پروژه درس معماری پیشرفته
(گرایش معماری کامپیوتر)

مروری بر تحقیقات گذشته

دانشجو:
ندا باغبان کاشانی
(۳۹۹۱۲۳۴۱۰۵۷۰۲۴)

خرداد ماه ۱۴۰۰

معرفی مقاله انتخاب شده

عنوان و مشخصات مقاله تحقیقاتی

Optimized data storage algorithm of IoT based on cloud omputing in distributed system

الگوریتم ذخیره‌سازی داده بهینه شده IoT براساس محاسبات ابری در سیستم توزیع‌شده

(Wang and Zhang, 2020)

(Wang and Zhang و همکاران، ۲۰۲۰)

Computer Communications

تعریف مسئله و هدف اصلی مقاله

- محاسبات ابری عمدتا مبتنی بر فن‌آوری اینترنت است و به کاربران خدمات، کاربردها و تعاملات ارائه می‌دهد.
- "ابر" استعاره‌ای برای اینترنت و شبکه است، که عمدتا یک بازنمایی انتزاعی از اینترنت و زیرساخت‌ها است.
- مزایای محاسبات ابری: مانند مقیاس فوق بزرگ، مجازی‌سازی، قابلیت اطمینان بالا، تطبیق پذیری، مقیاس‌پذیری بالا، خدمات مورد نیاز، هزینه پایین و غیره. علاوه بر این، محاسبات ابری می‌تواند علاوه بر خدمات محاسباتی، خدمات ذخیره‌سازی را نیز ارائه دهد.
- به‌عنوان یکی از خدمات بسیار ارائه‌شده توسط محاسبات ابری: ذخیره‌سازی ابری است که به دلیل کارایی بالا، انعطاف‌پذیری و سیستم پرداخت به‌جای شما، به‌طور گسترده‌ای مورد استقبال قرار گرفته و به کاربران اجازه می‌دهد تا داده‌ها را ذخیره و بر روی پلتفرم به اشتراک بگذارند.
- با این حال، داده‌های ذخیره‌شده در بستر ابر در یک دامنه غیر قابل کنترل قرار دارند و مالک داده‌ها (Data Owner-DO) کنترل داده‌ها را از دست می‌دهد، که خطر زیادی برای امنیت به همراه دارد.

تعریف مسئله و هدف اصلی مقاله

برای حل این مشکلات طرح‌های مختلفی توسط محققان ارائه شده که در ادامه به برخی از آن‌ها اشاره میکنیم:

- یک طرح رمزگذاری مبتنی بر اسناد (Attributed-Base Encryption, ABE) که از انتخاب برخی صفات و ویژگی‌های هویت کاربران یا همه صفات برای رمزگذاری و رمزگشایی داده‌ها استفاده می‌کند، توسط Sahai و Waters پیشنهاد شد که نمی‌تواند از حملات تبانی در میان چند کاربر جلوگیری کند.
- طرح رمزگذاری مبتنی بر ویژگی Key-Policy یا سیاست کلیدی (KP-ABE) که در آن مالک داده (DO) ساختار دسترسی را در کلید کاربر ایجاد می‌کند، و مجموعه‌ای از صفات و ویژگی‌ها برای انجام عملیات رمزگذاری بر روی داده‌ها مورد استفاده قرار گرفت، توسط Goyal و همکاران ارائه شد که در آن کنترل دسترسی بسیار دقیق و انعطاف‌پذیری در مدیریت حقوق کاربر را می‌توان به دست آورد، اما این راه‌حل، به‌روز رسانی تمام کلیدهای کاربران را به‌هنگام ابطال کلید نیاز دارد.
- Betten-court و همکاران ویژگی رمزگذاری مبتنی بر سیاست متن‌رمزی را پیشنهاد دادند. (CP-ABE)، که سربارهی محاسباتی آن در مقایسه با طرح KP-ABE بسیار بیشتر بود.
- براساس فن‌آوری استخراج داده و الگوریتم ژنتیک، Karimi و همکاران ترکیبی از خدمات آگاه از QoS (QoS-aware) را در محاسبات ابری پیشنهاد کردند که الزامات آن باید به صورت پویا پیاده‌سازی شوند، و نیاز به توازن بین بهترین وضعیت عملکرد و سرعت اجرای سبد خدمت دارد. با افزایش تعداد خدمات و گسترش فضای جستجو برای مشکلات، کارایی کافی برای روش‌های سنتی در ترکیب خدمات مورد نیاز در زمان معقول وجود نداشت.

تعریف مسئله و هدف اصلی مقاله

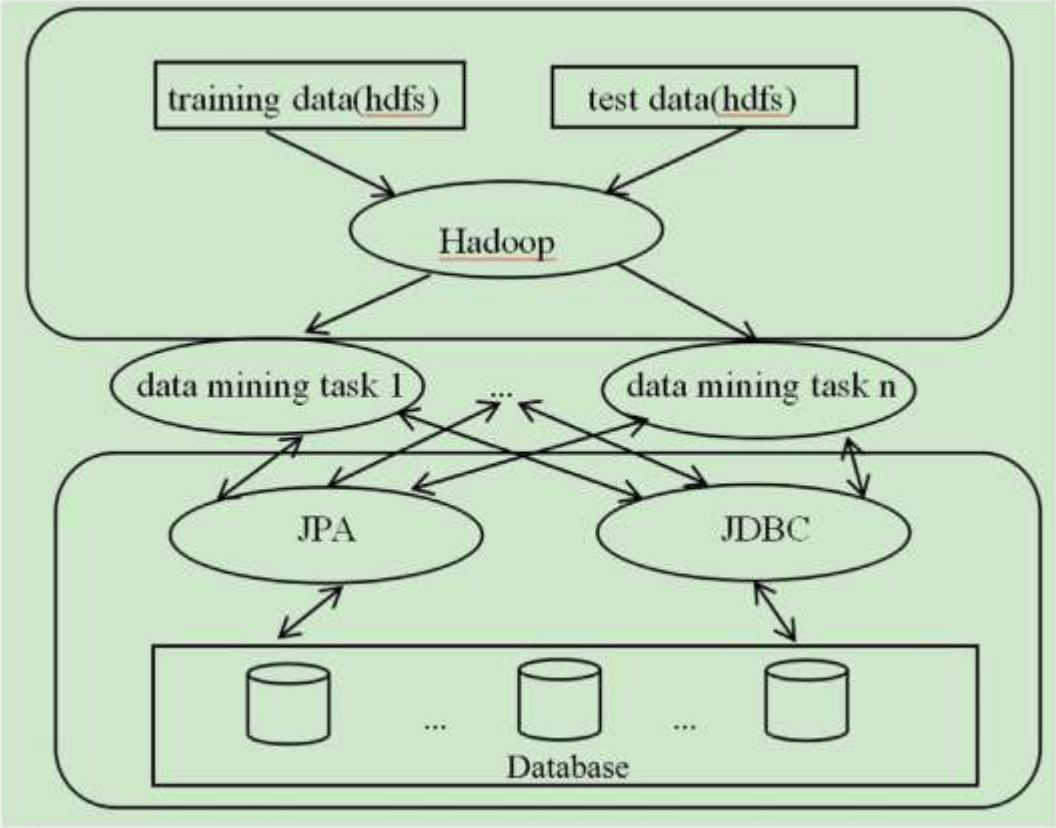
- امروزه، مدل محاسبات ابری متداول شامل مجموعه‌ای از ماشین‌های اختصاصی و گران‌قیمت است که خدمات محاسبات ابری را فراهم می‌کند و منجر به سرمایه‌گذاری‌های عظیم در هزینه‌های اولیه و هزینه‌های جاری می‌شود.
- این مقاله یک راه‌حل سیستم توزیع‌شده را اتخاذ می‌کند. همچنین می‌تواند به ابرهای ثابت و متحرک تقسیم شود.
- ابرهای ثابت از منابع محاسباتی استفاده می‌کنند که توسط ماشین‌های با اهداف عمومی کم‌تر مورد استفاده قرار می‌گیرند و ابرهای متحرک از منابع محاسباتی بلااستفاده دستگاه‌های موبایل استفاده می‌کنند.
- الگوریتم‌های ذخیره‌سازی دسترسی داده‌ها در محاسبات ابری دارای نقص کم در پردازش موثر داده‌ها و تحمل خطای کم هستند که نمی‌تواند نیازهای جامعه امروز را برای ذخیره‌سازی دسترسی داده‌ها برآورده کند.
- بنابراین، HDFS که به یک سیستم فایل توزیع‌شده اشاره دارد که تحمل خطا بالایی دارد و می‌تواند به ماشین‌های ارزان متصل شود، برای طراحی الگوریتم بهینه‌سازی ذخیره‌سازی دسترسی به داده‌ها در محاسبات ابری معرفی شده‌است.
- در همین حال، HDFS همچنین می‌تواند دسترسی داده‌های با توان عملیاتی بالا را فراهم کند که برای برنامه‌های ذخیره‌سازی اطلاعات دسترسی به داده‌ها ایده‌آل است.
- کاربرد HDFS تا حد زیادی می‌تواند کارایی پردازش داده و نرخ تحمل خطای الگوریتم بهینه‌سازی ذخیره دسترسی داده IoT را بهبود بخشد، که فن‌آوری‌های موثرتری را برای ذخیره اطلاعات دسترسی به داده IoT فراهم می‌کند.

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

ساختار استخراج داده در سیستم ذخیره‌سازی ابر بزرگ

- ایجاد یک مدل کلی ضرورت دارد.
- از طریق ساخت دسترسی به منابع در سیستم ذخیره‌سازی ابر در مقیاس بزرگ، تبادل اطلاعات و یکپارچه‌سازی منابع داده‌های چند منبعی محقق شده و خدمات کسب وکار چند بعدی و کنترل چند عملکردی نیز ارائه می‌شوند.
- سیستم ذخیره‌سازی ابر تحت سیستم دیجیتال سرویس اطلاعات توسط پرس و جوی داده‌های چند کاناله و داده‌های شبکه از طریق مکانیزم مدیریت حافظه نظارت می‌شود که می‌تواند کارایی و امنیت دسترسی و زمان‌بندی داده‌های شبکه را بهبود بخشد.
- به منظور ایجاد تعادل دینامیکی سیستم دسترسی QoS و خدمت در اطلاعات دیجیتال لایه MAC، مدل کلی چارچوب داده کاوی در پلتفرم هادوپ (Hadoop) در این مقاله ساخته شده‌است، که در شکل ۱ نشان‌داده شده‌است.

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:



شکل ۱- ساختار استخراج داده در پلتفرم Hadoop

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

طراحی الگوریتم بهینه‌سازی برای ذخیره‌سازی دسترسی به داده‌های IoT

- سیستم ذخیره‌سازی دسترسی به داده‌های IoT ویژگی‌های زیر را دارد:

(۱) اجرای عملی (Practical performance)

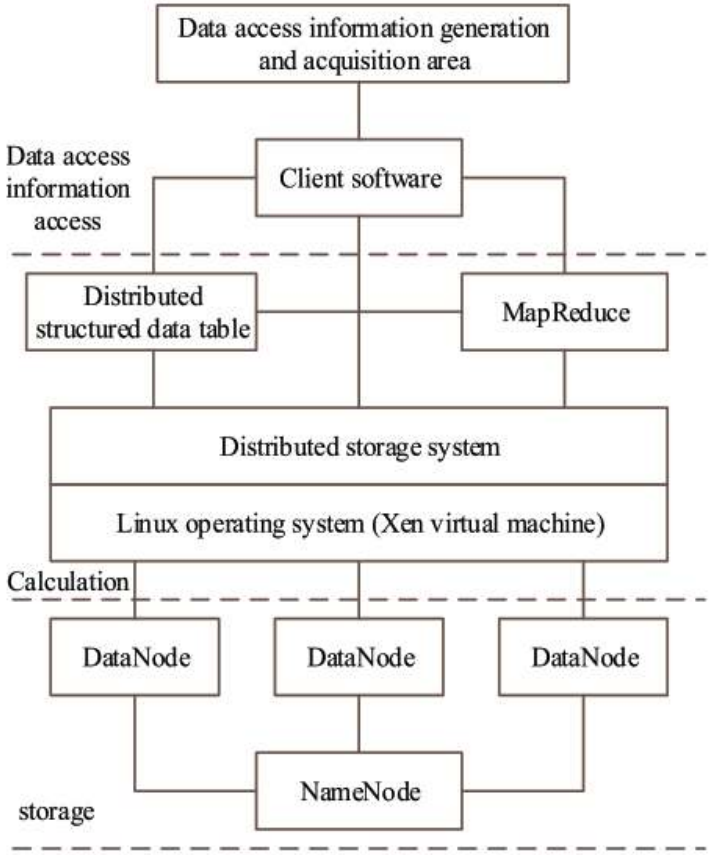
(۲) هزینه پایین (Low cost)

(۳) مقیاس پذیری (Scalability)

- همه موارد ذکر شده در بالا الزامات بالاتری را بر روی الگوریتم ذخیره‌سازی دسترسی به داده‌های IoT قرار داده‌اند. با این حال، الگوریتم های ذخیره دسترسی داده‌های موجود نمی‌توانند این الزامات را برآورده کنند، بنابراین آن‌ها باید بر این اساس بهینه‌سازی شوند. خلاصه فرآیند خاص آن به‌شرح زیر است:

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

بهینه‌سازی معماری ذخیره‌سازی دسترسی به داده



شکل ۲- نمودار بهینه‌سازی معماری ذخیره‌سازی دسترسی به داده

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

بهینه‌سازی استراتژی در توزیع ذخیره‌سازی دسترسی به داده‌ها

- پس از دسترسی به اطلاعات دسترسی داده، می توان آن را به چندین بلوک داده تقسیم کرد و براساس قوانین خاص بر روی گره‌های IoT توزیع کرد. از طریق تحقیق مشخص شد که عوامل اصلی موثر بر توزیع ذخیره‌سازی دسترسی داده‌ها به طور عمده به سه دسته تقسیم می‌شوند:

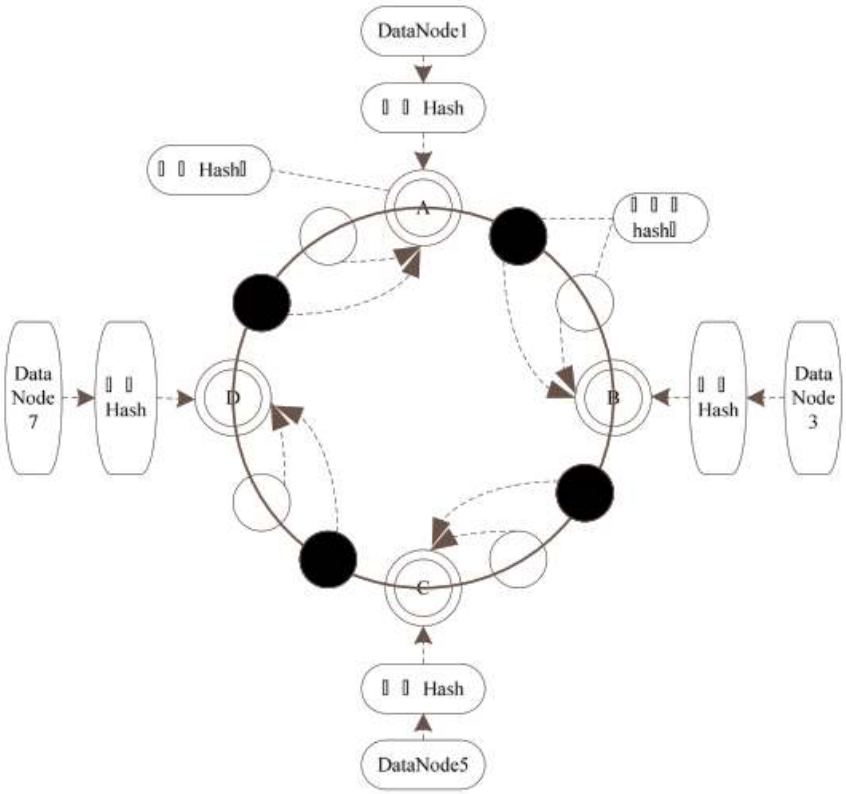
(۱) تعادل بار (Load balancing)

(۲) خطای گره (Node failure)

(۳) اجرای عملیات ذخیره‌سازی (Storage operation performance)

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

پیکربندی مقدار Hash برای دسترسی داده‌ها به موقعیت ذخیره اطلاعات

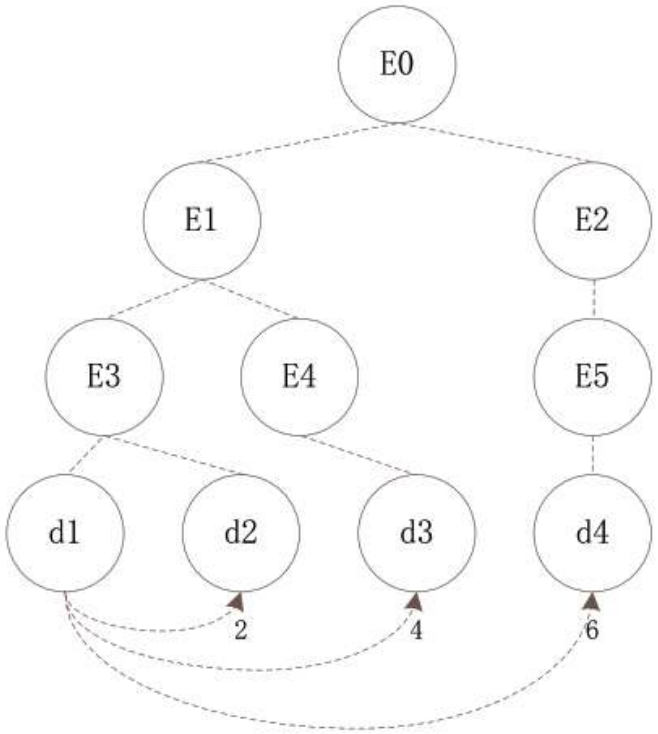


شکل ۳- نمودار شماتیک الگوریتم Hash

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

بهینه‌سازی توپولوژی شبکه IoT

توپولوژی‌های شبکه الگوریتم ذخیره‌سازی دسترسی به داده‌های IoT موجود مشکلات عمده‌ای دارند. بنابراین، توپولوژی شبکه برنامه‌ریز برای بهینه‌سازی استفاده می‌شود.



شکل ۴- نمودار بهینه‌سازی توپولوژی شبکه IoT

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

بهینه‌سازی اندازه بلوک داده

- به منظور بزرگ‌تر و سریع‌تر کردن ذخیره‌سازی دسترسی داده‌ها، الگوریتم اثر برای بهینه‌سازی اندازه بلوک داده استفاده می‌شود. فرمول بهینه‌سازی برای اندازه بلوک داده عبارت است از:

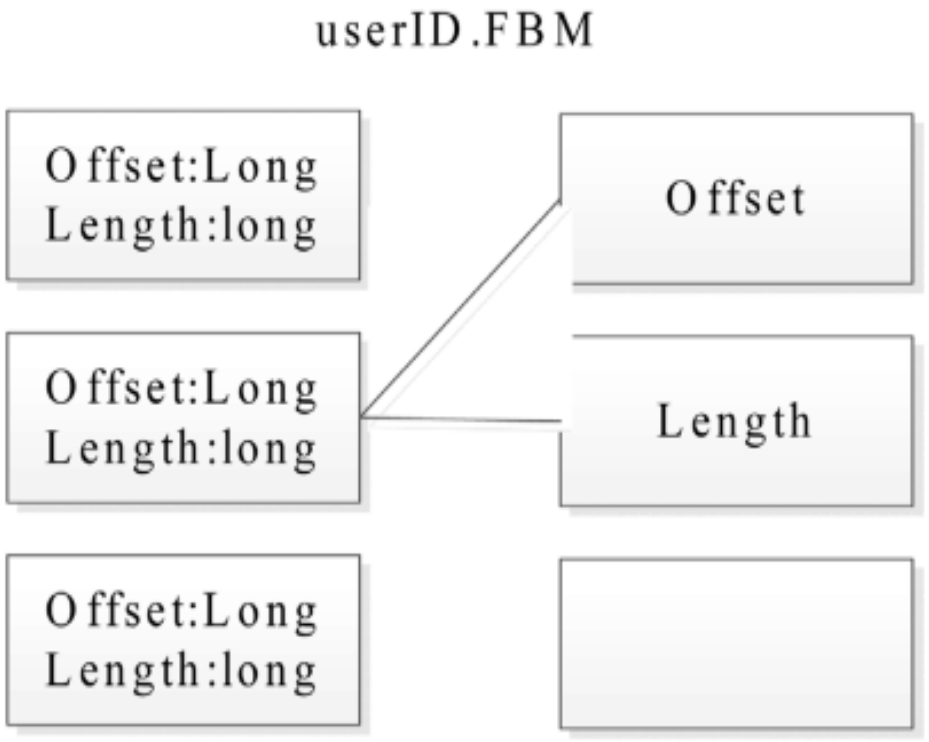
$$effect(\gamma) = f(x) \propto \frac{trans\_time}{trans\_time - seek\_time} \qquad (6)$$

که در آن transe_time دسترسی به اطلاعات دسترسی به داده‌ها و seek_time زمان پیکربندی محل ذخیره اطلاعات دسترسی به داده‌ها است.

- با توجه به فرمول بالا، اندازه بهینه بلوک داده به دست می‌آید و اطلاعات دسترسی داده به طور متناظر براساس آن ذخیره می‌شوند.
- معماری اصلی HDFS و حالت مدیریت بلوک برای معرفی متا سرور اطلاعات File-Block-Mapping (FBM) برای ذخیره‌سازی فضای کاربر IoT نگهداری می‌شوند.
- علاوه بر این، FBM اندازه فایل و مقدار offset فایل کاربر ذخیره‌شده در HDFS را ثبت می‌کند، و هر کاربر روی HDFS فایل کاربری خود را دارد و تمام فایل‌های خود را ذخیره می‌کند.
- علاوه بر آن، هر نام فایل مربوط به ID کاربر است، که رابطه نگاشت آن در شکل ۵ و بهینه‌سازی فایل در شکل ۶ نشان‌داده شده‌است.

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

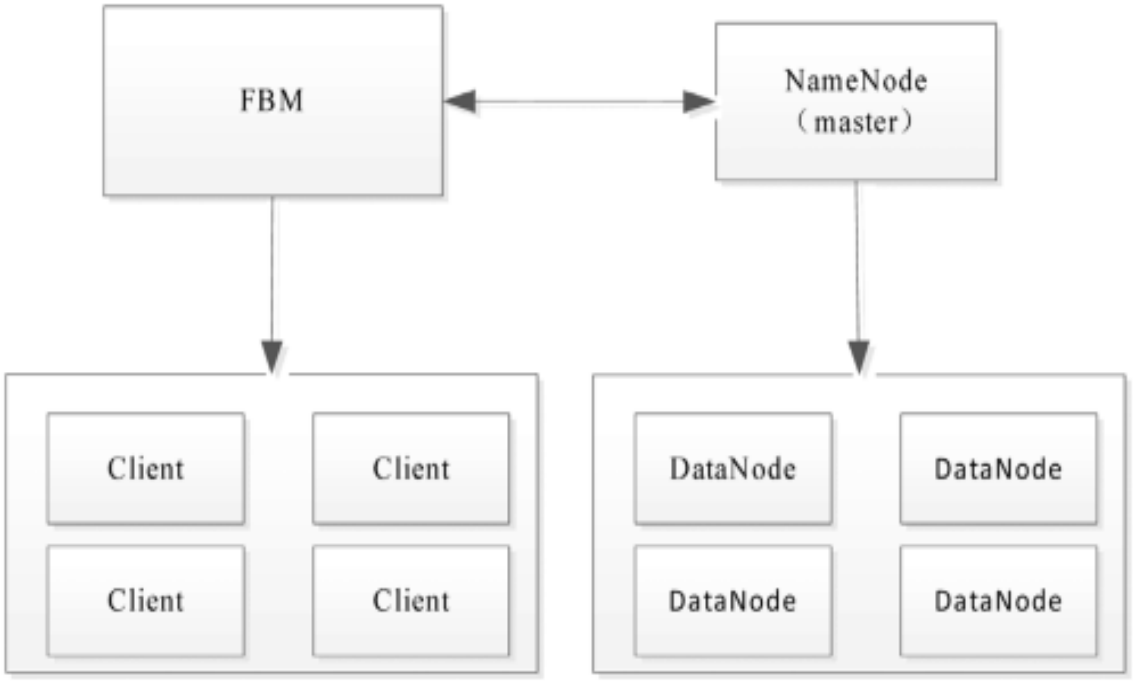
بهینه‌سازی اندازه بلوک داده



شکل ۵- ارتباط نگاشت (Mapping) بین فایل‌های FBM و فایل‌های کاربر

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

بهینه‌سازی اندازه بلوک داده

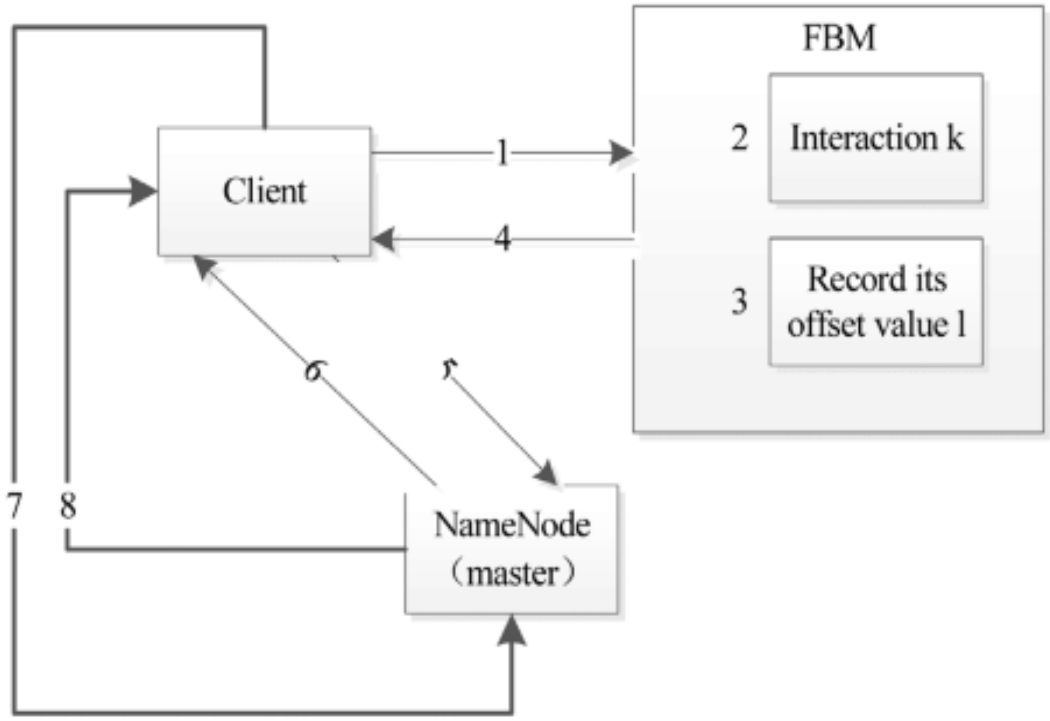


شکل ۶- ارتباط نگاشت (Mapping) بین فایل‌های FBM و فایل‌های کاربر

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

فرآیند خواندن و نوشتن برنامه بهینه‌سازی فایل

(۱) فرآیند نوشتن فایل در شکل ۷ نشان‌داده شده است.

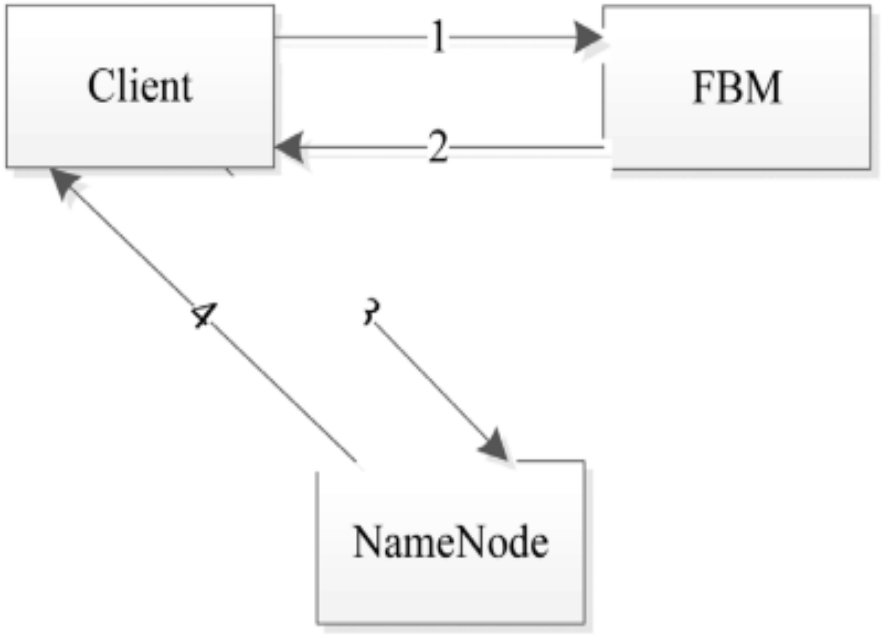


شکل ۷- نمودار فرآیند نوشتن فایل

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

فرآیند خواندن و نوشتن برنامه بهینه‌سازی فایل

(۲) فرآیند خواندن فایل کاربر در شکل ۸ نشان‌داده شده است.



شکل ۸- خواندن فایل‌ها

توضیح راه‌حل پیشنهادی مقاله برای حل مساله:

اجرای ذخیره‌سازی بهینه فایل

- به منظور بهینه‌سازی ذخیره‌سازی فایل‌های کوچک، کلاس SmallFileStatus به اطلاعات فایل کوچک متا-داده براساس HDFS اصلی معرفی می‌شود، که مشخصه‌های آفست و اندازه فایل‌های کوچک را در فایل‌های کاربر براساس FileStatus اضافه می‌کند.
- علاوه بر این، کلاس اصلی طراحی بهینه‌سازی ذخیره‌سازی فایل کوچک، کلاس FBM است، ویژگی‌های اصلی آن نقشه فایل است که متعلق به نوع HashMap است. علاوه بر این، جفت مقدار-کلید نام فایلی است که کلید آن فایل کوچکی است، و مقدار آن اطلاعات متا داده است.
- در این مقاله HDFS بر روی یک ماشین فیزیکی ساخته شده، و یک سناریو آزمایش شبیه‌سازی در OPNET Modeler انجام شده که به ۳ گره سرور IoT مجهز شده و برای انجام آزمایش‌ها به تجهیزات ارتباطی متصل شده است.
- نتایج آزمایش‌های انجام شده در مقاله نشان می‌دهد که داده‌های اینترنت اشیا بهینه شده توسط الگوریتم پیشنهادی در این مقاله از نظر سرعت انتقال، اشغال منابع سیستم و زمان پاسخ نسبت به حالت اصلی برتری دارد. علاوه بر این، حداکثر بازده پردازش داده‌های اینترنت اشیا از انتقال بهینه شده می‌تواند به ۹۹٪ و حداکثر میزان تحمل خطا از الگوریتم بهینه سازی شده می تواند به ۹۶.۱۲٪ برسد.

نقاط قوت و ضعف مقاله

مشکلات ناشی از راه‌حل بهینه‌سازی فایل و مزیت راه‌حل ارائه شده در مقاله

- هنگامی که سیستم روشن می‌شود، FBM اطلاعات فایل متنی هر کاربر و ورودی‌های شاخص فایل کوچک را بارگذاری می‌کند.
- سپس، سیستم اطلاعات متا داده کاربر را در حافظه بارگذاری می‌کند تا مشتری بتواند اطلاعات فایل را در مسیر فایل قرار دهد.
- عیب آن این است که اطلاعات فراداده بیش از حد در حافظه بارگذاری می‌شوند، که ممکن است منجر به سربار حافظه FBM بیش از حد بالا شود.
- برای حل مشکل ذخیره‌سازی مداوم FBM، پایگاه‌داده (database) رابطه‌ای برای ذخیره اطلاعات مرتبط متا داده در فضای پایگاه داده اتخاذ می‌شود، که مزایای به‌کارگیری آسان و کاهش مصرف حافظه FBM را دارد.
- هنگامی که کاربران جدید اینترنت اشیا ثبت‌نام می‌کنند، فایل‌های کاربری خودشان را خواهند داشت. فایل‌های کاربر به آرامی در یک سطح خاص جمع می‌شوند و تبدیل به فایل‌های بزرگ می‌شوند.
- بنابراین، در یک خوشه HDFS، زمانی که گره NameNode تنها دارای اطلاعات متا داده یک کاربر (userID.file) است، مصرف حافظه NameNode به‌میزان زیادی کاهش خواهد یافت.

جمع‌بندی و پیشنهادات برای کارهای آتی

- اینترنت اشیا (IoT) موجود از الگوریتم‌های ذخیره‌سازی دسترسی به داده محاسبات ابری استفاده می‌کند، در واقع یک الگوریتم Hash (Hash Algorithm) است که الگوریتم درهم سازی دارای نقص‌های بهره‌وری پردازش داده کم و نرخ تحمل خطای کم است. بنابراین، HDFS برای بهینه‌سازی الگوریتم‌های ذخیره‌سازی دسترسی به داده محاسبه ابری معرفی می‌شود.
- HDFS ابتدا برای بهینه‌سازی معماری ذخیره‌سازی دسترسی به داده‌ها با توجه به مشکلات معماری ذخیره‌سازی دسترسی به داده‌ها در اینترنت اشیا استفاده می‌شود.
- سپس توپولوژی IoT بهینه می‌شود و اندازه بلوک داده نیز با الگوریتم اثر بهینه می‌شود. در نهایت، طراحی ذخیره‌سازی فایل، بهینه شده است. از طریق آزمایش‌های شبیه‌سازی، ثابت شده است که روش ذخیره ابر بهینه‌شده دارای مزایای عملکرد واضحی در سرعت خواندن و نوشتن فایل و همچنین استفاده از حافظه است.

- در این مقاله به‌منظور به‌دست آوردن داده‌ها و نتایج تجربی دقیق‌تر، تحقیقات بیشتر در زمینه بهینه‌سازی الگوریتم بهینه ذخیره-سازی اطلاعات IoT در رایانش و محاسبات ابری پیشنهاد شده است.

شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

در این مقاله یک آزمایش شبیه‌سازی به منظور مقایسه عملکرد الگوریتم بهینه‌سازی ذخیره‌سازی دسترسی داده‌ها در رایانش و محاسبات ابری طراحی شده‌است که عمدتا نشان‌دهنده عملکرد الگوریتم توسط کارایی پردازش داده‌ها و تحمل خطا می‌باشد.

۱- پیکربندی محیط

جدول ۱- پیکربندی سیستم آزمایشی

	Configuration	Type
IoT server	CPU	Intel Xeon W-3175X 28-core 3.1G
	RAM	32GDDR4*4
IoT node	CPU	Intel i7-9700K Core Eight Core 3.6G
	RAM	16GDDR4*2
	hard disk	Seagate Galaxy Exos 7E88 TB 256 MB 7200 RPM

شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

به منظور ارزیابی عملکرد روش افزودن فایل‌های بهینه‌سازی FBM به IoT، ۶ مجموعه آزمایش مقایسه‌ای بر روی HDFS بهبود یافته و HDFS اصلی انجام می‌شود. پارامترهای مورد آزمایش در جدول ۲ نشان‌داده شده‌اند.

۲- تحلیل نتیجه

جدول ۲- پارامترهای به‌کار برده شده در آزمایش

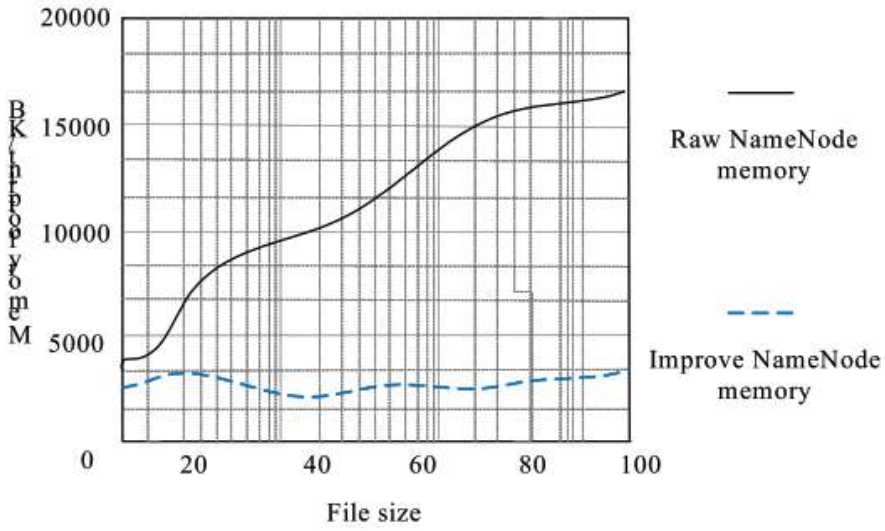
User scale	Number of files created	File size (w)
1000	100	10
2000		20
3000		40
4000		60
5000		80
6000		100

شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

اندازه حافظه فرآیند NameNode در طول آزمایش ثبت می‌شود، و نتایج آزمایش در شکل ۹ نشان داده شده است، که در آن محور افقی اندازه فایل و محور قائم مختصات اندازه حافظه NameNode را نشان می‌دهد که واحد آن KB است.

۲-۱- مقایسه حافظه فرآیند NameNode

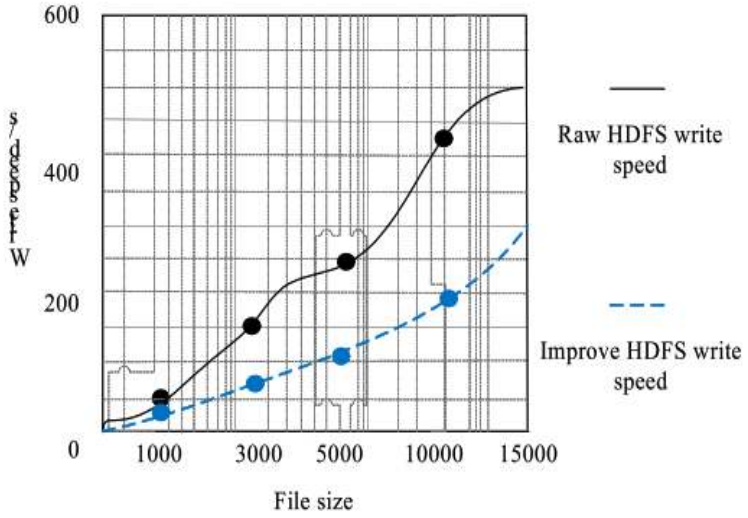


شکل ۹- مقایسه حافظه NameNode قبل و بعد از بهینه‌سازی فایل

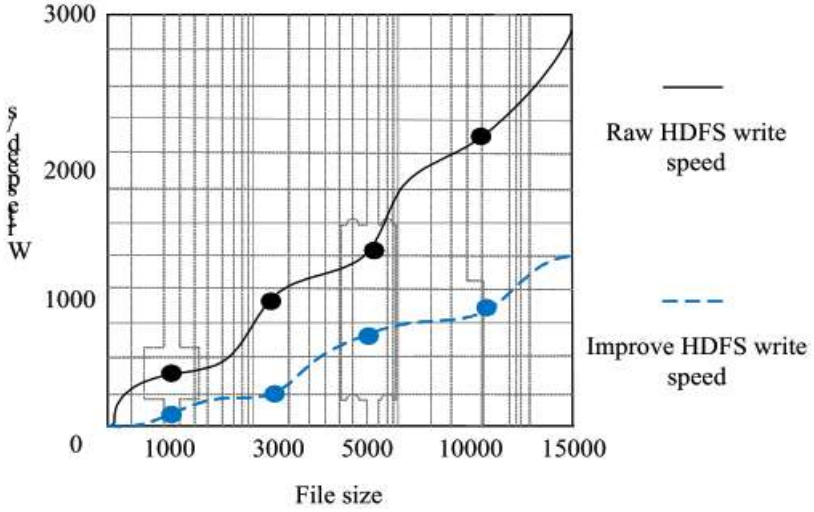
شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

۲-۲- مقایسه کارایی آپلود فایل



شکل ۱۰- مقایسه زمان صرف شده برای ساختن یک فایل (بدون نوشتن داده)

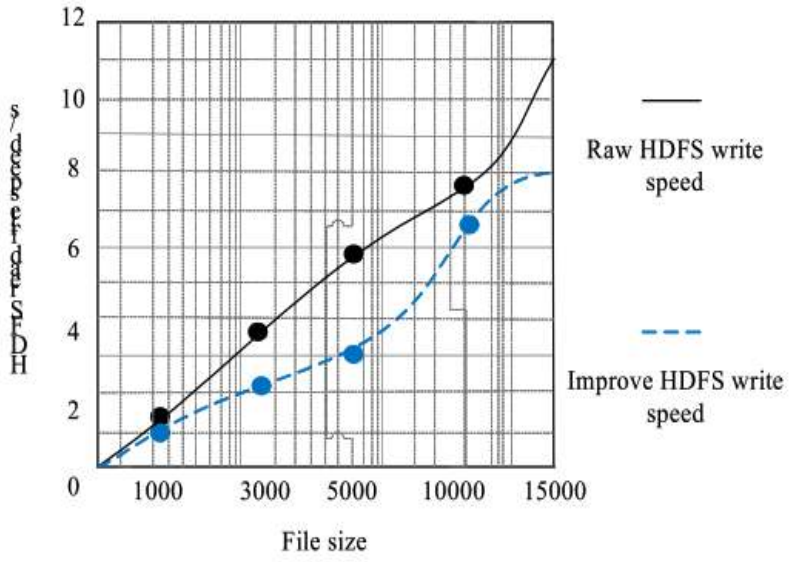


شکل ۱۱- مقایسه زمان صرف شده برای ساختن یک فایل (با نوشتن داده)

شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

۲-۳- مقایسه کارایی دانلود فایل

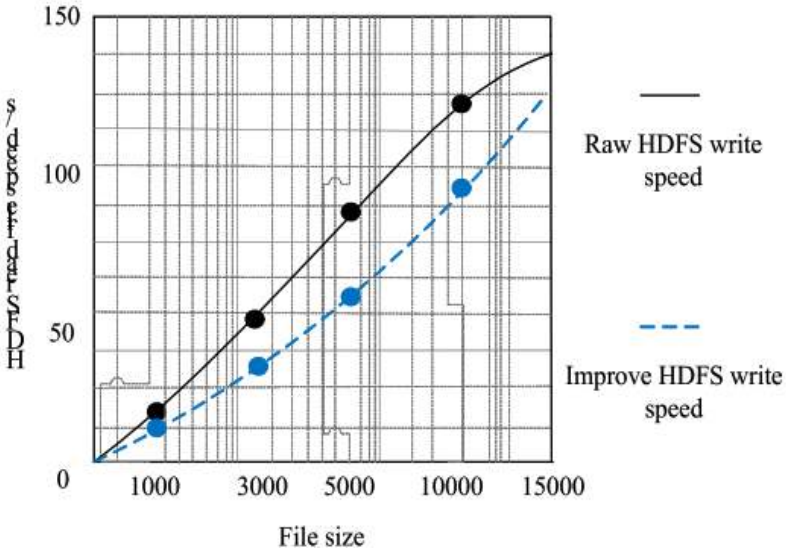


شکل ۱۲- مقایسه زمان صرف شده برای خواندن فایل‌ها
(بدون دانلود داده)

شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

۴-۲- تجزیه و تحلیل مقایسه‌ای بر روی کارایی پردازش داده‌ها



شکل ۱۳- مقایسه زمان صرف شده برای خواندن فایل‌ها (با دانلود داده)

جدول ۳- مقایسه کارایی پردازش داده‌ها

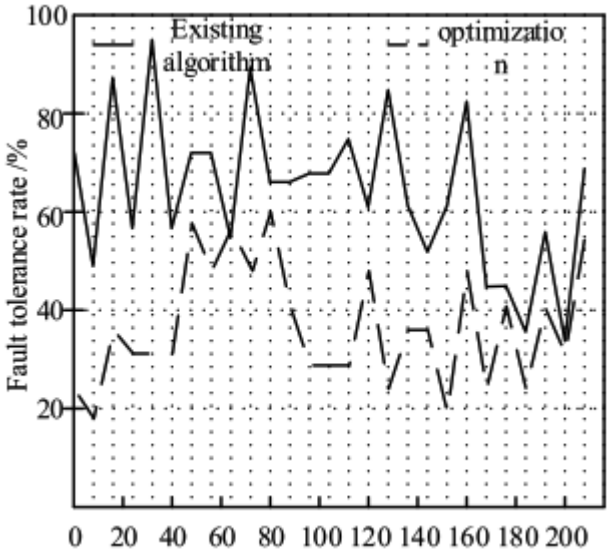
Experiment times	Existing algorithm	Optimization algorithm
20	56%	89%
40	64%	88%
60	50%	79%
80	49%	76%
100	67%	85%
120	73%	81%
140	62%	91%
130	53%	90%
180	69%	99%
200	61%	93%

شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

۲-۵- تجزیه و تحلیل مقایسه‌ای در تحمل خطا

همچنانکه در شکل ۱۴ مشاهده می‌شود تحمل خطا در الگوریتم موجود از ۱۹ ٪ تا ۶۰ ٪ متغیر است و الگوریتم بهینه دارای حداقل تحمل خطا ۳۴ ٪ و حداکثر ۹۶,۱۲ ٪ می‌باشد. بنابراین، تحمل خطا در الگوریتم بهینه بسیار بیشتر از الگوریتم های موجود است.



شکل ۱۴- مقایسه تحمل پذیری خطا

شبیه سازی

تست عملکرد الگوریتم بهینه‌سازی دسترسی به داده‌های ذخیره ابری IoT:

با توجه به نتایج تجربی ذکر شده در بالا، الگوریتم بهینه‌سازی ذخیره‌سازی دسترسی داده‌های IoT در محاسبات ابری پیشنهادی در این مقاله تا حد زیادی بازدهی پردازش داده‌ها و تحمل خطا را بهبود می‌بخشد، که به طور کامل نشان می‌دهد که الگوریتم بهینه‌سازی ذخیره‌سازی دسترسی داده‌های IoT در محاسبات ابری پیشنهادی در این مقاله عملکرد بهتری دارد.

با سپاس فراوان از

توجه شما