Pertemuan 9
CONSISTENCY AND REPLICATION

Pokok Bahasan

- Konsep consistency dan replikation
- Alasan replikasi
- Replikasi sebagai alat penskalaan
- Model konsistency data pusat
- Konsistency berkelanjutan
- Pemesanan oprasi yang konsisten
- Konsistency Kausal
- Kejadian konsistensi

Konsep Consistency dan Replycation

Masalah penting dalam sistem terdistribusi adalah replikasi data. Data umumnya direplikasi untuk meningkatkan keandalan atau meningkatkan kinerja. Salah satu masalah utama adalah menjaga agar replika tetap konsisten. Secara informal, ini berarti bahwa ketika satu salinan diperbarui, maka perlu memastikan bahwa salinan lainnya diperbarui juga; jika tidak, replikasi tidak akan sama lagi. Dalam pembahasan ini, dapat dilihat secara terperinci apa konsistensi dari data yang direplikasi. Sebenarnya makna dan berbagai cara agar konsistensi dapat dicapai. Dengan mulai dengan pengantar umum yang membahas mengapa replikasi berguna dan bagaimana hubungannya dengan skalabilitas.

Kelas penting dari apa yang dikenal sebagai model konsistensi mengasumsikan bahwa beberapa proses secara bersamaan mengakses data bersama. Konsistensi untuk situasi ini dapat dirumuskan sehubungan dengan proses apa yang dapat diharapkan saat membaca dan memperbarui data bersama, mengetahui bahwa orang lain juga mengakses data itu. Model konsistensi untuk data bersama seringkali sulit diterapkan secara efisien dalam sistem terdistribusi skala besar. Selain itu, dalam banyak kasus model yang lebih sederhana dapat digunakan, yang juga sering lebih mudah diimplementasikan. Satu kelas spesifik dibentuk oleh model konsistensi klien-sentris, yang berkonsentrasi, pada konsistensi dari perspektif klien tunggal (mungkin seluler). Model konsistensi klien-sentris dibahas dalam bagian terpisah.

Alasan Replikasi

Ada dua alasan utama untuk mereplikasi data: keandalan dan kinerja. Pertama, data direplikasi untuk meningkatkan keandalan suatu sistem. Jika sistem file telah direplikasi, dimungkinkan untuk terus bekerja setelah satu replika macet dengan hanya beralih ke salah satu replika lainnya. Juga, dengan mempertahankan banyak salinan, dimungkinkan untuk memberikan perlindungan yang lebih baik terhadap data yang rusak. Misalnya, bayangkan ada tiga salinan file dan setiap operasi baca dan tulis dilakukan pada setiap salinan. Hal tersebut dapat melindungi diri sendiri dari satu operasi penulisan yang gagal, dengan mempertimbangkan nilai yang dikembalikan oleh setidaknya dua salinan sebagai yang benar.

Alasan lain untuk mereplikasi data adalah kinerja. Replikasi kinerja penting ketika sistem terdistribusi perlu skala dalam jumlah dan wilayah geografis. Penskalaan dalam angka. Terjadi, misalnya, ketika semakin banyak proses perlu mengakses data yang dikelola oleh satu server. Dalam hal ini, kinerja dapat ditingkatkan dengan mereplikasi server dan kemudian membagi pekerjaan.

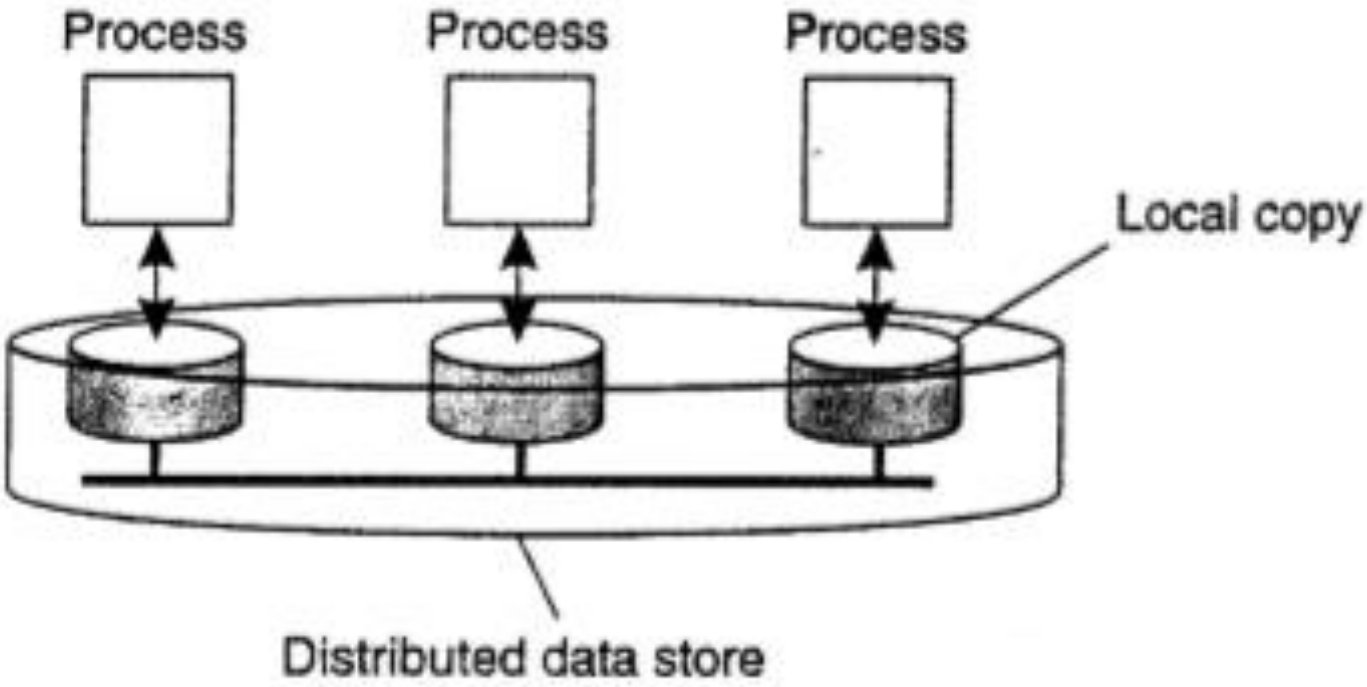
Replikasi sebagai Teknik Penskalaan

Replikasi dan caching untuk kinerja banyak diterapkan sebagai teknik penskalaan. Masalah skalabilitas umumnya muncul dalam bentuk masalah kinerja. Menempatkan salinan data yang dekat dengan proses yang menggunakannya dapat meningkatkan kinerja melalui pengurangan waktu akses dan dengan demikian memecahkan masalah stabilitas. Kemungkinan trade-off yang perlu dilakukan adalah menjaga salinan tetap up to date mungkin memerlukan lebih banyak bandwidth jaringan. Pertimbangkan proses P yang mengakses replika lokal N kali per detik, sedangkan replika itu sendiri diperbarui M kali per detik. Asumsikan bahwa pembaruan sepenuhnya menyegarkan versi replika lokal sebelumnya.

Jika N «M, yaitu, rasio akses terhadap pembaruan sangat rendah, maka memiliki situasi di mana banyak versi replika lokal yang diperbarui tidak akan pernah diakses oleh P, sehingga komunikasi jaringan untuk versi-versi itu tidak berguna. Dalam hal ini, mungkin lebih baik tidak memasang replika lokal yang dekat dengan P, atau menerapkan strategi berbeda untuk memperbarui replika.

MODEL KONSISTENSI DATA-PUSAT

Secara tradisional, konsistensi telah dibahas dalam konteks operasi baca dan tulis pada data bersama, tersedia melalui memori bersama (didistribusikan). database bersama (terdistribusi), atau sistem file (terdistribusi). Di bagian ini, digunakan penyimpanan data istilah yang lebih luas. Penyimpanan data dapat didistribusikan secara fisik di beberapa mesin. Secara khusus, setiap proses yang dapat mengakses data dari toko diasumsikan memiliki salinan lokal (atau terdekat) yang tersedia dari seluruh toko. Operasi penulisan disebarkan ke salinan lain, seperti yang ditunjukkan pada Gambar 1. Operasi data diklasifikasikan sebagai operasi tulis ketika ia mengubah data, dan jika tidak diklasifikasikan sebagai operasi tambahan.

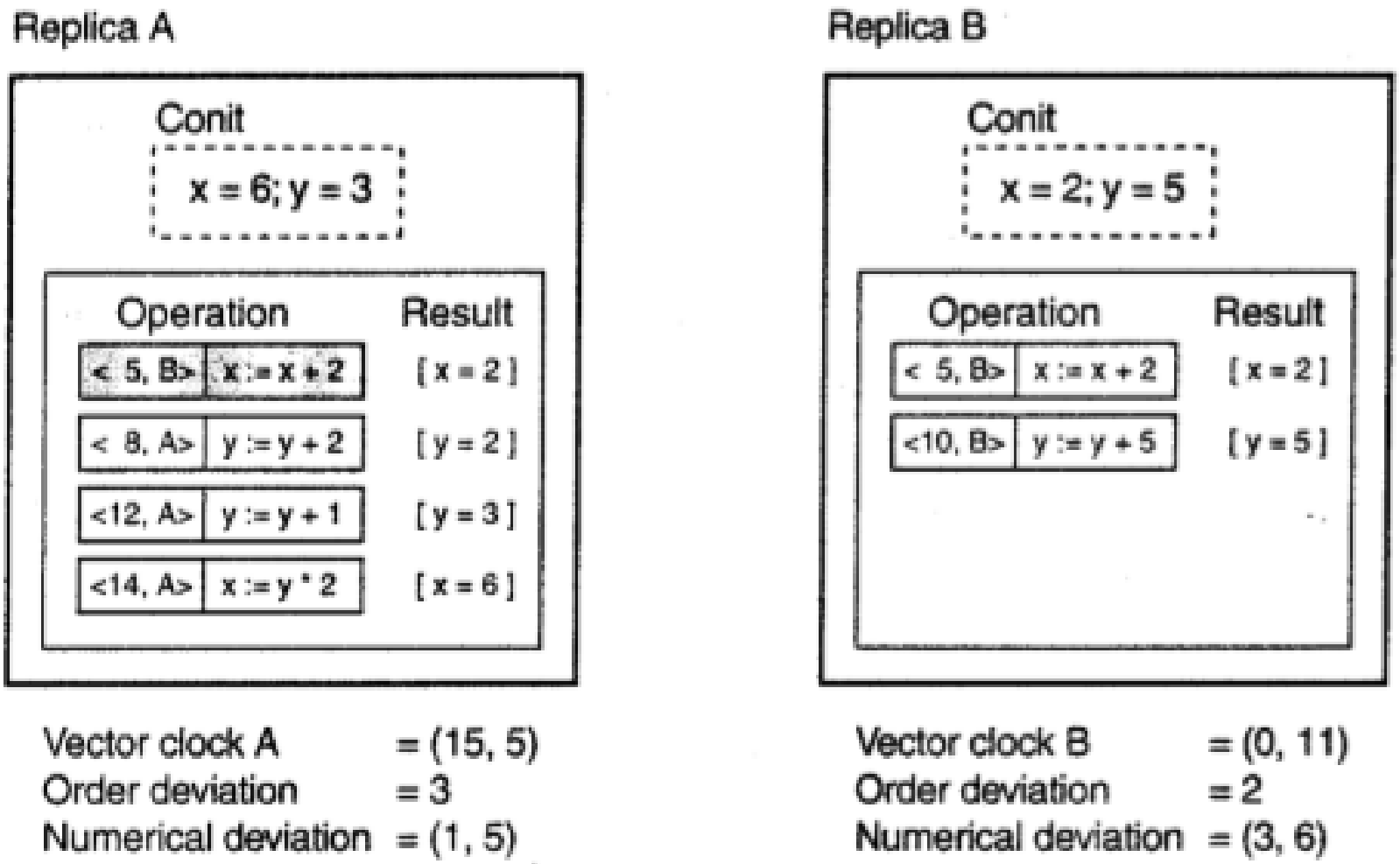


Gambar 1. Organisasi umum penyimpanan data logis, didistribusikan secara fisik dan direplikasi di berbagai proses.

Organisasi umum penyimpanan data logis, didistribusikan secara fisik dan direplikasi di berbagai proses. Model konsistensi pada dasarnya adalah kontrak antara proses dan penyimpanan data. Dikatakan bahwa jika proses setuju untuk mematuhi peraturan tertentu, toko berjanji untuk bekerja dengan benar. Biasanya, proses yang melakukan operasi dan item adata, mengharapkan operasi untuk mengembalikan nilai yang menunjukkan hasil dari operasi terakhir dari data tersebut.

Konsistensi Berkelanjutan

Secara jelas diaktakan bahwa tidak ada solusi terbaik untuk mereplikasi data. Replikasi data menimbulkan masalah konsistensi yang tidak dapat diselesaikan secara efisien secara umum. Hanya jika melonggarkan konsistensi maka ada harapan untuk mendapatkan solusi yang efisien. Sayangnya, tidak ada aturan umum untuk melonggarkan konsistensi: apa yang dapat ditoleransi sangat tergantung pada aplikasi.



Gambar 2. Contoh melacak penyimpangan konsistensi

Dalam contoh ini dapat dilihat dua replika yang beroperasi pada conit yang berisi item data x andy. Kedua variabel diasumsikan telah diinisialisasi ke O.Replica A menerima operasi 5, B: x ~ x +2 dari replika B dan menjadikannya permanen (mis., Operasi telah dilakukan pada A dan tidak dapat digulirkan kembali). Replika A memiliki tiga operasi pembaruan sementara: 8, A, 12, A, dan 14, A, yang membawa deviasi pemesanan ke 3. Juga perhatikan bahwa karena operasi terakhir 14, A, jam vektor A menjadi (15,5).

Satu-satunya operasi dari B yang A belum terlihat adalah IO, B, yang membawa deviasi numerik sehubungan dengan operasi ke 1. Dalam contoh ini, bobot deviasi ini dapat dinyatakan sebagai perbedaan maksimum antara nilai (komitmen) dari x dan y atA, dan hasil dari operasi di B tidak terlihat olehA. Nilai yang dikomit di atA adalah (x, y) = (2,0), sedangkan-untuk operasi A yang tak terlihat di B menghasilkan perbedaan ofy = 5.

Alasan yang sama menunjukkan bahwa B memiliki dua operasi pembaruan sementara: 5, B dan 10, B, yang berarti memiliki penyimpangan pemesanan 2. Karena B belum melihat satu operasi dari A, jam vektornya menjadi (0,11)). Penyimpangan numerik adalah 3 dengan bobot total 6. Nilai terakhir ini berasal dari fakta bahwa nilai komitmen B adalah (x, y) = (0,0), sedangkan operasi tentatif di A sudah akan membawa ke 6.

Perhatikan bahwa ada trade-off antara mempertahankan conit berbutir halus dan kasar. Jika suatu conit mewakili banyak data, seperti database yang lengkap, maka pembaruan dikumpulkan untuk semua data dalam conit.

Pemesanan Operasi yang Konsisten

Selain konsistensi berkelanjutan, ada banyak pekerjaan pada model konsistensi data-sentris dari dekade terakhir. Kelas model yang penting berasal dari bidang pemrograman bersamaan. Dihadapkan dengan fakta bahwa dalam komputasi paralel dan terdistribusi banyak proses perlu berbagi sumber daya dan mengakses sumber daya ini secara bersamaan, para peneliti telah berusaha untuk mengungkapkan semantik dari akses bersamaan ketika sumber daya bersama direplikasi. Ini telah menyebabkan setidaknya satu model konsistensi penting yang banyak digunakan. Berikut ini, berkonsentrasi pada apa yang dikenal sebagai konsistensi berurutan, dan juga akan membahas varian yang lebih lemah, yaitu konsistensi kausal.

Model-model yang dibahas di bagian ini semuanya berurusan dengan operasi pemesanan yang konsisten pada data yang dibagikan dan direplikasi. Pada prinsipnya, model-model tersebut menambah konsistensi yang berkesinambungan dalam arti bahwa ketika pembaruan tentatif di replika perlu dilakukan, replika perlu mencapai kesepakatan tentang pemesanan global pembaruan tersebut. Dengan kata lain, mereka perlu menyepakati urutan pembaruan yang konsisten. Model konsistensi yang diskusikan selanjutnya adalah tentang mencapai pemesanan yang konsisten.

Gambar 3. Perilaku dua proses yang beroperasi pada item data yang sama. Sumbu horizontal adalah waktu.

Sebagai contoh, pada Gambar 2. PI melakukan penulisan ke item data x, memodifikasi nilainya menjadi a. Perhatikan bahwa, pada prinsipnya, operasi ini WI (x) a pertama kali dilakukan pada salinan penyimpanan data yang bersifat lokal untuk PI, dan kemudian disebarkan ke salinan lokal lainnya. Dalam contoh, P2 kemudian membaca nilai NIL, dan beberapa saat setelah itu a (dari salinan lokal toko).

Apa yang dapat dilihat di sini adalah butuh waktu untuk memperbarui pembaruan ofx toP2, yang dapat diterima dengan sempurna. Konsistensi sekuensial adalah model konsistensi data-sentris penting, yang pertama kali didefinisikan oleh Lamport (1979) dalam konteks memori bersama untuk sistem multiprosesor. Secara umum, penyimpanan data dikatakan konsisten secara berurutan ketika memenuhi kondisi berikut:

Hasil dari setiap eksekusi adalah sama seperti jika operasi (baca dan tulis) oleh semua proses pada penyimpanan data dieksekusi dalam beberapa urutan berurutan dan operasi-setiap proses individu muncul dalam urutan ini dalam urutan yang ditentukan oleh programnya.

Konsistensi Kausal

Model konsistensi sebab akibat mewakili melemahnya konsistensi sekuensial dalam hal itu membuat perbedaan antara peristiwa yang berpotensi terkait secara kausal dan yang tidak. Dengan menemukan hubungan sebab akibat ketika mendiskusikan waktu vektor. Jika peristiwa b disebabkan atau dipengaruhi oleh peristiwa sebelumnya a, kausalitas mengharuskan semua orang melihat terlebih dahulu, kemudian melihat b.

Pertimbangkan interaksi sederhana dengan menggunakan database bersama yang didistribusikan. Misalkan proses P, menulis item data x. Kemudian P2 membaca x dan menulis y. Di sini, pembacaan x dan penulisan y berpotensi terkait secara kausal karena perhitungan y mungkin bergantung pada nilai x seperti yang dibaca oleh Pz (yaitu, nilai yang ditulis oleh PI) 'Di sisi lain, jika dua proses secara spontan dan secara bersamaan menulis dua item data yang berbeda, ini tidak terkait secara kausal. Operasi yang tidak terkait secara kausal dikatakan konkuren. Agar suatu penyimpanan data dianggap konsisten secara kausal, maka perlu bahwa storeobeys kondisi berikut:

Menulis yang berpotensi terkait secara kausal harus dilihat oleh semua proses dalam urutan yang sama. Penulisan bersamaan dapat dilihat dalam urutan berbeda pada mesin yang berbeda.

Di sini memiliki urutan kejadian yang diizinkan dengan toko yang konsisten, tetapi yang dilarang dengan toko yang konsisten secara berurutan atau toko yang sangat konsisten. Hal yang perlu diperhatikan adalah bahwa penulisan Wz (x) b dan WI (x) c adalah bersamaan, sehingga tidak diperlukan bahwa semua proses melihatnya dalam urutan yang sama.

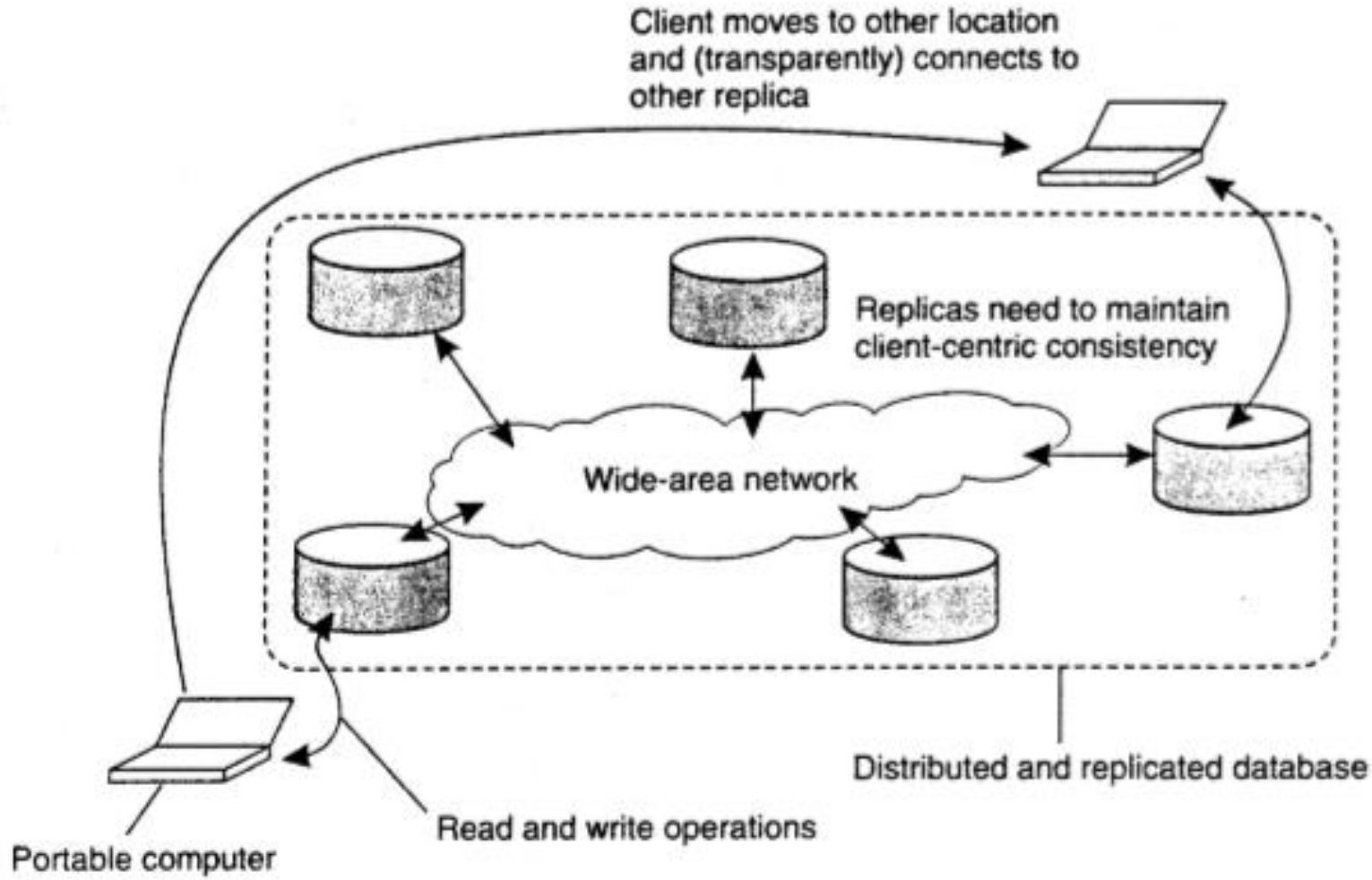
P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

Gambar 4. Urutan ini diizinkan dengan toko kausal-konsisten, tetapi tidak dengan toko konsisten berurutan.

Kejadian Konsistensi

Sejauh mana proses benar-benar beroperasi secara bersamaan, dan sejauh mana konsistensi perlu dijamin, dapat bervariasi. Ada banyak contoh di mana concurrency hanya muncul dalam bentuk terbatas. Misalnya, dalam banyak sistem basis data, sebagian besar proses hampir tidak pernah melakukan operasi pembaruan; mereka kebanyakan membaca data dari database. Hanya satu, atau sangat sedikit proses melakukan operasi pembaruan. Pertanyaannya kemudian adalah seberapa cepat pembaruan harus dibuat tersedia untuk proses hanya membaca.

Menyimpan data yang akhirnya konsisten dengan demikian memiliki properti yang tanpa adanya pembaruan, semua replika bertemu satu sama lain. Konsistensi akhirnya pada dasarnya hanya mensyaratkan bahwa pembaruan dijamin untuk menyebar ke semua replika. Konflik tulis-tulis seringkali relatif mudah dipecahkan ketika mengasumsikan bahwa hanya sekelompok kecil proses yang dapat melakukan pembaruan. Oleh karena itu konsistensi akhirnya seringkali murah untuk diterapkan. Menyimpan data konsisten akhirnya berfungsi selama klien selalu mengakses replika yang sama.



Gambar 5. Prinsip pengguna ponsel mengakses replika berbeda dari database terdistribusi.

