

Replikasi

Aloysius S Wicaksono, 32701
Glagah Seto S Katon, 32839
Jurusan Teknik Elektro FT UGM,
Yogyakarta

I. PENDAHULUAN

Replikasi adalah kunci untuk ketersediaan tinggi dan toleransi kesalahan di dalam system terdistribusi. Ketersediaan yang tinggi merupakan peningkatan ketertarikan dari pergerakan mengikuti aturan terhadap komputasi bergerak dan secara konsekuensi memutuskan operasi. Toleransi kesalahan menyangkut layanan yang disediakan dalam *safety-critical* dan system penting lainnya, sehingga apabila terjadi kerusakan tidak akan mengheatkan proses dalam system jaringan

II. PENGERTIAN REPLIKASI

A. Definisi

Replikasi adalah suatu teknik untuk melakukan copy dan pendistribusian data dan objek-objek database dari satu database ke database lain dan melaksanakan sinkronisasi antara database sehingga konsistensi data dapat terjamin.

Dengan menggunakan teknik replikasi ini, data dapat didistribusikan ke lokasi yang berbeda melalui koneksi jaringan lokal maupun internet. Replikasi juga memungkinkan untuk mendukung kinerja aplikasi, penyebaran data fisik sesuai dengan penggunaannya, seperti pemrosesan transaksi online dan DSS (Decision Support System) atau pemrosesan database terdistribusi melalui beberapa server.

Keuntungan replikasi tergantung dari jenis replikasi tetapi pada umumnya replikasi mendukung ketersediaan data setiap waktu dan dimanapun diperlukan.

Adapun keuntungan lainnya adalah :

1. Memungkinkan beberapa lokasi menyimpan data yang sama. Hal ini sangat berguna pada saat lokasi-lokasi tersebut membutuhkan data yang sama atau memerlukan server yang terpisah dalam pembuatan aplikasi laporan.
2. Aplikasi transaksi online terpisah dari aplikasi pembacaan seperti proses analisis database secara online, data smart atau data warehouse.
3. Memungkinkan otonomi yang besar. Pengguna dapat bekerja dengan meng-copy data pada saat tidak terkoneksi kemudian melakukan perubahan untuk dibuat database baru pada saat terkoneksi
4. Data dapat ditampilkan seperti layaknya melihat data tersebut dengan menggunakan aplikasi berbasis Web
5. Meningkatkan kinerja pembacaan
6. Membawa data mendekati lokasi individu atau kelompok pengguna. Hal ini akan membantu mengurangi masalah karena modifikasi data dan pemrosesan query yang dilakukan oleh banyak pengguna karena data dapat didistribusikan melalui jaringan dan data dapat dibagi berdasarkan kebutuhan masing-masing unit atau pengguna.
7. Penggunaan replikasi sebagai bagian dari strategi standby server.

8. menyembunyikan perbedaan-perbedaan antara layanan replicated dan non-replicated

Replikasi dapat digunakan apabila sebuah organisasi atau perusahaan didukung oleh hardware dan aplikasi software dalam sebuah sistem yang terdistribusi. Aplikasi yang berbeda mempunyai kebutuhan yang berbeda untuk otonomi dan konsistensi data. Replikasi diperlukan dalam sistem terdistribusi apabila berikut ini:

1. Mengcopy dan mendistribusikan data dari satu atau lebih lokasi
2. Mendistribusikan hasil copy data berdasarkan jadwal
3. Mendistribusikan perubahan data ke server lain
4. Memungkinkan beberapa pengguna di beberapa lokasi untuk melakukan perubahan dan kemudian menggabungkan data yang telah dimodifikasi
5. Membangun aplikasi data yang menggunakan perlengkapan online maupun offline
6. Membangun aplikasi Web sehingga pengguna dapat melihat volume data yang besar.

B. Jenis Replikasi

Terdapat beberapa jenis – jenis replikasi diantaranya adalah sebagai berikut :

1. Snapshot replication

Mendistribusikan data yang dapat dilihat pada saat tertentu tanpa melakukan update. Biasanya digunakan pada saat memerlukan tampilan data seperti : daftar harga, katalog, data yang digunakan untuk pengambilan keputusan. Data-data ini sifatnya hanya 'read only'.

Replikasi ini membantu pada saat :

- data sebagian besar statis dan tidak sering berubah
- dapat menerima copy data yang telah melewati batas waktu yang ditentukan
- datanya sedikit

2. Transactional replication

Memelihara kekonsistenan transaksi yang terjadi

3. Merge replication

Merge replication memungkinkan pengguna bekerja dan merubah data sesuai dengan wewenangnya. Pada saat server tidak dikoneksikan ke seluruh lokasi dalam topologi, replikasi merubah ke nilai data yang sama.

III. MODEL SISTEM

A. Definisi Model Sistem

Kita mengasumsikan asinkronus sistem yang prosesnya mungkin saja gagal karena tabrakan. Asumsi dasar kita adalah bahwa partisi network bisa tidak terjadi, tapi kadang-kadang kita harus memperhitungkan akibatnya bila itu terjadi. Partisi network membuatnya lebih keras untuk membangun detektor kegagalan, yang kita gunakan untuk mencapai reliabilitas dan multicast secara total.

Untuk kepentingan generalisasi, kita gambarkan arsitektur komponen oleh tugasnya dan tanpa maksud untuk menyatakan bahwa komponen-komponen seharusnya diimplementasikan oleh proses yang berbeda (atau hardware). Sistem model membutuhkan tiruan yang didapat oleh replica pengelola yang berbeda (lihat gambar 14.1), yang mana adalah komponen yang mengandung tiruan pada komputer yang diberikan dan menjalankan operasi pada nya secara langsung. Model umum ini bisa diaplikasikan pada lingkungan client-server, jika replika pengelola adalah server. Malahan kadang-kadang kita harus menyederkanakan pemanggilannya. Secara sama, hal itu mungkin diaplikasikan pada aplikasi dan aplikasi proses bisa menjadi client dan replika pengelola. Contohnya, laptop user di kereta api bisa memiliki aplikasi yang bertindak sebagai replika pengelola untuk catatannya.

Kita selalu membutuhkan bahwa replika pengelola menjalankan operasi pada replikanya. Ini mengijinkan user untuk mengasumsikan bahwa operasi pada replika pengelola tidak meninggalkan hasil yang inkonsisten jika melalui jalannya. Kadang2 kita memerlukan replika pengelola untuk menjadi state machine. Seperti replika pengelola menjalankan operasi kepada replikanya secara otomatis, sehingga eksekusinya ekuivalen untuk melakukan operasi dengan urutan yang tepat. Lebih lebih state dari replikanya adalah fungsi yang bisa ditentukan dari state awal dan urutan operasinya. Pembacaan clock atau sensor tidak melahirkan pada nilai statenya. Tanpa asumsi ini, konsistensi jaminan antara replika pengelola yang menerima update operasi secara independen tidak terjadi. Sistem hanya menentukan operasi mana meng-apply pada semua replika pengelola dan pada urutan apa -- itu tidak bisa menghasilkan non deterministik efek. Asumsi menyatakan bisa saja tidak mungkin, tergantung threading arsitektur untuk server menjadi multi thread.

Jarang replika pengelola memelihara setiap objek dan kita asumsikan ini kecuali kita state sebaliknya. Tapi pada replika umumnya dari objek yang berbeda bis dipelihara oleh replika pengelola yang berbeda pula. Misalnya sebuah objek diperlukan oleh banyak klien pada satu jaringan dan klien lain pada jaringan yg beda. Hal itu mudah untuk menambah dengan mereplikasinya pada replika pengelola di jaringan yg beda.

Kumpulan replika pengelola bisa statik atau dinamik. Pada dinamik sistem, replika pengelola yg baru bisa terlihat (semisal sekertaris dua mengkopi diri ke laptopnya) yang tidak ada di statis sistem. Pada dinamik sistem, replika pengelola bisa crash, dan mereka dianggap telah meninggalkan sistem (meskipun mungkin saja mereka digantikan). Pada statis sistem, replika pengelola tidak crash, tapi mereka berhenti beroperasi untu waktu yang tidak diketahui.

Model umum dari replika management terlihat pada gambar 14.1. Koleksi replika pengelola menyediakan layanan pada klien. Klien melihat layanan yang memberi mereka akses pada objek, yang mana sebenarnya direplikasi oleh pengelola. Klien meminta series operasi. Operasi involve a kombinasi pembacaan objek dan update objek. Permintaan operasi yang involve tidak mengupdate disebut read only request. Permintaan yang mengupdate objek disebut update request.

Tiap klien pertama kali dipegang oleh komponen yang disebut front end. Peran front end adalah mengkomunikasikan oleh pesan melewati dengan satu atau lebih replika pengelola, daripada memaksa klien melakukan sendiri secara eksplisit. Itu adalah cara untuk membuat replikasi transparan. Front end bisa diimplementasikan pada alamat klien atau proses yang terbagi.

Secara umum, lima fase pada performance single request pada objek tereplikasi. Aksinya menurut jenis sistem seperti menjadi clear dalam dua sesi selanjutnya. Misalnya layanan yang mendukung operasi terputus berlaku berbeda dengan fault tolerance service. Fase2nya adalah:

Front end issue request ke satu atau lebih replika pengelola. Kemungkinan pertama adalah front end untuk berkomunikasi dengan satu replika pengelola, yang berkomunikasi dengan replika pengelola lainnya. Yang kedua adalah untuk front end ke multicast request ke replika pengelola.

Koordinasi: replika pengelola mengkoordinasi dalam persiapan eksekusi permintaan secara konsisten. replika pengelola memutuskan pada ordering dari permintaan relatif pada lainnya. Semua tipe ordering mendefinisikan multicast section 11.4.3 juga apply request handling dan kita definisikan order again untuk konteks ini:

- FIFO ordering: Jika front end menghasilkan r kemudian r', sehingga replika pengelola menghandle r' sebelum r.
- Casual ordering : jika request r terjadi sebelum request r', kemudian replika pengelola yg tepat bisa menghandle r' sebelumnya.
- Total ordering: Jika replika pengelola yg tepat menghandle r sebelum r', sehingga replika pengelola yg tepat bisa menghandle r' sebelumnya.
- Banyak aplikasi memerlukan fifo ordering. kita bahas kebutuhan casual dan total ordering.
- Eksekusi: replika pengelola mengeksekusi permintaan. Yang bisa dikembalikan efeknya nanti.
- Persetujuan : replika pengelola mencapai konsensus pada efek dari permintaan yang akan dilakukan. Misal dalam transaksional sistem replika pengelola bisa secara kolektif setuju untuk menghentikan transaksi pada this stage.
- Response: satu atau lebih replika pengelola bertanggung jawab pada front end. Pada beberapa sistem, satu replika pengelola mengirim jawaban. Lainnya front end menerima jawaban dari replika pengelola dan memilih untuk dikembalikan ke klien.

Sistem berbeda bisa membuat pilihan berbeda tentang penguatan fase.

IV. TOLERANSI TERHADAP KESALAHAN

Fault tolerance atau yang dimaksud toleransi terhadap kesalahan yaitu adanya toleransi terhadap kesalahan perangkat keras computer dan memberikan toleransi terhadap terjadinya kondisi buruk pada system jaringan dengan memberikan perlindungan data , sehingga apabila terjadi kerusakan tidak akan menghentikan proses dalam system jaringan .Kerusakan seberapa parahpun akan masih tetap di lindungi sehingga tidak akan membuat data menjadi hilang atau rusak dengan kata lain system fault tolerance akan menjaga sebuah data di hard disk agar data yang ada tetap merupakan data yang asli , tidak cacat , dan tidak berubah.

Hampir semua system operasi jaringan menyediakan fasilitas fault tolerance untuk menjaga keamanan dan ketersediaan data dalam bentuk media penyimpanan (hard disk) . Ada dua system utama yang hampir selalu tersedia pada system operasi jaringan yaitu :

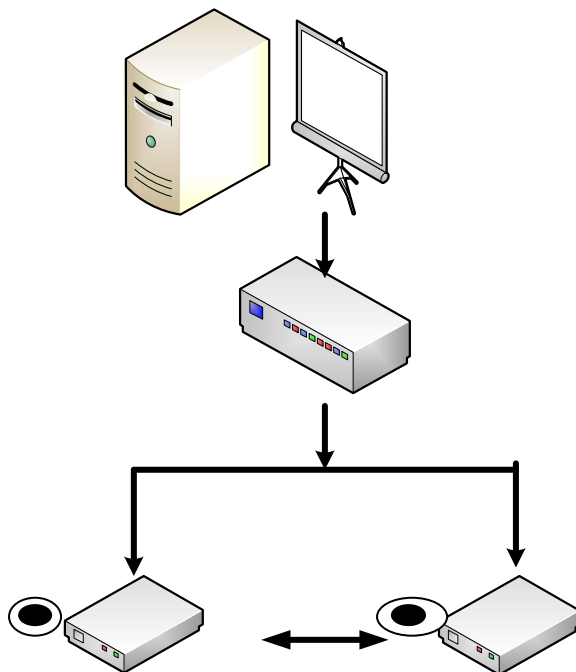
Disk Mirroring

Disk Duplexing

A. *Disk Mirroring*

Fasilitas fault tolerance dengan menggunakan disk mirroring adalah system pengamanan data dengan menggunakan dua buah partisi pada dua buah hard disk untuk menyimpan data yang sama. Satu hard disk yang berfungsi sebagai media penyimpanan utama sedangkan hard disk yang lain berfungsi sebagai hard disk bayangan atau sebagai mirror disk yang memiliki kapasitas yang sama atau lebih besar dari disk utama . Dengan menggunakan system disk mirroring , saat proses penulisan data akan dilakukan penulisan ke dua hard disk tersebut tapi system operasi akan menganggapnya sebagai satu hard disk .

Sistem disk mirroring memiliki kelebihan yaitu apabila terjadi kerusakan pada data pada salah satu hard disk , maka hard disk yang lain dapat mengambil alih dan data akan tetap aman sehingga proses kerja tetap berlansung seolah tidak ada kerusakan yang terjadi . Begitu juga jika salah satu hard disk tersebut berhenti akibat kerusakan fisik , maka data yang ada pada hard disk tersebut masih bisa di akses atau dibuka pada hard disk yang lain . Pada umumnya proses membaca dan menulis hard disk drive dengan menggunakan system disk mirroring ini sangat berbeda dengan proses membaca dan menulis pada satu hard disk drive . Proses yang terjadi pada system disk mirroring ini adalah dilakukannya secara serentak proses penulisan ke dalam dua hard disk drive tersebut sedangkan pada proses pembacaan pada system disk mirroring akan dilakukan secara berurutan proses pembacaan kedalam hard disk drive tersebut , hal ini terjadi karena proses pembacaan akan memerlukan peningkatan kecepatan pembacaan data dari disk . :



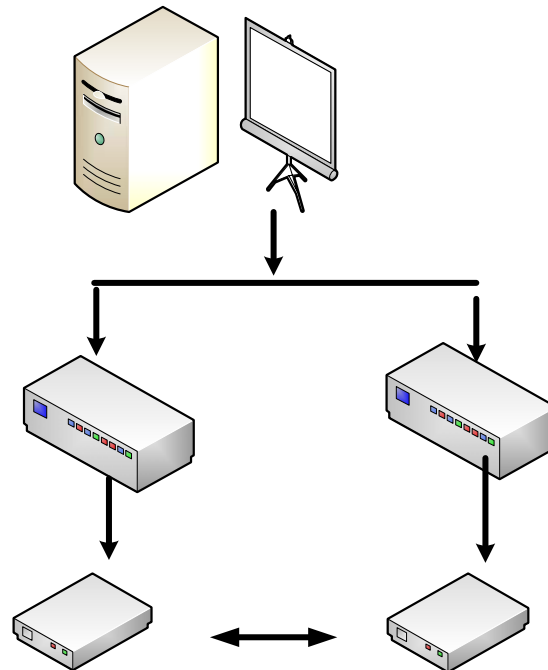
Gambar 1. disk mirroring pada komputer server

Gambar diatas menunjukkan bagaimana bentuk disk mirroring pada computer server bekerja yang mana system operasi jaringan seperti windows dan novell merupakan system operasi jaringan yang memiliki fasilitas fault tolerance

B. Disk Duplexing

Pada dasarnya system disk duplexing sama dengan system disk mirroring ,yaitu fasilitas fault tolerance dengan menerapkan system penulisan data pada hard disk drive , di mana satu hard disk bertindak sebagai disk utama dan hard disk yang lain bertindak sebagai mirroring disk . Perbedaannya terletak pada pengontrolannya yang mana system disk mirroring , kedua hard disk yang digunakan dikontrol oleh satu

controller , Sedangkan pada sistem disk duplexing masing – masing hard disk di control oleh controller sendiri .



Gambar 2. contoh system duplexing

Pada gambar di atas menunjukkan bentuk system disk duplexing yang mana pada dasarnya sistem disk duplexing memiliki kelebihan yang sama dengan sistem disk mirroring . Bahkan proses sistem penulisan dan proses pembacaan data dari disk menggunakan logika yang sama sehingga pada operasi windows server kedua sistem ini dianggap sama . Keuntungan sistem disk duplexing dari pada menggunakan disk mirroring adalah digunakannya controller disk yang terpisah untuk kedua hard disk tersebut dengan kata lain kemungkinan terhentinya sistem akibat kerusakan controller disk menjadi sangat kecil , sedangkan pada sistem disk mirroring , jika disk controller mengalami kerusakan maka kedua hard disk yang ada menjadi tidak berfungsi .

Disk Con

C. Layanan Toleransi pada Kesalahan

Mempertimbangkan sebuah sistem replikasi yang polos, dimana ketika terdapat dua replika pengelola pada computer A dan B masing-masing memelihara replika dari dua akun bank x dan y. Klien membaca dan memperbaharui akun pada replica pengelola lokal mereka tapi mencoba replika pengelola lainnya ketika ada satu yang gagal. Pengelola replica memperbanyak perbaharuan ke satu dan lainnya pada latar belakang setelah merespon kepada klien. Kedua akun memiliki saldo \$0.

Klien 1 memperbaharui saldo dari x pada pengatur replica lokal B untuk menjadi \$1 dan kemudian menerima perbaharuan dari saldo B sebesar \$2 tapi begitu mengetahui B gagal. Klien 1 kemudian memperbaharui A seketika. Sekarang klien 2 dapat membaca saldo pada pengelola replica lokal A dan menemukan kalau y memiliki saldo \$2 dan x \$0. perbaharuan dari x akun bank dari B tidak sampai, dikarenakan B gagal. Situasi dibawah menunjukkan operasi contoh dari kasus diatas

<p>Client 1:</p> <p><i>setBalance_B(x, 1)</i></p> <p><i>setBalance_A(y, 2)</i></p>	<p>Client 2:</p> <p><i>getBalance_A(y) → 2</i></p> <p><i>getBalance_A(x) → 0</i></p>
---	---

Gambar 3. contoh kasus

Ekskusi ini tidak menunjukkan kesamaan dari spesifikasi kebiasaan akun bank: klien 2 seharusnya telah membaca saldo \$1 untuk x, yang juga membaca \$2 pada y. Hal inilah yang kita tidak inginkan oleh sebab itu inilah termasuk kegunaan dari layanan toleransi terhadap kesalahan.

Lalu seharusnya bagaimana agar toleransi dari kesalahan dapat kita terapkan pada kasus diatas? Terdapat beberapa kriteria pembenahan pada objek replikasi salah satunya adalah *linearizability* dimana pada kasus diatas ketika komputer A dan B gagal tapi jika perbaharuan x pada klien1 yang terjadi pada B tidak sampai ke A ketika klie 2 membacanya. Maka dengan menggunakan *interleaving* kendala tersebut dapat diatasi dimana nantinya keduanya mampu menerima pembaharuan x, dan y dari klien 1 dan 2.

REFERENCES

- [1] Jean Dollimore, Tim Kindberg, and George Coulouris, "Disributed System: Concept and Design, 4th ed, May 2005.
- [2] http://elearning.gunadarma.ac.id/docmodul/AS400/AS400_B1/04Replikasirev.pdf
- [3] <http://kambing.ui.ac.id/bebas/v02/org/vlsm/fusikom-ui/fusikom-93-s193abs.html>
- [4] <http://74.125.153.132/search?q=cache:wcoO0v1ZryUJ:sitaliyah.staff.gunadarma.ac.id/Downloads/files/11435/Week1%2BPengantar%2BSisTer.pdf+replikasi+%2B+budsus&cd=2&hl=en&ct=clnk&client=firefox-a>