que un protocolo de difusión atómica (fiable + orden total).
  - Hay que etiquetar de alguna manera las operaciones no deterministas, para que tanto el líder como sus seguidores las ejecuten de manera adecuada.

Índice

3.Protocolos de replicación de BD

- En [WS05] se distinguen estos tipos de protocolo de replicación:
  - Replicación activa.
  - Certificación.
  - Votación débil.
  - Replicación pasiva.
- El primero y el cuarto coinciden con los modelos de replicación “clásicos”.

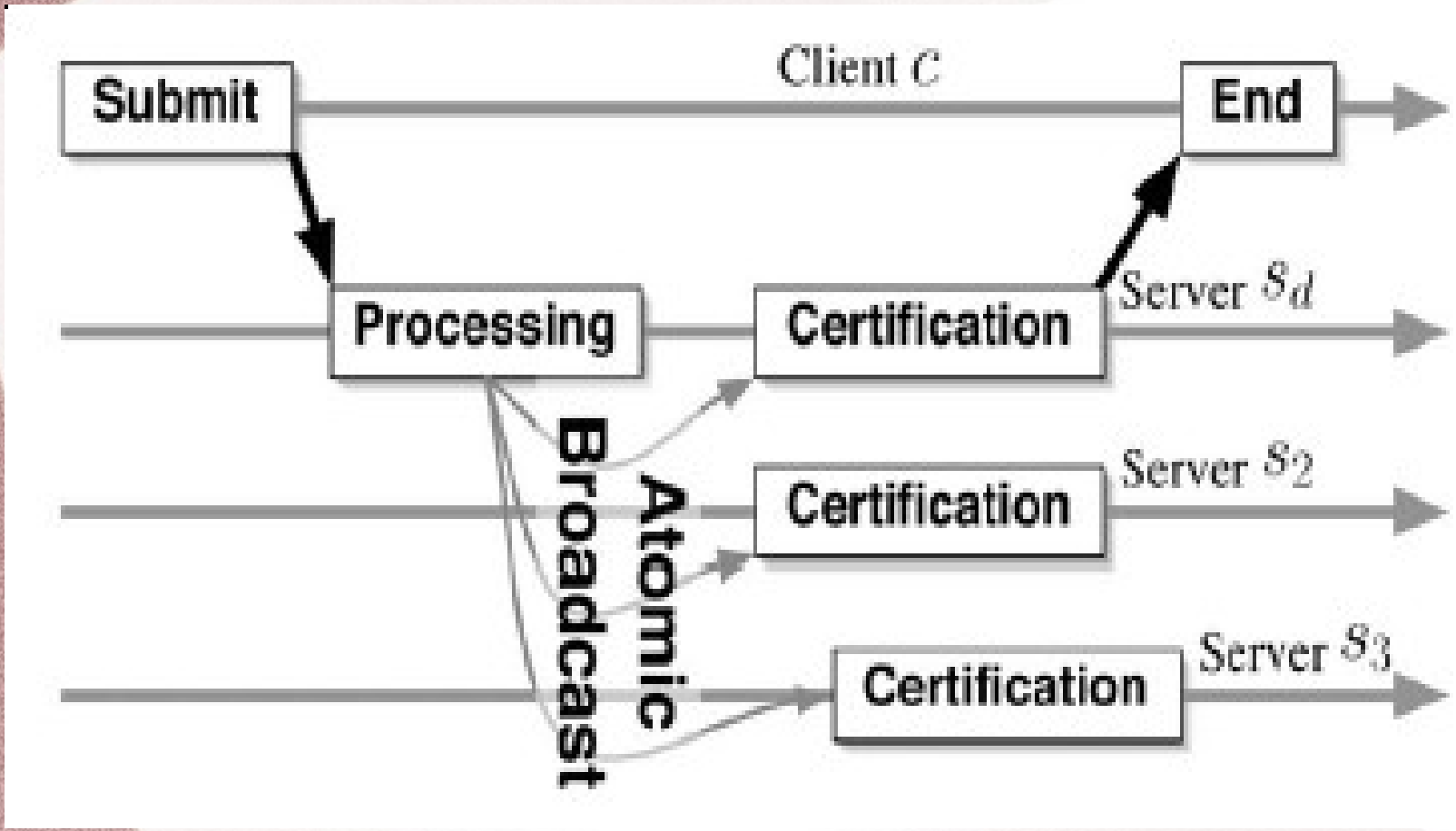
3. Protocolos de replicación de BD

- Los dos protocolos “nuevos” asumen:
  - Que la transacción se ejecuta inicialmente en un nodo “delegado”.
  - Antes de servir su petición de “commit”:
    - Se recoge el conjunto de elementos modificados: writeset.
    - Se recoge el conjunto de elementos consultados: readset.
  - Se difunde el “writeset” en orden total.
  - Se comprueban conflictos con transacciones concurrentes en un paso de “certificación”.
    - La transacción sólo es aceptada si no hay transacciones conflictivas concurrentes que la precedan dentro del orden total.

3.1. Certificación

- El cliente utiliza una réplica delegada para ejecutar directamente su transacción.
  - Cada transacción puede emplear una réplica delegada distinta.
- Antes de terminar la transacción la réplica delegada difunde el writeset en orden total.
- Cada réplica “certifica” la transacción localmente.
  - Paso simétrico en todos los nodos.
  - Comparten un mismo histórico de transacciones previas.
- Al terminar, la réplica delegada contesta al cliente.

3.1. Certificación



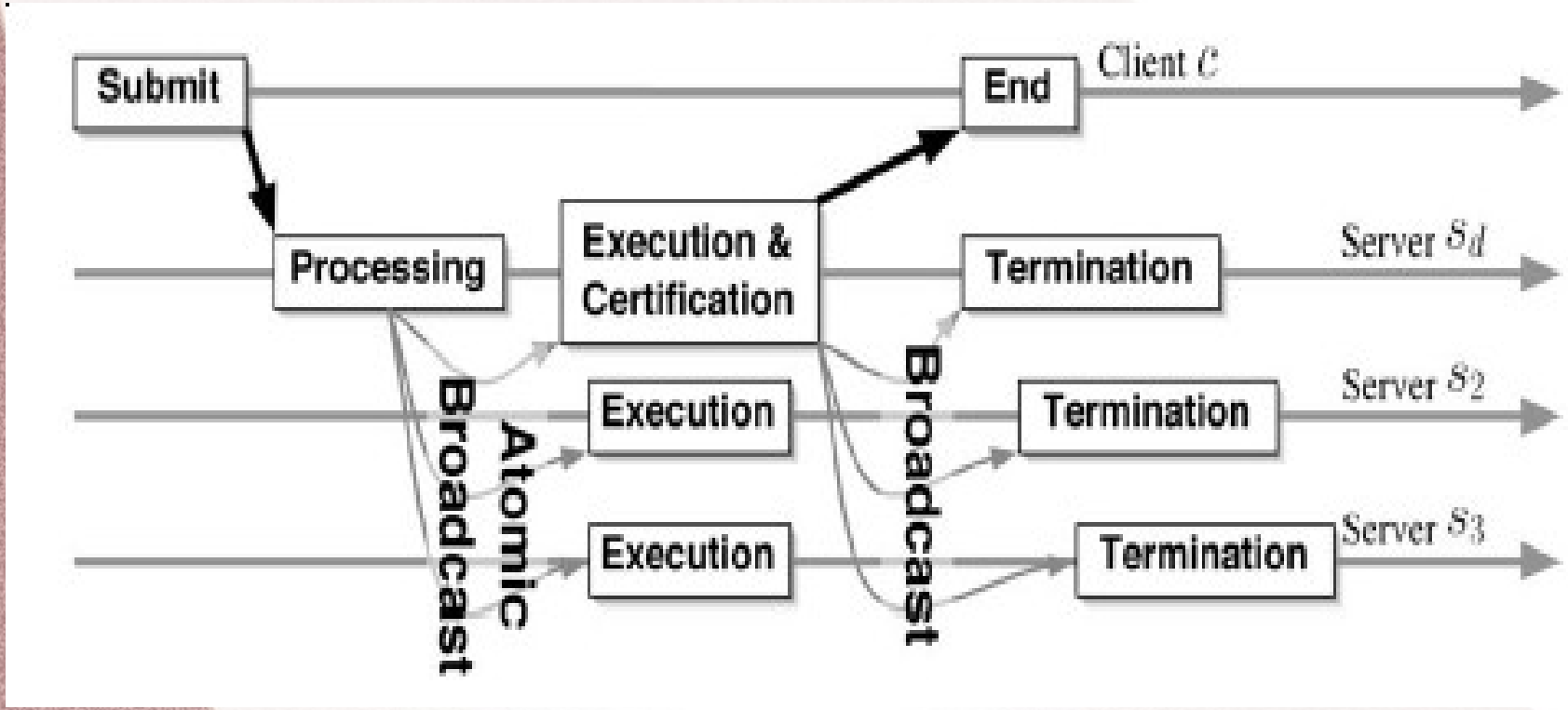
3.1. Certificación

- Protocolo únicamente válido para el nivel de aislamiento “snapshot”.
  - Los “readsets” no son relevantes para la evaluación de conflictos.
  - Se asume un control de concurrencia multiversionado en el SGBD subyacente.
    - Las operaciones de lectura no necesitan utilizar “locks” implícitos.
    - Las lecturas jamás bloquearán la transacción.

3.2. Votación débil

- Esquema similar a la certificación: proceso en una réplica delegada y difusión de su “writeset” en orden total.
- Pero aquí no “certifican” todas las réplicas, sino sólo la delegada.
- La decisión de la delegada se difunde a todas las demás (difusión fiable, sin orden).
- Aplicable al nivel de aislamiento “serializable”.
  - Necesita comprobar conflictos readset-writeset.
  - El readset lo mantiene la réplica delegada.
    - Lo compara con los writesets de las transacciones concurrentes, para ver si intersectan.
    - En ese caso, aborta la transacción local.

3.2. Votación débil



3.3. Rendimiento

- En [WS05] se analiza el rendimiento de las cuatro clases de protocolos mencionadas en esta sección 3.
  - Además, se comparan con:
    - Replicación perezosa (mejores prestaciones posibles, aunque sin garantías de consistencia)
    - Uso de “locks” distribuidos (control de concurrencia pesimista).

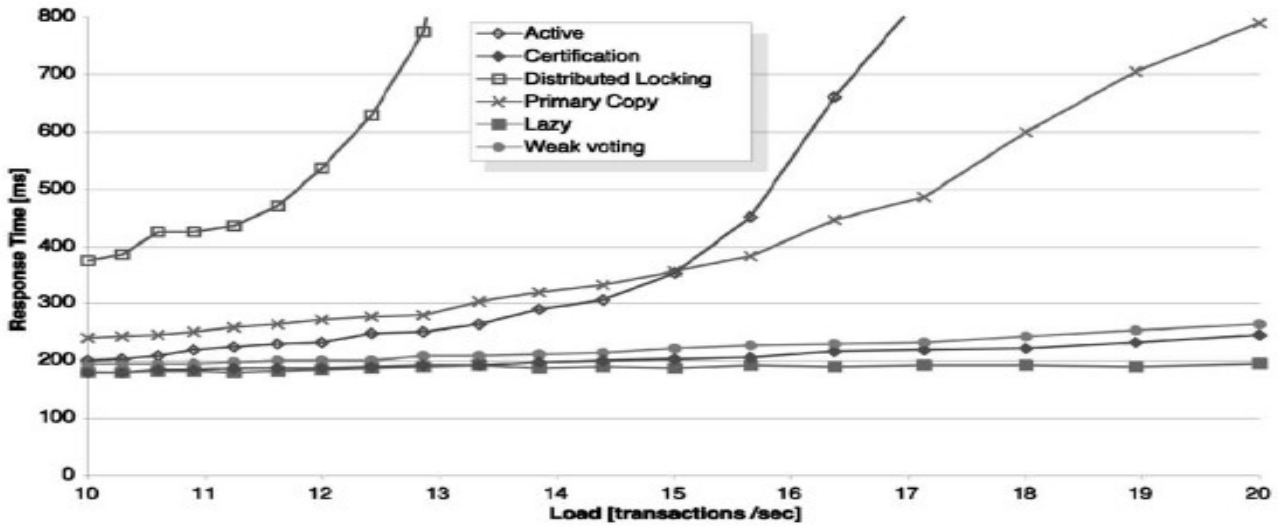
3.3. Rendimiento

TABLE 1
Simulator Parameters

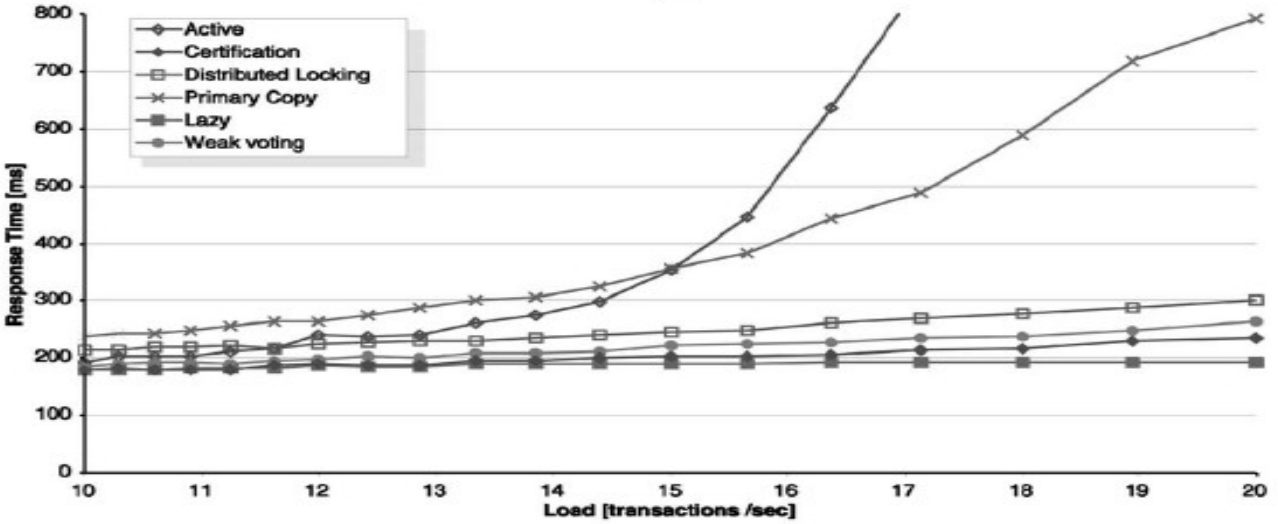
Parameter	Value
Items in the database	10'000
Number of Servers	9
Number of Clients per Server	2 or 4
Disks per Server	2
CPUs per Server	2
Transaction length	10 – 20 Operations
Operation is a write	50%
Operation is a query	50%

Parameter	Value
Buffer hit ratio	20%
Time for a read	4 – 12 *ms*
Time for a write	4 – 12 *ms*
CPU Time used for an I/O operation	0.4 *ms*
Time for a message on the Network	0.5 or 0.07 *ms*
CPU time to send/receive a message	0.5 or 0.07 *ms*
Time for a broadcast on the Network	0.5 or 0.07 *ms*
CPU time to send/receive a broadcast	0.5 or 0.07 *ms*

3.3. Rendimiento



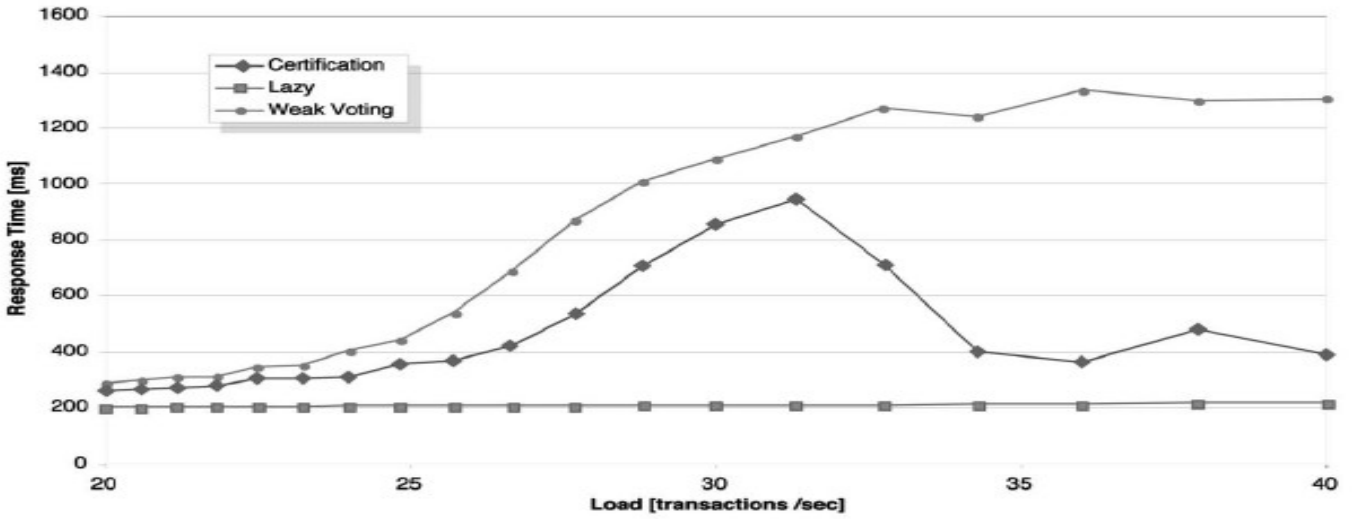
(a)



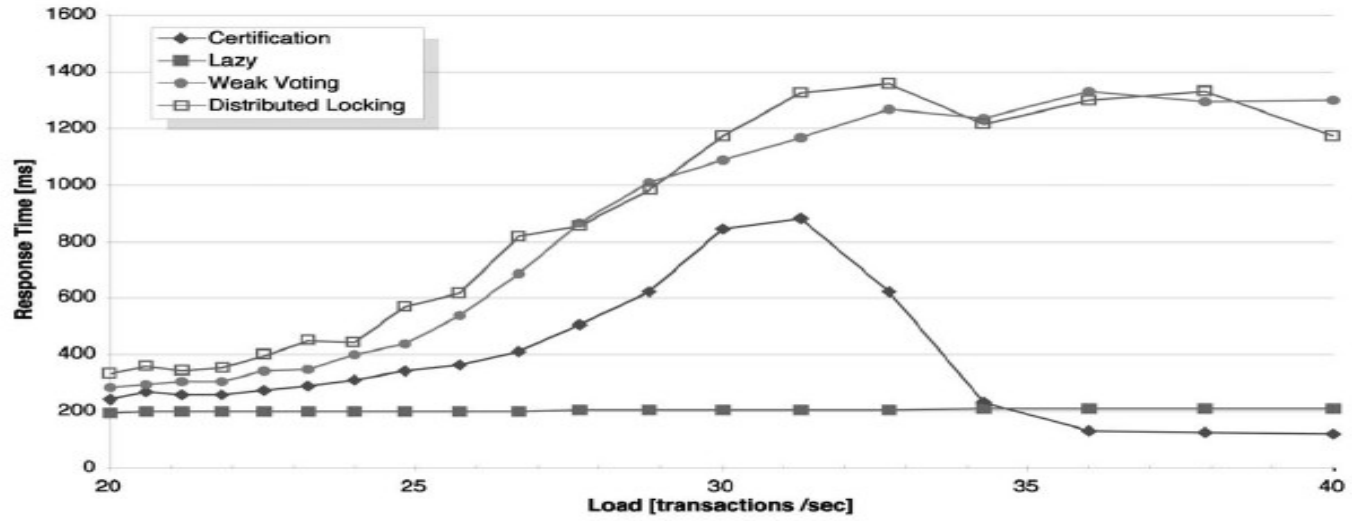
(b)

performance medium-load (a) slow network and (b) fast network.

3.3. Rendimiento



(a)



(b)

nce high-load (a) slow network and (b) fast network.

3.3. Rendimiento

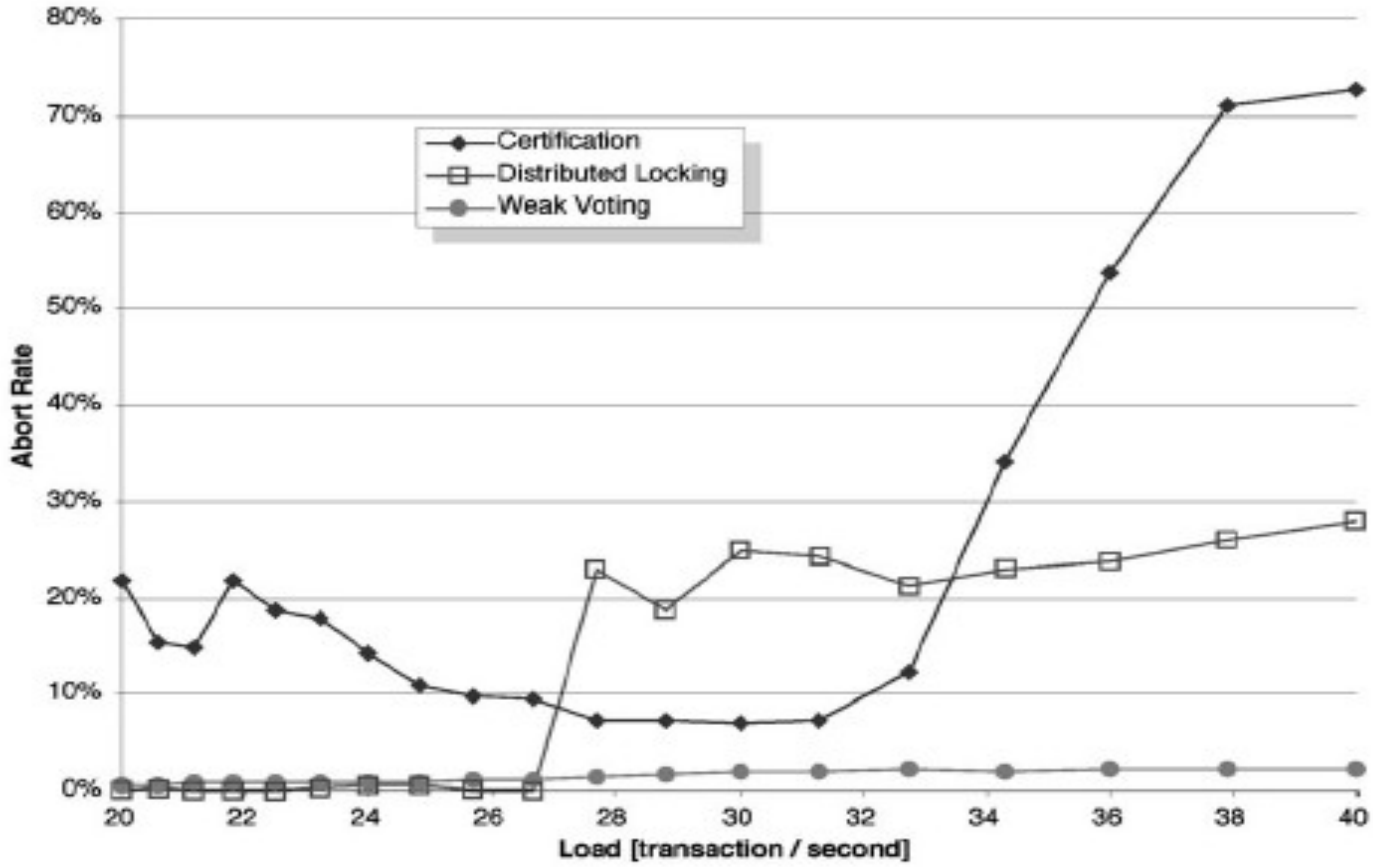


Fig. 14. Abort rate with high-load, fast network.