METODE-METODE PENENTUAN REPLIKA UNTUK REPLIKASI DATA PADA SISTEM TERDISTRIBUSI

Adkhan Sholeh

Program Studi Manajemen Informatika
STMIK Jenderal A. Yani Yogyakarta

adkhan75@live.com

Abstrak

Jaringan komputer telah berkembang baik dari sisi jumlah node yang tersambung maupun pada ragam layanan yang ditawarkan. Layanan-layanan jaringan komputer ini bertumpu pada sejumlah data yang dikelola oleh node pemilik data. Pada jaringan yang besar seperti Internet, besarnya jumlah node pengakses data dan sebaran pengakses data dari seluruh dunia menjadikan penyedia layanan mengalami overload dan kualitas layanan menurun. Hal ini melatarbelakangi diciptakannya sistem terdistribusi, suatu rancangan sistem komputer yang antara lain bertujuan membagi beban komputasi dan beban pelayanan terhadap client ke banyak node. Guna membagi beban permintaan layanan berbasis data, telah dibuat berbagai metode untuk mereplikasikan data, dengan maksud agar ada banyak node yang dapat menyediakan data dimaksud. Replikasi data dilakukan pada node-node yang kemudian disebut node replika. Penentuan node mana yang akan menjadi replika harus mempertimbangkan faktor-faktor tertentu agar replika yang dibuat dapat memberikan layanan secara optimal.

Kata Kunci: replika, replikasi data, sistem terdistribusi.

1. Pendahuluan

Pertumbuhan jaringan komputer, khususnya Internet, terlihat jelas berdasarkan jumlah *node* yang tersambung ke internet dan berdasarkan ragam layanan/aplikasi yang diakses melalui jaringan ini. Terdapat banyak situs internet yang populer dan layanannya diakses dari seluruh dunia sepanjang waktu. Volume permintaan layanan yang tinggi, beragamnya kualitas layanan jaringan, serta jauhnya jarak node pengakses terhadap penyedia layanan – baik letak geografis maupun berdasarkan koneksi jaringan, merupakan faktor-faktor yang menyebabkan turunnya kualitas layanan kepada node pengakses. Sistem terdistribusi dibuat dengan sasaran membagi beban komputasi dan mendekatkan *node* penyedia data kepada *node* pengakses layanan. Sistem terdistribusi berusaha mengatasi masalah tersebut, salah satunya dengan membuat mekanisme replikasi data. Makalah ini menyajikan metode-metode penentuan *node* replika dalam sistem terdistribusi. *Node* replika adalah *node* dalam jaringan komputer yang akan digunakan untuk replikasi data.

2. Replikasi Data dalam Sistem Terdistribusi

Replikasi data dilakukan dengan menyalin dan mendistribusikan data dari *node* pemilik data ke *node* lain yang memerlukan. Replikasi data juga menyangkut sinkronisasi data untuk menjamin konsistensi data tersebut. Melalui teknik replikasi ini, data dapat didistribusikan ke *node* yang berbeda melalui koneksi jaringan lokal maupun internet. Dengan adanya replika tersebut maka layanan berbasis data ini tidak lagi ditangani oleh satu komputer. Pada akhirnya, replikasi file ini berupaya meningkatkan reliabilitas sistem dan meningkatkan kinerja sistem secara keseluruhan (Gunadarma, 2010).

Peningkatan reliabilitas sistem dapat dirasakan ketika *file* utama mengalami kerusakan atau tidak dapat diakses, maka dimungkinkan untuk mengakses salinan *file* tersebut pada *node* yang lain (replika). Sedangkan peningkatan kinerja sebagai efek replikasi *file* antara lain dapat ditunjukkan ketika sistem memperoleh beban permintaan akses dalam *volume* besar. Apabila *file* hanya dimiliki oleh satu komputer/*server*, maka akan terjadi antrian yang panjang. Sebagian permintaan akses tersebut tidak bisa dilayani secara cepat. Replikasi file memungkinkan permintaan tersebut disebar ke dan dipenuhi oleh banyak *node* yang mempunyai file yang sama. Kinerja yang lebih baik – berupa waktu respon yang lebih cepat – juga diperoleh pada replikasi *file* ke komputer-komputer yang secara geografis maupun secara koneksi jaringan lebih dekat dengan komputer-komputer yang mengajukan permintaan akses.

3. Manfaat Replikasi

Manfaat teknik replikasi tergantung dari jenis replikasi yang dilakukan dan pada tingkat mana *user* mengakses suatu data. Secara umum replikasi mendukung ketersediaan data dan meningkatkan kinerja.

Berikut ini beberapa contoh (Tanenbaum dan Steen, 2006) manfaat yang bisa diperoleh dari replikasi:

1. Memungkinkan beberapa *node* memiliki data yang sama. Hal ini secara tidak langsung berfungsi sebagai *back up* atau cadangan. Selain itu, bila akses ke satu *node* tidak menemukan data yang dimaksud, maka dimungkinkan untuk menemukan data tersebut di *node* yang lain.
2. Memungkinkan rancangan aplikasi transaksi *online* yang memisahkan bagian antarmuka sistem dengan bagian yang mengelola data.
3. Dapat menghemat biaya akses. Contohnya sudah dilakukan oleh situs-situs *web* luar negeri yang membuat replika *web service* di dalam negeri,

sehingga penggunaan *bandwidth* jaringan ke luar negeri yang biayanya lebih mahal bisa dikurangi.

4. Meningkatkan kualitas layanan, antara lain dengan mempersingkat waktu respon terhadap permintaan akses data.

Meski meningkatkan reliabilitas dan kinerja, replikasi data juga memunculkan masalah. Salah satunya terkait dengan konsistensi data. Jika salah satu data mengalami perubahan secara sah/valid, maka replika data lainnya akan menjadi tidak valid. Hal ini memaksa modifikasi dilakukan ke seluruh replika.

4. *Replica Hit*

Efektifitas suatu replika dinyatakan dengan angka *replica hit*. *Replica hit* terjadi ketika satu permintaan akses terhadap *file* justru dipenuhi oleh komputer/*node replica*, bukan *node* pemilik file orisinalnya. Suatu metode replikasi yang efektif akan menghasilkan angka *replica hit* yang tinggi. *Replica hit* dihitung sebagai persentase dari jumlah *query* yang diselesaikan oleh replika dibagi dengan total jumlah *query* (Shen, 2010).

5. Metode-Metode Replikasi

Hingga saat ini terdapat beberapa metode replikasi, yang diklasifikasikan menjadi tiga kategori. Ketiganya adalah *ServerSide*, *ClientSide*, dan *Path* (Shen, 2010).

Metode *ServerSide* mereplikasikan suatu file berdekatan dengan *node* pemilik file tersebut. Cara ini memiliki kelebihan pada pencapaian *replica hit* yang baik dan efisiensi *query* untuk mengakses file. Kerugiannya, metode ini tidak memperpendek jarak *node* pengakses menuju *node* penyedia file secara signifikan. Menempatkan replika hanya pada *node* yang berdekatan dengan pemilik *file* dengan sendirinya juga membatasi jumlah *node* yang bisa menjadi replika. Akibatnya, hanya akan diperoleh jumlah *node* yang terbatas dan *node-node* tersebut rentan menerima permintaan dalam jumlah yang masih cukup besar.

Beberapa metode yang termasuk dalam kategori *ServerSide* adalah PAST, CFS, Backlash dan Overlook. Metode PAST merupakan fungsi penyimpanan pada jaringan P2P global berbasis Internet. Metode ini dilengkapi manajemen penyimpanan dan sistem *caching*. PAST akan mereplikasi setiap *file* ke sejumlah *node* yang ID *node*-nya paling dekat/mirip dengan ID *node* pemilik

file asal. Jumlah replika dipilih sesuai kebutuhan atas keberadaan suatu *file*, secara relatif terhadap tingkat kegagalan yang mungkin terjadi pada tiap *node*. Selain itu, PAST juga mengantisipasi kondisi apabila operasi mereplikasi *file* gagal dilakukan. Guna memperkecil kelambatan (*latency*) *query* dan menyeimbangkan beban *query*, PAST menerapkan *caching file* di sepanjang jalur pencarian (*lookup path*).

Seperti PAST, *Cooperative File System* (CFS) juga digunakan pada jaringan P2P. CFS melakukan replikasi untuk tujuan pengaksesan secara *read-only* (tidak untuk dimodifikasi). CFS dikembangkan dari teknik Chord dengan mereplikasi blok-blok sebuah *file* ke *node-node* lain segera setelah pemilik blok muncul dalam ring Chord. CFS juga membuat *cache* penunjuk lokasi *file* di jalur pengaksesan untuk meningkatkan efisiensi *query* dan menghindari *overload* pada *server* yang menyimpan *file-file* populer.

Metode Backlash dibangun pada *overlay* jaringan P2P terstruktur dan secara agresif melakukan *caching* pada sebuah *file* yang tinggi permintaannya. *Node-node* Backlash membagi *storage* yang tersedia ke dalam dua kategori, yaitu *replica space* untuk menampung replika dalam waktu lama dan *temporary cache space* untuk penampungan sementara. Replika-replika ditempatkan dalam *overlay* dengan menyisipkan tabel *hash* terdistribusi. *Cache* temporer merupakan salinan dokumen yang ditempatkan secara oportunis pada suatu *node*.

Metode Overlook dikembangkan dari metode Pastry. Overlook menempatkan replika sebuah *file* pada *node* yang paling banyak menerima permintaan pencarian. Overlook melakukan hal ini dengan memilih *node* yang paling sering mem-*forward* permintaan yang diterimanya kepada *node* pemilik *file*. *Node* pemilik *file* akan mengirim permintaan kepada *node* tersebut untuk membuat replika.

ClientSide membuat replikasi pada *node* yang dekat dengan atau pada *node* pengakses *file* tersebut. Ketika *file* yang direplikasi suatu *node* diakses sendiri oleh *node* tersebut atau oleh komputer-komputer di dekatnya, maka akan diperoleh efisiensi *query* yang sangat tinggi. Sayangnya, hal ini tidak sering terjadi. Penyebabnya adalah bervariasinya permintaan *node* terhadap *file* yang ada. Oleh karena itu, *ClientSide* tidak dapat menjamin *replica hit* dan utilisasi replika yang tinggi.

Gnutella, salah satu yang menerapkan *ClientSide*, mereplikasikan *file-file* pada *node* yang mengalami kelebihan permintaan ke *node-node* yang meminta

file-file tersebut. *Node* hanya akan menyimpan dan melayani *file* yang pernah dia minta. Selain Gnutella terdapat juga FarSite, sebuah sistem *file* tradisional yang tingkat reliabilitas dan ketersediaannya bagus. FarSite mereplika jumlah *file* yang sama di komputer *client* untuk mencapainya. Metode LAR juga termasuk kategori *ClientSide*. LAR merupakan protokol replikasi yang ringan, adatif, dan netral terhadap sistem. Dia mengukur tingkat *overload* sebuah *server* untuk memutuskan apakah sebuah *file* perlu direplikasi kepada *client*. Selain mereplikasi *file* kepada *client* yang memintanya, LAR juga mereplikasi petunjuk-petunjuk lokasi *file* di sepanjang jalur *lookup*. Ketika sebuah replika baru dibuat, LAR akan memasang status *cache* pada jalur dari replika baru ke *node* yang membuat replika tersebut.

Adapun metode *Path*, akan membuat replika pada *node* yang ada di jalur (*path*) yang menghubungkan pengakses dengan pemilik *file* tersebut. Prinsip *Path* adalah menghindari masalah-masalah yang dijumpai pada metode *ServerSide* dan *ClientSide*. Dengan beragamnya *node* asal yang melakukan permintaan, maka jalur yang terbentuk antara *node-node* pengakses *file* menuju *node* pemilik *file* juga beragam. *Path* berupaya agar *replica hit* tetap tinggi dan memperpendek jalur pencarian. Akibat banyaknya variasi jalur, maka efektivitas *Path* harus dibayar dengan keharusan melakukan replikasi dan memelihara replika dalam jumlah yang lebih besar dibanding dua metode sebelumnya. Kekurangan lain dari *Path* adalah munculnya replika-replika yang tingkat pemanfaatannya rendah (*underutilized*).

Salah satu yang menerapkan *Path* adalah Freenet, yang mereplikasi *file* pada jalur antara peminta *file* dengan pemilik *file*. Mekanisme permintaan Freenet akan secara transparan mereplikasikan data yang populer dan menempatkannya di jalur yang lebih dekat dengan *node-node* peminta.

Controlled Udate Propagation atau CUP merupakan protokol untuk memelihara *cache* metadata pada jaringan-jaringan P2P. Propagasi atau penyebaran dilakukan dengan membuat pohon CUP yang mirip dengan pohon *multicast* pada level aplikasi. Sebuah *node* secara proaktif menerima pembaruan untuk item-item metadata dari *node* tetangga hanya jika *node* tersebut mengirim permintaan atas data yang diinginkannya pada *node* tetangga tersebut. Sebaliknya, *node-node* perantara di sepanjang jalur yang menerima indeks yang sudah diperbarui meski *node* perantara itu tidak memerlukannya.

Dynamic tree-based Update Propagation atau DUP dibuat untuk meningkatkan CUP. DUP membangun pohon propagasi pembaruan dinamis di atas struktur pencarian indeks yang sudah ada. Karena pohon penyebaran *update* hanya melibatkan *node-node* yang penting untuk penyebaran *update*, *overhead* DUP sangat kecil dan kelambatan *query* dapat dikurangi secara signifikan.

Teknik lain dalam kategori *Path* adalah LessLog. LessLog menggunakan sebuah pohon *template* untuk membuat pohon pencarian binomial yang unik pada tiap *node*. Pohon pencarian tersebut berhubungan dengan waktu pencarian pada O(log N) dalam suatu jaringan P2P N-*node*. Berdasarkan data pada pohon pencarian *template* itu kemudian ditentukan *node-node* mana yang akan menjadi replika atas *node* yang mengalami *overload*. Penentuan ini tidak memerlukan akses informasi kepada *client* (pengakses *file*) (Huang, dkk, 2004).

6. Penutup

Hingga saat ini telah dilakukan berbagai percobaan dan implementasi metode replikasi data untuk sistem terdistribusi. Berdasarkan penentuan letak *node* yang akan dijadikan replika, terdapat tiga jenis *node* replika: *ServerSide*, *ClientSide*, dan *Path*. *ServerSide* menempatkan replika tidak jauh dari pemilik data, sehingga dicapai tingkat *replica hit* yang baik namun tidak signifikan dalam mengurangi jarak tempuh akses *client* terhadap penyedia data. Sebaliknya, *ClientSide* memperpendek jarak akses dengan menempatkan replika lebih dekat atau pada *client*. Terlalu dekatnya replika pada *ClientSide* menimbulkan kerugian rendahnya tingkat penggunaan replika (*underutilized*). Metode *Path* berusaha mengambil jalan tengah dari kedua metode sebelumnya, dengan menempatkan *node-node* replika di sepajang rute akses *client* kepada penyedia data.

Daftar Pustaka

Gunadarma, 2010, *Modul Pelatihan AS/400*, Universitas Gunadarma, Jakarta.

Huang, K.L., Huang, T.Y.H. and Chou, J.C.Y., 2004, LessLog: A Logless File Replication Algorithm for, *Proceeding International Parallel and Distributed Processing Symposium*.

Shen, H., 2010, An Efficient and Adaptive Decentralized File Replication Algorithm in P2P File Sharing Systems, *IEEE Transactions On Parallel and Distributed Systems*, vol. 21, no. 6, hal. 827-840.

Tanenbaum, A.S. dan Steen, M.V., 2006, *Distributed Systems Principles and Paradigms*, Pearson Education Inc.