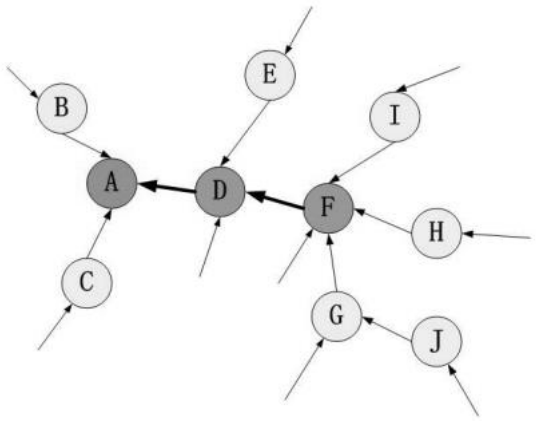


شکل (۲) : درخواست‌های مشتری‌ها در سیستم ذخیره‌سازی ابری جهانی [11]

دسترس‌پذیری سطوحی دارد که از آنجایی که روش مکان‌دهی ارائه‌شده در این مقاله یک رویکرد مکان‌دهی بین مرکزداده‌ای است لذا بالاترین سطح دسترس‌پذیری داده را دارد [11].

روش پیشنهادی که از یک رفتار ترافیک‌گرا استفاده می‌کند، مرکزداده‌های با بیشترین ترافیک ایجادشده حاصل از حرکت درخواست‌ها در مسیر مستقیم مسیریابی به سمت مقصد، را برای مکان‌دهی داده‌ها انتخاب می‌کند. با توجه به شکل (۲) مرکز داده‌های D و F که به طور تقریبی در مسیر ضروری بیشتر درخواست‌ها وجود دارند و به مانند گره‌های قطب در نظر گرفته می‌شوند؛ برای تکرار داده‌ی موجود در A ترجیح داده می‌شوند.

۳-۲- تعیین ترافیک

ترافیک با پرس‌وجو متفاوت است. از آنجایی که روش پیشنهادی مبتنی بر ترافیک هست، ابتدا مفهوم ترافیک و تفاوت آن با پرس‌وجو بیان می‌شود و سپس شیوه‌ی محاسبه‌ی آن برای هر واحد داده بیان می‌شود.

پرس‌وجو، اطلاعات درخواستی است که مشتری آن را برای دسترسی به یک واحد داده خاص می‌فرستد. این پرس‌وجو در فرآیند مسیریابی در هر گره ترافیکی ایجاد می‌کند؛ که این ترافیک بسته به فرآیند مسریابی می‌تواند متفاوت باشد.

منظور از ترافیک، ترافیک رو به جلو در مسیر مستقیم به سمت مقصد هست که در گره‌ها (در اینجا منظور از گره، یک مرکز داده است) ایجاد می‌شود؛ و ترافیکی که حاصل از ارسال پرس‌وجو به گره‌های همسایه هست، مدنظر نیست. درخواست‌دهنده‌ی j برای دسترسی به واحد داده F_i از یک مسیر می‌گذرد که در تمام گره‌های A_{ij} موجود در این مسیر، ترافیک ایجاد می‌کند. رابطه (۱) نشان می‌دهد.

$$A_{ij} = \{N_k \mid N_k \text{ is at the routing path from } N_i \text{ to } N_j\} \quad ,[11] \quad (۱)$$

مطابق رابطه‌ی (۲) ترافیک اولین گره موجود در مسیر مسیریابی، برابر با پرس‌وجوی وارد شده به آن است.

$$tr_{ijjt} = q_{ijt} \quad ,[11] \quad (۲)$$

هر گره N_k یا در مسیر مسیریابی هست و یا نیست. یعنی احتمال آن $P_{ijk} \in \{0,1\}$ است. بنابراین ترافیک گره N_k برای واحد داده F_i در بازه زمانی t مطابق رابطه‌ی (۳) محاسبه می‌شود.

$$tr_{ikt} = \sum_{j=1}^{N} \max(0, q_{ijt} - \sum_{k^x \in A_{jk}} c_{ik}) \cdot p_{ijk} \quad (۳)$$

که C_{ik} ظرفیت گره N_k است برای واحد داده F_i است. به معنی تعداد درخواست‌هایی که می‌تواند در یک دوره پاسخ دهد. به منظور جلوگیری از شیب تند تغییرات نرخ پرس‌وجوها، از تاریخچه‌ی اطلاعات استفاده می‌کنیم و برای این منظور از فاکتور هموارسازی α استفاده می‌کنیم. بنابراین میانگین ترافیک سیستم برای دوره t از طریق رابطه‌ی (۴) به دست می‌آید.

$$tr_{ikt} = \alpha.tr_{ik(t-1)} + (1-\alpha).tr_{ikt}, 0 < \alpha < 1 \quad ,[11] \quad (۴)$$

۳-۳- تعیین تعداد تکرار برای هر واحد داده

برای تعیین حداقل تعداد تکرارهای هر واحد داده از حداقل دسترس‌پذیری مورد نیاز (که در قرارداد سطح سرویس تعیین می‌شود) استفاده می‌شود.

$$1 - \sum_{j=1}^{m} (-1)^{j+1} C_m^j \left(\prod_{i=1}^{r_j} f_i\right) \geq A_{\text{expect}} \quad ,[15] \quad (۵)$$

که f_i احتمال شکست و r_i هم تعداد تکرارهای واحد است.

همه واحدهای داده باید حداقل به تعدادی که از رابطه (۵) حاصل می‌شود، تکرار شوند. اما با توجه به محبوبیت‌شان می‌توانند بیشتر تکثیر شوند. که معیار محبوبیت، حجم ترافیک ایجاد شده برای آن واحد است. اگر این ترافیک در هر گره از حدآستانه تکرار بیشتر بود باید این داده در آن گره‌ها تکرار شود. تعداد تکرارها از رابطه‌ی (۶) محاسبه می‌شود.

$$r = max\ (r_{min}\ ,\ number\ of\ (\{N_k \mid tr_{ikt} > threshold_{replication}\})) \quad (۶)$$

۳-۴- شرایط مهاجرت و حذف تکرار هر داده

هنگامی که ترافیک یک گره برای داده کمتر از حدآستانه تکرار باشد و آن گره شامل یک نسخه از آن داده باشد. باید یا آن نسخه را حذف کند، یا داده به مرکزداده‌ای که به عنوان مقصد انتخاب شده، مهاجرت کند و یا در غیر این صورت آن را نگه دارد. اگر شرط مهاجرت برقرار بود، داده به مقصد مناسب مهاجرت می‌کند. اگر شرط مهاجرت برقرار نبود، در صورتی که تعداد تکرارها بیش از تعداد تعیین شده بود، آن را حذف می‌کند. در غیر این صورت داده را نگه می‌دارد.

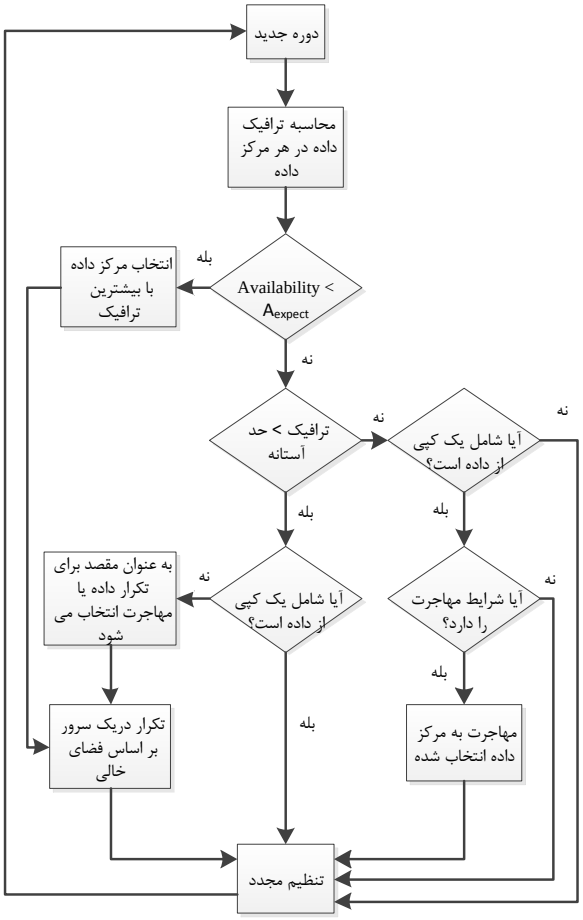
شرط مهاجرت یک داده از مبدأ به مقصد به این صورت است که اگر اختلاف ترافیک مبدأ و مقصد از حدآستانه مهاجرت بیشتر باشد، مهاجرت از مبدأ به مقصد صورت می‌گیرد و اگر کمتر باشد مهاجرت انجام نمی‌شود. اگر شرط رابطه‌ی (۷) برقرار باشد واحد داده F_i از گره‌ی k به گره‌ی k' مهاجرت می‌کند.

$$tr_{ik't} - tr_{ikt} > threshold_{migration} \quad (۷)$$

۳-۵- درخت تصمیم‌گیری

خوشه‌های سیستم روش پیشنهادی در هر مرکزداده مبتنی بر HDFS است. در هر مرکزداده گره‌نام[۸] ترافیک هر واحد داده را محاسبه می‌کند. مولفه‌ی مدیر تکرار[۹] که مولفه‌ی مرکزی اصلی برای تصمیم‌گیری تکرارداده در روش پیشنهادی است، به صورت دوره‌ای این اطلاعات را جمع‌آوری می‌کند و برای تکرار واحدهای داده در هر یک از این مرکزداده‌ها تصمیم‌گیری می‌کند. مرکزداده‌ها هم از طریق ارتباطاتی که با هم دارند و از طریق گره‌داده[۱۰]ها، دستورات مدیر تکرار را اجرا می‌کنند.

تصمیم‌گیری بر اساس فلوچارت شکل (۳) انجام می‌شود.



شکل (۳) : فلوچارت تصمیم‌گیری تکرارداده

٤- ارزیابی کارایی

٤-۱- راه‌اندازی آزمایش

یک محیط ذخیره‌سازی ابری مطابق شکل (۲) با استفاده از برنامه‌نویسی جاوا شبیه‌سازی شده است. هر مرکزداده یک خوشه‌ی هادوپ است که برای شبیه‌سازی آن از شبیه‌ساز HDFSsim که در [3] معرفی شده، استفاده شده است. HDFSsim تا ۱۰۰ هزار گره مقیاس‌پذیر است. سپس برای کار پیشنهادی آن را توسعه دادیم. ۱۰ تا مرکزداده از لحاظ جغرافیایی در نقاط مختلف جهان توزیع شده‌اند. از آنجا که سیاست مکان‌دهی بین مرکزداده‌ای است، پس مفهوم رک در مرکزداده نادیده گرفته می‌شود. هر مرکزداده شامل یک گره‌نام و ۱۰ تا گره‌داده است. همه گره‌ها از لحاظ ظرفیت رایانشی یکسان هستند.

داده به بلوک‌هایی تقسیم می‌شود (بر اساس HDFS سایز هر بلوک ۶۴ مگابایت در نظر گرفته می‌شود).

در طول دوره‌ی شبیه‌سازی تعدادی پرس‌وجو با توزیع زمانی پواسون روی سیستم توزیع می‌شود. مطالعات متعدد بر روی درخواست‌های وب و شبکه‌های اجتماعی، استفاده از اصل قانون قدرت که توزیع پرتو آن را مدل می‌کند، برای توزیع درخواست‌های دسترسی روی داده‌ها پیشنهاد می‌دهند [2]. از توزیع پرتو که نزدیک‌ترین توزیع به بارکاری کاربردهای وب است، استفاده شده‌است. تنظیمات آزمایش در جدول شماره (۱) آورده‌شده‌است. آزمایش‌ها در دو دوره انجام می‌گیرد. در دوره‌ی اول ۸۰ درصد درخواست‌ها از سمت مرکزداده‌های I، J و H می‌آیند؛ سپس با یک تغییر شدید در دوره‌ی دوم ۸۰ درصد درخواست‌ها از سمت مرکزداده‌های A، B و C می‌آیند.

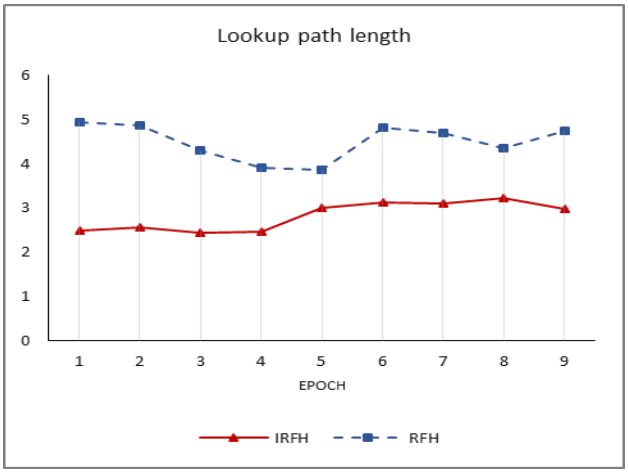
جدول (۱) : تنظیمات شبیه‌سازی

تعداد گره‌ها در سیستم	۱۰۰
حداقل دسترس‌پذیری داده	۹۹ درصد
تعداد داده‌های توزیع شده در سیستم	۵۰۰
سایز هر داده	۲۵۶ مگابایت
پهنای‌باند بین مرکزداده‌ها	۱۰۲۴ مگابیت بر ثانیه
تعداد بلوک‌های داده	۲۰۰۰
حدآستانه تکرار	۱۰
حدآستانه مهاجرت	۵
نرخ تولید درخواست با توزیع پواسون	۳۰ درخواست بر ثانیه
فاکتور هموارسازی (α)	۰,۲
احتمال خرابی	۰,۱
دوره‌ی تکرار	۳۰۰ ثانیه
زمان شبیه‌سازی	۳۰۰۰ ثانیه

در کارهای متعدد مرتبط انجام‌شده، سایر رویکردهای تکرارداده به دفعات با هم قیاس شده‌اند. روش RFH هم با سه رویکرد موجود تکرارداده مقایسه شده است و کارایی بهتری ارائه کرده است. لذا در این مقاله نیازی به مقایسه روش پیشنهادی با سایر رویکردهای تکرارداده دیده نشد. با مقایسه روش پیشنهادی با روش RFH و رجوع به مقاله‌ی [11] می‌توان این روش را با همه رویکردهای دیگر مورد سنجش قرار داد. بنابراین در آزمایش‌ها روش پیشنهادی تنها با روش RFH مقایسه می‌شود.

٤-۲- طول مسیر مراجعه

طول مسیر مراجعه[۱۱] معیار بسیار مهمی است که از آن می‌توان به عنوان معیاری برای سنجش زمان پاسخ و مصرف پهنای‌باند استفاده کرد. در واقع میانگین فاصله‌ی بین مکان درخواست‌دهنده‌ها تا محل واحد داده‌ی درخواستی را نشان می‌دهد. بدیهی است که هرچه این معیار کوچک‌تر باشد، زمان پاسخ و مصرف پهنای‌باند برای انتقال داده بین مشتری و سیستم کمتر خواهد بود. در روش پیشنهادی این معیار بهبود بسیار خوبی پیدا کرده و دلیل آن هم تکرار داده‌ها متناسب با محبوبیت داده و عدم مقاومت در برابر تکثیر داده‌ها است. در نمودار شکل (۴) مشاهده می‌شود که هر دو روش عملکرد خوبی در مقابل رخداد تغییر شلوغی در دوره‌ی ششم نشان می‌دهند.



شکل (٤) : نمودار طول مسیر مراجعه

٤-٣- هزینه سربار فضایی

این آزمایش سربار فضایی ایجاد شده‌ی الگوریتم تکرارداده را نشان می‌دهد. هر چه تعداد تکرارهای داده‌ها در کل سیستم بیشتر باشد سربار فضا افزایش می‌یابد. مطابق شکل (۵) روش پیشنهادی به دلیل این که مقاومت کمتری برای تکثیر داده‌ها نشان می‌دهد، سربار فضایی بیشتری دارد و همچنین در دوره‌ی هفتم به دلیل پدیده تغییر شلوغی که در دوره‌ی ششم اتفاق افتاده است، داده بیشتری تکثیر می‌کند و فضای بیشتری مصرف می‌شود.

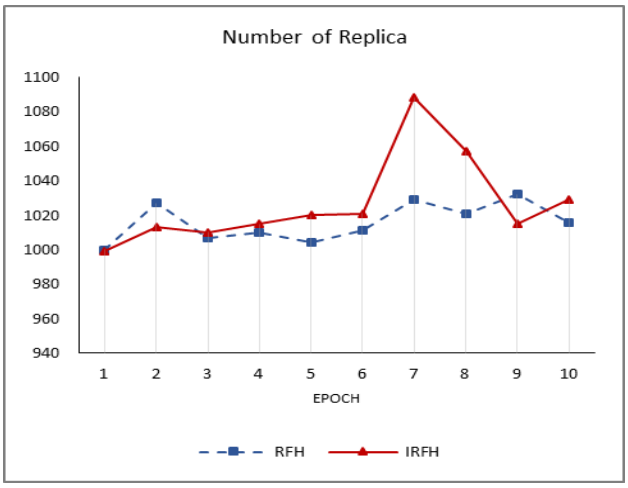
٤-٤- هزینه مصرف پهنای‌باند داخلی

از آنجا که الگوریتم پویا هست، هر دوره یک تکرارداده متناسب با وضعیت جاری سیستم لحاظ می‌کند. مکان‌دهی دوره‌ی جدید باعث می‌شود برخی داده‌ها در مکان‌های جدید تکرار شوند و برخی داده‌ها مهاجرت کنند که موجب مصرف پهنای‌باند می‌شود. این انتقال داده در هر دوره، پهنای‌باند مصرف می‌کند که تحت عنوان هزینه مصرف پهنای‌باند داخلی نلقی می‌شود. معیار مصرف پهنای‌باند داخلی سیستم، حجم داده‌ی جابه‌جاشده به منظور تکرار یا مهاجرت داده از مبدأ به مقصد برای اجرای دوره‌ی جدید تکرارداده است [6]. برای این آزمایش ترافیک انتقال‌یافته برای این منظور در دو روش مقایسه می‌شود. شکل (۶) ترافیک جابه‌جا شده بر حسب گیگابایت را نشان می‌دهد. مشاهده می‌شود که در دوره‌ی ششم پهنای‌باند زیادی مصرف شده است که به دلیل رخداد تغییر شلوغی است. در این دوره سیستم به منظور داشتن بهترین مکان‌دهی برای داده‌ها، برای مهاجرت و تکرار داده‌ها پهنای‌باند بیشتری مصرف می‌کند. با این وجود روش پیشنهادی در کل دوره آزمایش، به طور میانگین پهنای‌باند کمتری نسبت به روش RFH مصرف می‌کند.

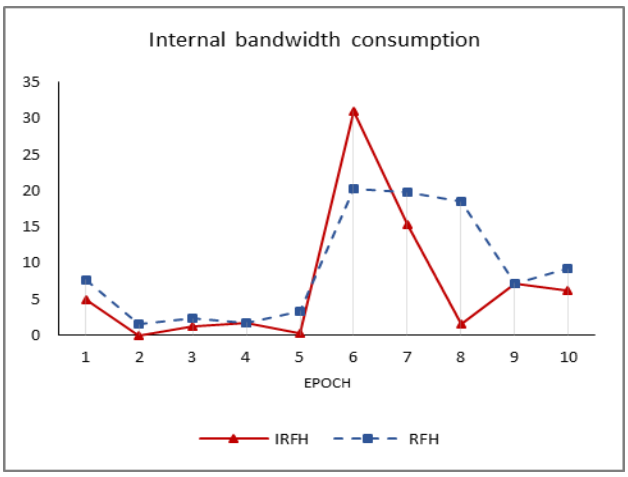
٤-٥- خرابی گره‌ها و بازیابی

هدف اصلی الگوریتم تکرارداده، تحمل خطا و خرابی‌ها و جلوگیری از دست دادن داده در خرابی گره‌ها است. به این منظور از ابتدا تا انتهای آزمایش ۱۰ درصد گره‌ها، با توزیع پواسون خراب می‌شوند و عکس العمل سیستم در تحمل این خرابی‌ها با پارامتر مهم دسترس‌پذیری داده نشان داده می‌شود.

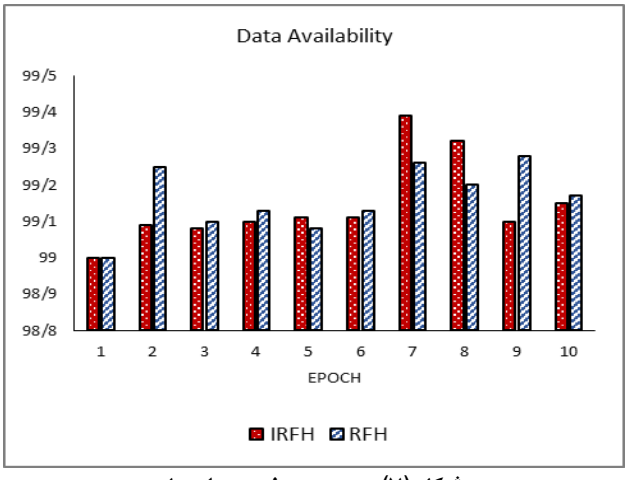
شکل (۷) نمودار دسترس‌پذیری داده را نشان می‌دهد که با وجود شکست‌های متوالی در طول دوره‌ی شبیه‌سازی همچنان در همه لحظات در هر دو روش دسترس‌پذیری داده بالاتر از حداقل دسترس‌پذیری تعیین‌شده (۹۹ درصد) است.



شکل (٥) : نمودار تعداد تکرارها



شکل (٦) : نمودار مصرف پهنای‌باند داخلی



شکل (٧) : دسترس‌پذیری داده‌ها

٤-٦- مقیاس‌پذیری

این روش برای سیستم‌های مقیاس‌بزرگ، مقیاس‌پذیر است. در آزمایشی با مقیاس ۱۰۰۰ گره در هر مرکزداده موفق به اجرا شده است. اما برای ارزیابی، سیستم با مقیاس کوچکتری پیکربندی شده است. به دلیل اینکه این مقاله با مقاله‌ی معرفی‌شده در [11] مقایسه شده است. لذا در مقیاس سیستم پیکربندی شده در این مقاله، پیکربندی شده است.

٥- نتیجه‌گیری

تکرارداده به هدف تحمل خطا برای سیستم‌های ذخیره‌سازی توزیع‌شده ارائه شده‌است. اما اهداف مهم دیگری هم مانند کاهش زمان انتظار کاربران، دسترسی سریع‌تر و کاهش مصرف پهنای‌باند داخلی، در تکرارداده دنبال می‌شوند. الگوی بارکاری در کاربردهای وب بسیار نامنظم است و یکی از ویژگی‌های آن پدیده‌ی تغییر شلوغی ناگهانی است که یک چالش اساسی به حساب می‌آید. روش RFH با رویکرد ترافیک‌گرا عملکرد مناسبی در مواجه با این پدیده ارائه داد. روش پیشنهادی به منظور بهبود کارایی این روش ضمن عملکرد مناسب در مقابل تغییر شلوغی ارائه شده‌است؛ توانست تا حد قابل قبولی این زمان پاسخ و مصرف پهنای‌باند را در قبال هزینه فضایی بیشتر نسبت به RHF، بهبود دهد. اما این مسأله همچنان باز است و نیاز به کارهای بیشتری در آینده، برای ارائه یک روش جامع، کارا و موثر تکرارداده احساس می‌شود.

مراجع

[1] Arokia Paul Rajan, R., Shanmugapriyaa, S., *"Evolution of Cloud Storage as Cloud Computing Infrastructure Service"*, IOSR Journal of Computer Engineering (IOSRJCE), Vol. 1, No. 1, pp. 38-45, May-June 2012.

[2] Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A., *"Energy-efficient data replication in cloud computing datacenters"*, Cluster Computing Journal, Vol. 18, No. 1, pp. 385-402, 2015.

[3] Debains, C., Alvarez-Tabio Togores, P., Karakusoglu, F., *"Reliability of Data-Intensive Distributed File System: A Simulation Approach"*, in proceedings of ACM conference, 2010.

[4] Feng, Q., Han, J., Gao, Y., Meng, D., *"Magicube: High Reliability and Low Redundancy Storage Architecture for Cloud Computing"*, in proceedings of IEEE Seventh International Conference on Networking, Architecture, and Storage, 2012.

[5] Ghemawat, S., Gobioff, H., Leung, S., *"The Google File System"*, in proceedings of 19th ACM Symposium on Operating Systems Principles(SOSP 2003), New York, USA, October, 2003.

[6] H uang, K., Li, D., Sun, Y., *"CRMS: a Centralized Replication Management Scheme for Cloud Storage System"*, in proceedings of IEEE/CIC International Conference on Communications in China (ICCC), pp. 344-348, October 2014.

[7] Janpet, J., Wen, YF., *"Reliable and Available Data Replication Planning for Cloud Storage"*, in proceedings of IEEE 27th International Conference on Advanced Information Networking and Applications, 2013.

[8] Kumar, K.A., Quamar, A., Deshpande, A., Khuller, S., *"SWORD: workload-aware data placement and replica selection for cloud data management systems"*, The VLDB Journal, Vol. 23, No. 6, pp. 845-870, 2014.

[9] Liao, B., Yu, J., Sun, H., Nian, M., *"A QoS-aware Dynamic Data Replica Deletion Strategy for Distributed Storage Systems under Cloud Computing Environments"*, in proceedings of Second International Conference on Cloud and Green Computing (CGC), pp. 219-225. November 2012.

[10] Liu, Y., Vlassov, V., *"Replication in Distributed Storage Systems: State of the Art, Possible Directions, and Open Issues"*, in proceedings of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 225-232, October 2013.

[11] Qu, Y., Xiong, N., *"RFH: A Resilient, Fault-Tolerant and High-efficient Replication Algorithm for Distributed Cloud Storage"*, in proceedings of 41st International Conference on Parallel Processing, 2012.

[12] Shvachko, K., Kuang, H., Radia, S., Chansler, R., *"The Hadoop Distributed File System"*, in proceedings of the 26th Symposium on Mass Storage Systems and Technologies, pp. 1-10, Incline Village, NV, USA, May, 2010,

[13] Sun, DW., Chang, GR., Gao, S., et al., *"Modeling a dynamic data replication strategy to increase system availability in cloud computing environments"*, Journal of Computer Science and Technology, Vol. 27, No. 2, pp. 256–272, March 2012.

[14] Wei, Y., Polytechnic, N., *"Auto-configurable, Reliable and Fault-tolerant Cloud Storage with Dynamic Parameterization"*, in proceedings of IEEE 10th World Congress on Services, 2014.

[15] Wei, Q., Veeravalli, B., Gong, B., Zeng, L., Feng, D., *"CDRM: A Cost-effective Dynamic Replication Management Scheme for Cloud Storage Cluster"*, in proceedings of IEEE International Conference on Cluster Computing (ICCC), 2010.

[16] Wu, J., Ping, L., Ge, X., Wang, Y., Fu, J., *"Cloud Storage as the Infrastructure of Cloud Computing"*, in proceedings of International Conference on Intelligent Computing and Cognitive Informatics (ICICCI), 2010.

[17] Xie, S., Cheng, Y., *"RAFR: A High Reliability Replication Algorithm for Cloud Storage"*, in proceedings of IEEE International Conference on Computer Science and Automation Engineering (CSAE), Vol. 2, pp. 378-381, 2012.

پانویس ها

[1] Data replication

[2] Cluster

[3] Google File System

[4] Hadoop Distributed File System

[5] Flash crowd

[6] Improved Resilient, Fault-Tolerant and High-efficient Replication

[7] Request oriented

[8] Namenode

[9] Replication Manager Component

[10] Datanode

[11] Lookup path length