Pertemuan 6
SYNCHRONIZATION

Pokok Bahasan

- Konsep synchronization
- Physical clock
- Global positioning system
- Clock Synchronization Algorithms
- Logical clocks
- Jam Logical Lamport
- Vector Clocks
- Mutual Exclusion
- Centralized Algorithm
- Decentralized Algorithm
- Token Ring Algorithm

Konsep Synchronization

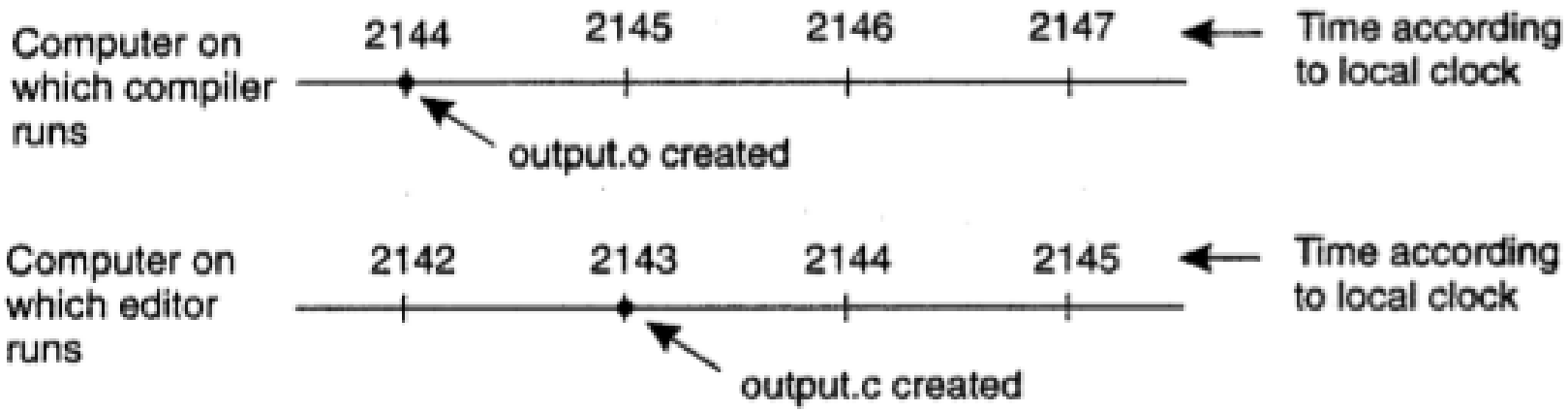
Dalam pembahasan sebelumnya, telah melihat proses dan komunikasi antara proses. Meskipun komunikasi itu penting, itu bukan keseluruhan cerita. Yang terkait erat adalah bagaimana proses bekerja sama dan menyinkronkan satu sama lain. Kerja sama sebagian didukung oleh cara penamaan, yang memungkinkan proses setidaknya berbagi sumber daya, atau entitas pada umumnya. Dalam hal ini, berkonsentrasi pada bagaimana proses dapat disinkronkan. Misalnya, penting bahwa beberapa proses tidak secara bersamaan mengakses sumber daya bersama, seperti printer, tetapi bekerja sama dalam memberikan satu sama lain akses eksklusif sementara.

Ternyata, sinkronisasi dalam sistem terdistribusi seringkali jauh lebih sulit dibandingkan dengan sinkronisasi dalam sistem uniprocessor atau multiprosesor. Masalah dan solusi yang dibahas dalam pembahasan ini, pada dasarnya, agak umum, dan terjadi dalam banyak situasi berbeda dalam sistem terdistribusi. Dengan dimulai dengan diskusi tentang masalah sinkronisasi berdasarkan waktu aktual, diikuti oleh sinkronisasi di mana hanya masalah pemesanan relatif daripada pemesanan dalam waktu absolut.

Dalam sistem terpusat, waktu tidak ambigu. Ketika suatu proses ingin mengetahui waktu, itu membuat panggilan sistem dan kernel mengatakannya. Jika proses A meminta waktu. dan kemudian sedikit kemudian proses B meminta waktu, nilai yang B dapatkan akan lebih tinggi dari (atau mungkin sama dengan) nilai yang didapat A. Ini tentu tidak akan lebih rendah. Dalam sistem tidak terdistribusi, mencapai kesepakatan tentang waktu bukanlah hal sepele. Jika suatu program terdiri dari 100 file, tidak harus mengkompilasi ulang semuanya karena satu file telah diubah sangat meningkatkan kecepatan di mana programmer dapat bekerja.

Cara membuat biasanya berfungsi adalah sederhana. Ketika pemrogram selesai mengubah semua file sumber, ia menjalankan make, yang memeriksa waktu di mana semua file sumber dan objek terakhir diubah. Jika file sumber input.c memiliki waktu 2151 dan objek file input.o yang sesuai memiliki waktu 2150, pastikan bahwa input.c telah diubah sejak input.o dibuat, dan dengan demikian input.c harus dikompilasi ulang. Di sisi lain, jika output.c hastime 2144 dan output.o punya waktu 2145, tidak diperlukan kompilasi. Dengan demikian make menelusuri semua file sumber untuk mencari tahu yang mana yang perlu dikompilasi ulang dan memanggil kompiler untuk mengkompilasi ulang mereka. Sekarang bayangkan apa yang bisa terjadi dalam sistem terdistribusi di mana tidak ada kesepakatan global tentang waktu.

Misalkan output.o memiliki waktu 2144 seperti di atas, dan tidak lama kemudian output.c dimodifikasi tetapi diberi waktu 2143 karena jam pada mesinnya sedikit di belakang, seperti yang ditunjukkan pada Gambar 6-1. Make tidak akan memanggil kompilator. Program biner yang dapat dieksekusi kemudian akan berisi campuran file objek dari sumber lama dan sumber baru. Ini mungkin akan crash dan programmer akan menjadi gila mencoba memahami apa yang salah dengan kode.



Gambar 1. Ketika masing-masing mesin memiliki jamnya sendiri, suatu peristiwa yang terjadi setelah peristiwa lain mungkin akan ditetapkan sebelumnya

1. Physical Clocks

Hampir semua komputer memiliki sirkuit untuk melacak waktu. Meskipun penggunaan kata "jam" secara luas digunakan untuk perangkat ini, mereka sebenarnya bukan jam dalam arti biasa. Timer mungkin kata yang lebih baik. Timer komputer biasanya kristal kuarsa mesin tepat. Ketika disimpan di bawah tekanan, kristal kuarsa berosilasi pada frekuensi yang ditentukan dengan baik yang tergantung pada jenis kristal, bagaimana itu dipotong, dan jumlah ketegangan. Terkait dengan masing-masing kristal adalah dua register, counter dan register holding. Setiap osilasi kristal menurunkan counter dengan satu. Ketika penghitung mencapai nol, interupsi dihasilkan dan penghitung dimuat ulang dari holding register.

Dengan cara ini, dimungkinkan untuk memprogram timer untuk menghasilkan interupsi 60 kali per detik, atau pada frekuensi lain yang diinginkan. Setiap interupsi disebut satu centang jam. Ketika sistem di-boot, biasanya ia meminta pengguna untuk memasukkan tanggal dan waktu, yang kemudian dikonversi ke jumlah kutu setelah beberapa tanggal awal yang diketahui dan disimpan dalam memori. Sebagian besar komputer memiliki RAM CMOS yang didukung baterai khusus sehingga tanggal dan waktu tidak perlu dimasukkan pada boot berikutnya. Pada setiap tick jam, prosedur layanan interupsi menambahkan satu ke waktu yang tersimpan dalam memori. Dengan cara ini, jam (perangkat lunak) tetap uptodate.

2. Global Positioning System

Sebagai langkah menuju masalah sinkronisasi jam aktual, pertama-tama mempertimbangkan masalah terkait, yaitu menentukan posisi geografis seseorang di mana pun di Bumi. Masalah penentuan posisi ini dengan sendirinya dipecahkan dengan sangat spesifik. sistem terdistribusi khusus, yaitu GPS, yang merupakan akronim untuk sistem penentuan posisi global. GPS adalah sistem terdistribusi berbasis satelit yang diluncurkan pada tahun 1978. Meskipun telah digunakan terutama untuk aplikasi militer, dalam beberapa tahun terakhir GPS telah menemukan jalannya ke banyak aplikasi sipil, terutama untuk navigasi lalu lintas. Namun, ada lebih banyak domain aplikasi.

Misalnya, ponsel GPS sekarang memungkinkan penelepon melacak posisi masing-masing, fitur yang mungkin terlihat sangat berguna saat Anda tersesat atau dalam masalah. Prinsip ini dapat dengan mudah diterapkan untuk melacak hal-hal lain juga, termasuk hewan peliharaan, anak-anak, mobil, kapal, dan sebagainya. Ikhtisar GPS yang sangat baik

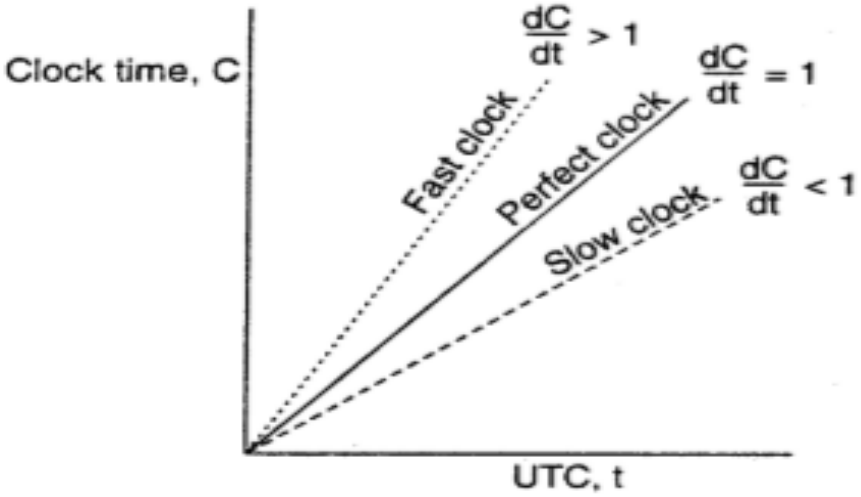
3. Clock Synchronization Algorithms

Semua algoritma memiliki model dasar sistem yang sama. Setiap mesin diasumsikan memiliki timer yang menyebabkan interupsi H kali per detik. Ketika timer ini padam, pengendali interupsi menambahkan 1 ke jam perangkat lunak yang melacak jumlah kutu (interupsi) sejak beberapa waktu yang disepakati di masa lalu. Lalu sebut nilai jam ini C. Lebih khusus lagi, ketika waktu UTC adalah t, nilai jam pada mesin p adalah CpU) .Dalam dunia yang sempurna, akan memiliki CpU) = t untuk allp dan semua t. Dengan kata lain, C; U) = dCldt idealnya harus 1.C; (t) disebut frekuensi p '»jam pada waktu t. Kemiringan jam didefinisikan sebagai C; (t) - 1 dan menunjukkan sejauh mana yang frekuensinya berbeda dengan clock yang sempurna.

Offset relatif terhadap waktu tertentu t CpU) - t. Penghitung waktu nyata tidak mengganggu tepat H kali sedetik. Secara teoritis, timer dengan H = 60 harus menghasilkan 216.000 kutu per jam. Dalam praktiknya, kesalahan relatif yang dapat diperoleh dengan chip pengatur waktu modem adalah sekitar 10-5, yang berarti bahwa mesin tertentu dapat mendapatkan nilai di kisaran 215.998 hingga 216.002 kutu per jam. Lebih tepatnya, jika ada beberapa hal yang konstan

$$1 - \rho \leq \frac{dC}{dt} \leq 1 + \rho$$

timer dapat dikatakan bekerja dalam spesifikasinya. Konstanta p ditentukan oleh pabrikan dan dikenal sebagai laju drift maksimum. Perhatikan bahwa laju drift maksimum menentukan sejauh mana kemiringan jam dibiarkan berfluktuasi. Jam lambat, sempurna, dan cepat ditampilkan di Gambar 1.



Gambar 1. Hubungan antara waktu jam dan UTe saat jam dicentang pada tingkat yang berbeda.

LOGICAL CLOCKS

Sejauh ini diasumsikan bahwa sinkronisasi jam secara alami terkait dengan waktu nyata. Namun, juga dapat dilihat bahwa mungkin cukup bahwa setiap node menyetujui waktu saat ini, tanpa waktu itu harus sama dengan waktu sebenarnya. Dapat melangkah lebih jauh. Untuk menjalankan make, misalnya, cukup bahwa dua node setuju bahwa input.o sudah usang oleh versi baru input.c. Dalam hal ini, melacak kejadian masing-masing (seperti memproduksi versi input.c baru) adalah yang penting. Untuk algoritma ini, adalah konvensional untuk berbicara tentang jam sebagai jam logis.

1. Jam Logical Lamport

Untuk menyinkronkan jam logis, Lamport mendefinisikan hubungan yang disebut terjadi sebelumnya. Ekspresi a ~ b dibaca "a terjadi sebelum b" dan berarti bahwa semua proses setuju bahwa peristiwa pertama a terjadi, kemudian sesudahnya, peristiwa b terjadi. Relasi terjadi-sebelum dapat diamati secara langsung dalam dua situasi: 1. Jika a dan b adalah peristiwa dalam proses yang sama, dan a terjadi sebelum b, maka a ~ b benar. 2. Jika a adalah peristiwa dari suatu pesan yang dikirim oleh satu proses, dan b adalah peristiwa dari pesan yang diterima oleh proses lain, maka a ~ b

2. Vector Clocks

Jam logis Lamport mengarah ke situasi di mana semua peristiwa dalam sistem terdistribusi benar-benar dipesan dengan properti bahwa jika peristiwa terjadi sebelum peristiwa b, maka surat wasiat juga akan ditentukan dalam pemesanan sebelum b, yaitu, C (a) <C (b). Namun, dengan jam Lamport, tidak ada yang bisa dikatakan tentang hubungan antara dua peristiwa a dan b dengan hanya membandingkan nilai waktu mereka C (a) dan C (b), masing-masing. Dengan kata lain, jika C (a) <C (b), maka ini tidak berarti bahwa memang pernah terjadi sebelumnya b. Sesuatu yang lebih dibutuhkan untuk itu.

Namun, dengan jam Lamport, tidak ada yang bisa dikatakan tentang hubungan antara dua peristiwa a dan b dengan hanya membandingkan nilai waktu mereka C (a) dan C (b), masing-masing. Dengan kata lain, jika C (a) <C (b), maka ini tidak berarti bahwa memang pernah terjadi sebelumnya b. Sesuatu yang lebih dibutuhkan untuk itu.

MUTUAL EXCLUSION

Fundamental untuk sistem terdistribusi adalah konkurensi dan kolaborasi antara berbagai proses. Dalam banyak kasus, ini juga berarti bahwa proses perlu mengakses sumber daya yang sama secara bersamaan. Untuk mencegah akses serentak seperti itu merusak sumber daya, atau membuatnya tidak konsisten, solusi diperlukan untuk memberikan akses timbal balik yang saling menguntungkan melalui proses. Pada bagian ini, dapat dilihat beberapa algoritma terdistribusi yang lebih penting yang telah diusulkan.

Algoritme eksklusi mutual terdistribusi dapat diklasifikasikan ke dalam dua kategori yang berbeda. Dalam solusi berbasis token saling pengecualian dicapai dengan mengirimkan pesan khusus antara proses, yang dikenal sebagai token. Hanya ada satu token yang tersedia dan siapa yang pernah memiliki token itu diizinkan untuk mengakses sumber daya bersama. Setelah selesai, token diteruskan ke proses selanjutnya. Jika suatu proses memiliki token tidak tertarik mengakses sumber daya, itu hanya meneruskannya. Solusi berbasis token memiliki beberapa sifat penting. Pertama, tergantung pada bagaimana prosesnya diatur, mereka dapat dengan mudah memastikan bahwa setiap proses akan mendapat kesempatan untuk mengakses sumber daya.

Centralized Algorithm

Cara paling mudah untuk mencapai mutual.exclusion dalam sistem terdistribusi adalah dengan mensimulasikan bagaimana hal itu dilakukan dalam sistem satu-prosesor. Satu proses dipilih sebagai koordinator. Setiap kali suatu proses ingin mengakses sumber daya bersama, ia mengirimkan pesan pertanyaan kepada koordinator yang menyatakan sumber daya mana yang ingin diakses dan meminta izin. Jika tidak ada proses lain saat ini mengakses sumber daya itu, koordinator mengirimkan kembali izin pemberian jawaban

Decentralized Algorithm

Memiliki koordinator tunggal seringkali merupakan pendekatan yang buruk. untuk menggunakan algoritma pemungutan suara yang dapat dieksekusi menggunakan sistem berbasis aDHT. Intinya, solusi mereka memperluas koordinator pusat dengan cara berikut. Setiap sumber daya diasumsikan direplikasi n kali. Setiap replika memiliki koordinator sendiri untuk mengendalikan akses dengan proses bersamaan. Namun, setiap kali suatu proses ingin mengakses sumber daya, ia hanya perlu mendapatkan suara mayoritas dari 111> nl2 koordinator.

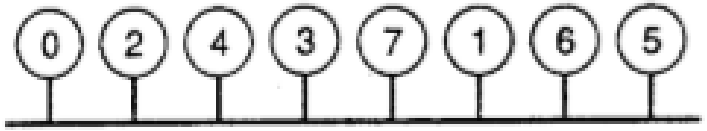
Distributed Algorithm

Bagi banyak orang, memiliki algoritma yang benar secara aprobabilistically tidak cukup baik. Jadi para peneliti telah mencari algoritma eksklusi mutual mutual terdistribusi deterministik. Makalah Lamport tahun 1978 tentang sinkronisasi jam menyajikan yang pertama. Ricart dan Agrawala (1981) membuatnya lebih efisien. Algoritma Ricart dan Agrawala mensyaratkan bahwa ada urutan total semua peristiwa dalam sistem. Artinya, untuk setiap pasangan peristiwa, seperti pesan, itu harus jelas mana yang sebenarnya terjadi terlebih dahulu. Algoritma Lamport disajikan dalam salah satu cara untuk mencapai pemesanan ini dan dapat digunakan untuk memberikan stempel waktu untuk pengeluaran bersama yang didistribusikan.

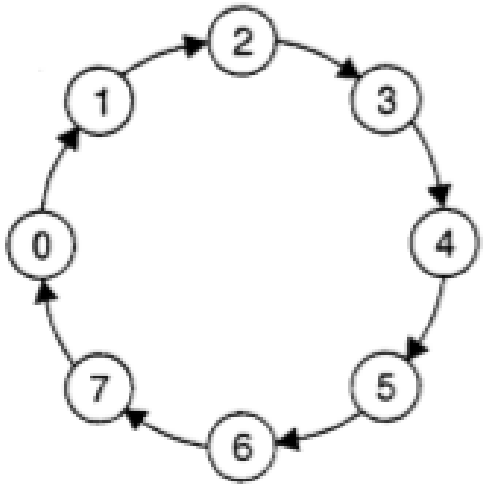
Token Ring Algorithm

Pendekatan yang sama sekali berbeda untuk mencapai saling pengecualian secara deterministik dalam sistem terdistribusi. Pada jaringan bus, seperti yang ditunjukkan pada Ethernet, tanpa urutan proses yang melekat. Dalam perangkat lunak, cincin logis dibangun di mana setiap proses ditugaskan posisi di cincin. Posisi dering dapat dialokasikan dalam urutan numerik alamat jaringan atau cara lain. Tidak masalah apa pemesanannya. Yang penting adalah bahwa setiap proses mengetahui siapa yang akan mengikuti inline berikutnya dengan sendirinya.

Token Ring Algorithm



(A)



(B)

(A)Sekelompok proses yang tidak berurutan di jaringan.
(B) Cincin logis dibangun dalam perangkat lunak.

