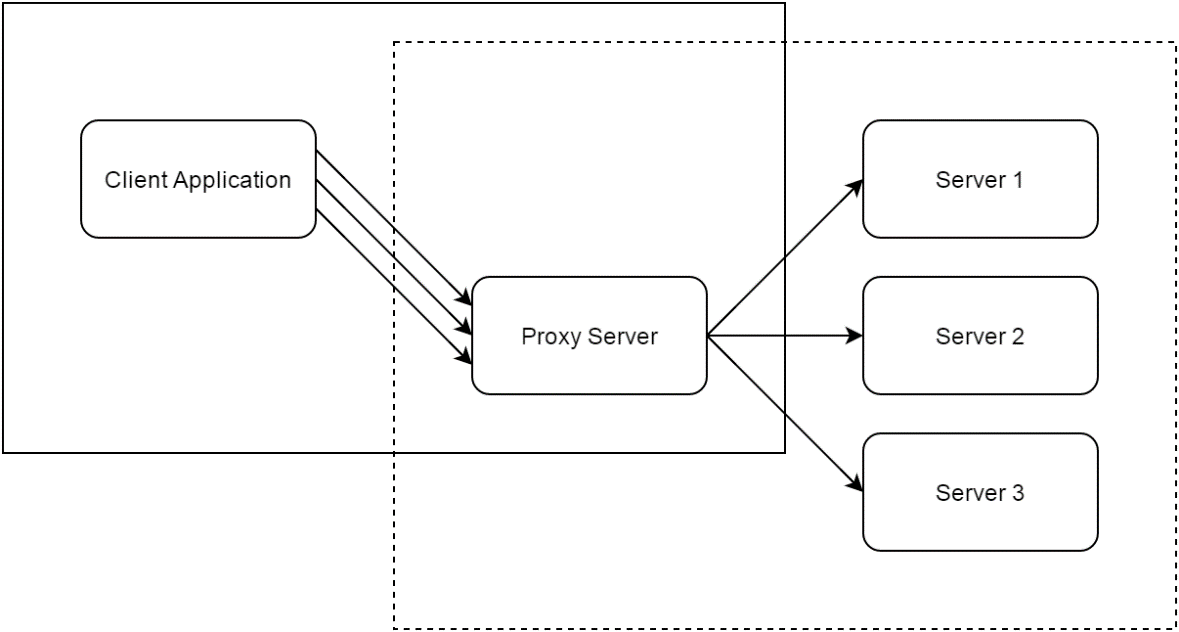
1. **Proxy**

Dalam perspektif sistem terdistribusi, proxy dapat diartikan sebagai server yang berperan sebagai penghubung antara client dan server. Proxy merupakan representasi dari seluruh set dari server, sehingga dalam melakukan seluruh komunikasi, client akan secara langsung terhubung dengan proxy [1, p. 1]. Proxy bersifat fleksibel, yang berarti dapat diprogram secara menyeluruh. Berikut merupakan sifat-sifat dari proxy [1, p. 9]:

1. Encapsulation. Service yang disediakan oleh server dapat diibaratkan sebuah black box, strukturnya tidak dapat diexpose, dan hanya dapat diakses melalui proxy.
2. Locality. Beberapa request dari client akan direspon oleh proxy, dan buffer akan disimpan secara local oleh proxy.
3. Access Protocol. Proxy memberlakukan urutan yang ketat terhadap client (request - acknowledgment - access - release).
4. Capability. Proxy dapat memberlakukan access control, mengetes validitas argument, maupun melakukan suatu operasi. Semua hal tersebut sepenuhnya terprogram.
5. Stub. Proxy merupakan sebuah potongan (stub) [2], melakukan pemaketan data, dan melakukan pengaksesan terhadap jaringan.
6. Trusted Communication. Sebuah proxy dibuat oleh service itu sendiri, sehingga dapat dipastikan komunikasi yang datang ke service berasal dari partner-nya sendiri. Dengan itu, kompleksitas service dapat dikurangi.
7. Protocol Encapsulation. Protokol antara client dengan service terbungkus (encapsulated) dalam sebuah object yang dibentuk oleh proxy dan principal-nya (service(s))



Gambar xx. Ilustrasi Proxy

1. **Jenis-jenis Proxy**

Ada banyak jenis proxy, tetapi ada beberapa jenis diantaranya yang paling umum digunakan [3]:

1. Anonymous Proxy, sering juga disebut dengan web proxy, merupakan tipe proxy yang meng-anonim-kan client dengan cara menyembunyikan IP address dari client.
2. Distorting Proxy, adalah tipe proxy yang mengidentifikasi dirinya sebagai proxy, tetapi membuat IP address yang tidak benar pada HTTP headers.
3. High Anonymity Proxy, merupakan tipe proxy yang tidak mengidentifikasi dirinya sebagai proxy, juga tidak memberikan IP address yang sebenarnya.
4. Intercepting Proxy, merupakan tipe proxy yang menggabungkan antara proxy dengan gateway. Koneksi yang dibuat oleh client diteruskan ke proxy tanpa mengubah konfigurasi pada client. Proxy jenis ini dapat dideteksi oleh server melalui HHTP header.
5. Reverse Proxy, merupakan proxy yang meneruskan request dari internet, melalui firewall, ke private network.
6. Transparent Proxy atau juga dikenal dengan Transparent Forward Proxy, merupakan jenis proxy yang tidak menerapkan local policies, seperti : menambah, mengubah, maupun mengurangi atribut atau isi sebuah informasi. Transparent proxy lebih banyak digunakan untuk cache website, sehingga mengurangi beban server.
7. **Caching**

Caching merupakan sebuah topic yang telah lama dipelajari dalam kaitannya dengan desain memori sistem computer [4]. Cache dapat dikatakan sebagai temporary object yang dapat digunakan kemudian [5]. Kegunaan utama dari caching antara lain [6]: meningkatkan availability dari data, mengurangi latency terhadap client, mengurangi beban dari server, dan mengurangi konsumsi bandwith. Beberapa pendekatan caching yang sekarang banyak digunakan antara lain [7]:

1. Pendekatan back-end caching

Pendekatan back-end telah lama diusulkan untuk mempercepat content yang degenerate secara dinamis, misalnya caching hasil query database pada DBMS dan caching table database di memory utama [8]. Beberapa pendekatan yang berbasis back-end caching beroperasi pada presentation layer, seperti WebLogic dari BEA System dan SpyderCache. Back-end caching beroperasi mirip dengan cara kerja reverse proxy, tetapi caching tipe ini beroperasi pada infrastruktur situs, biasanya sebagai plugin dari Web server [7].

1. Pendekatan proxy-based caching

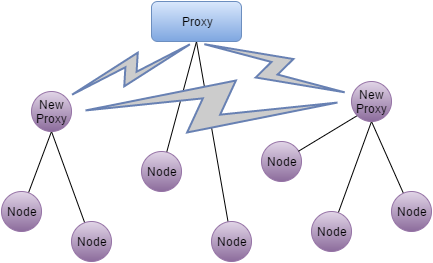
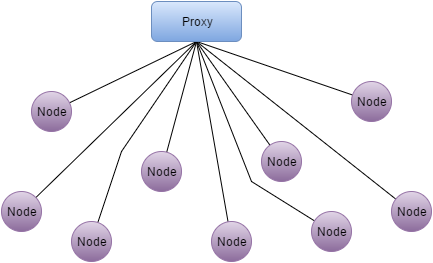
Pendekatan proxy pada caching merupakan pendekatan yang mendasarkan pada caching konten diluar infrastruktur dari sebuah situs. Pada pendekatan ini, digunakan sebuah proxy server untuk menyimpan cache dari sebuah konten.

Dalam kaitannya dengan caching, algoritma penggantian (replacement algorithm) memiliki peranan yang sangat penting dalam mengurangi response time. Algoritma cache replacement biasanya memaksimalkan cache hit ratio dengan men-cache data yang akan banyak digunakan kedepannya. Tetapi karena data yang akan banyak digunakan cenderung susah diprediksi, pendekatan yang paling banyak dipakai adalah dengan menggunakan data yang paling banyak dipakai sebelumnya [9].

Peningkatan pada response time dengan men-cache dokumen tidaklah murah. Cache harus menjaga konsistensi dengan primary server dengan menggenerate extra requests. Konsistensi konten dijaga dengan mengimplementasikan consistency algorithm. Salah satu implementasi dari consistency algorithm adalah Time to Live (TTL). Client menggunakan conditional HTTP headers (If-Modified-Since) untuk mengecek waktu modifikasi terakhir dari cache [9].

1. **Dynamic Proxy**

Pada kondisi tertentu, misalnya ketika data yang harus di-cache sangat besar atau ketika karakteristik client bersifat tersebar, maka diperlukan penggunaan lebih dari sebuah proxy. Tantangan dalam penggunaan multiple proxy adalah membuat proxy bertambah secara otomatis ketika dibutuhkan. Cobarzan dkk mengusulkan sebuah novel proxy-cache system yang dapat men-spawn proxy baru ketika diperlukan dengan menggunakan split operation [6].



Skema rancangan Cobarzan dkk membagi proxy menjadi dua komponen, yaitu dispatchers dan daemons. Dispatchers merupakan proses atau thread yang berjalan di node yang sama dengan proxy dan dapat dianggap sebagai front-end dari proxy. Fungsi dari dispatchers meliputi :

1. Meng-handle incoming requests

Merespon request baik dari local cache maupun dari original server. Jika tidak memungkinkan, request akan di-forward ke dispatcher/proxy yang lain atau request ditolak. Jika terdapat lebih dari satu dispatcher yang dapat di-forward, maka kandidat terbaik dipilih. Pemilihan kandidat didasarkan pada ketersediaan local cache atau beban yang terkecil.

1. Me-manage kode proxy

Dispatcher menyimpan kode proxy dan mengirimkannya kepada node dimana proxy yang baru akan dijalankan (dengan mendeteksi daemon/service yang berjalan pada node target).

1. Me-manage child proxy process

Dispatcher bertanggungjawab pada proses stop/pause/restart child proxy yang dijalankannya. Proses stop/pause/restart didasarkan pada beban, jumlah client yang ditangani, maupun volume local data.

Sementara itu, daemon atau service secara default berjalan pada setiap node yang terkoneksi pada jaringan. Daemon mempunyai tanggung jawab :

1. Me-manage kode proxy

Daemon menerima kode yang dikirimkan dispatcher pada proxy yang lain yang menginisiasi proxy split operation

1. Me-manage proses stop/pause/restart

Proses stop/pause/restart pada proxy selain dapat di manage oleh proxy induknya, juga dapat di manage oleh daemon yang berjalan pada node tersebut, tergantung pada kondisi pada node tersebut.

REFERENCES

[1] M. Shapiro, “Structure and Encapsulation in Distributed Systems: the Proxy Principle,” presented at the Int. Conf. on Distr. Comp. Sys. (ICDCS), 1986, pp. 198–204.

[2] B. J. Nelson, “Remote procedure call,” *CERN Document Server*, 1981. [Online]. Available: http://cds.cern.ch/record/132187. [Accessed: 08-May-2016].

[3] “Types of Proxy Servers, Transparent and Anonymous Proxies,” *WebToolHub*. [Online]. Available: http://info.webtoolhub.com/kb-a14-types-of-proxy-servers-transparent-and-anonymous-proxies.aspx. [Accessed: 08-May-2016].

[4] F. J. Hill and G. R. Peterson, *Digital systems: hardware organization and design*. John Wiley & Sons, Inc., 1978.

[5] B. D. Davison, “A survey of proxy cache evaluation techniques,” in *Proceedings of the Fourth International Web Caching Workshop (WCW99)*, 1999, pp. 67–77.

[6] C. Cobârzan, “Dynamic Proxy-Cache Multiplication Inside LANs,” in *Euro-Par 2005 Parallel Processing*, J. C. Cunha and P. D. Medeiros, Eds. Springer Berlin Heidelberg, 2005, pp. 890–900.

[7] A. Datta, K. Dutta, H. Thomas, D. VanderMeer, Suresha, and K. Ramamritham, “Proxy-based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation,” in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2002, pp. 97–108.

[8] Q. Luo, J. F. Naughton, R. Krishnamurthy, P. Cao, and Y. Li, “Active query caching for database web servers,” in *The World Wide Web and Databases*, Springer, 2000, pp. 92–104.

[9] J. Shim, P. Scheuermann, and R. Vingralek, “Proxy cache algorithms: Design, implementation, and performance,” *Knowl. Data Eng. IEEE Trans. On*, vol. 11, no. 4, pp. 549–562, 1999.