# مسائل مربوط به اجرای یک پلت فرم محاسبات ابر


## چکیده

محاسبات ابر یک توسعه سیستم مبتنی بر اینترنت است که در آن منابع محاسباتی با مقیاس بزرگ پذیر به عنوان یک سرویس بر روی اینترنت برای کاربران ارائه شده اند. مفهوم محاسبات ابری شامل زیرساخت های وب ، نرم افزار به عنوان خدمات (SaaS) ، وب ۲,۰ و دیگر فن آوری های در حال ظهور می شود ، و بیشتر و بیشتر از جامعه صنعت و پژوهش جلب توجه نموده است. در این مقاله، ما تجربه و درس های آموخته شده در ساخت و ساز از پلت فرم محاسبات ابر را توضیح می دهیم. به طور خاص ، ما سیستم فایل سازگار GFS را با اندازه Chunk متغیر به منظور تسهیل پردازش گسترده داده ها طراحی می کنیم، و برخی از ارتقائات پیاده سازی بر روی MapReduce را به منظور بهبود توان عملیاتی سیستم معرفی می کنیم. همچنین در مورد برخی از مسائل عملی برای پیاده سازی سیستم بحث می کنیم. در انجمن آرشیو چین وب (وب سایت InfoMall) که ما از سال ۲۰۰۱ (در حال حاضر شامل بیش از سه میلیارد صفحات وب چینی) انباشته نموده ایم, این مقاله تلاش ما برای پیاده سازی یک پلت فرم برای دامنه سرویس محاسبات ابر خاص را با معدن نمودن وب متنی با مقیاس بزرگ به عنوان نرم افزار مورد هدف قرار داده ارائه می دهد. و امیدوارانه محققان در کنار تلاش های ما از ابر بهره مند خواهند شد هنگامی که آماده شود.


## ۱ مقدمه

همانطور که جنبه های بیشتری از کار و زندگی شخصی آنلاین حرکت می کند و اینترنت پلت فرمی برای جامعه انسان های مجازی شده است، الگویی جدید در مقیاس بزرگ محاسبات توزیع شده بوجود آمده است. شرکت های مبتنی بر وب ، مانند گوگل و آمازون ، ، زیرساخت های وب را برای مقابله با اینترنت در مقیاس ذخیره سازی داده ها و محاسبات ساخته اند. اگر ما چنین زیرساختی را به عنوان "کامپیوتر مجازی" در نظر بگیریم، این مورد امکان مدل محاسباتی جدید را نشان می دهد, به عنوان مثال ، داده ها و محاسبات بر روی کامپیوتر "فوق العاده" با ذخیره

سازی بی سابقه و محاسبات متمرکز، توانایی ها، که می تواند به عنوان ساده ترین شکل از محاسبات ابری به آن نگاه شود.

به طور کلی ، مفهوم محاسبات ابری می تواند شامل فن آوری های کامپیوتری مختلف ، از جمله ترکیب زیرساخت های وب ، وب ۲،۰ و بسیاری از فن آوری های دیگر در حال ظهور باشد. مردم ممکن است دارای دیدگاه های مختلف از بینش های مختلف باشند. به عنوان مثال، از نظر کاربر نهایی ، خدمات محاسبات ابر خدمات نرم افزار نرم افزار سیستم عامل را از دسکتاپ به سمت ابر حرکت می دهد که باعث می شود کاربران در هر زمان از هر کجا قادر به پلاگین باشند و از ذخیره سازی در مقیاس بزرگ و منابع محاسباتی استفاده نمایند. از سوی دیگر ، ارائه دهنده خدمات محاسبات ابری ممکن است روی این مورد تمرکز نماید که چگونه توزیع و برنامه ریزی منابع کامپیوتر صورت می گیرد. با این حال ، ذخیره سازی و محاسبات بر روی داده های عظیم از فن آوری های کلیدی در زیرساخت های ابر محاسبات است.

گوگل فن آوری های زیرساخت های آن را برای محاسبات ابری در سال های اخیر توسعه داده است، از جمله سیستم فایل گوگل (GFS) [8] MapReduce [7] و .[6] Bigtable. GFS یک سیستم فایل توزیع مقیاس پذیر است، که بر تحمل خطا تاکید می کند زیرا برای به اجرا در آوردن اقتصادی مقیاس پذیر طراحی شده است، اما به ناچار سخت افزار کالای غیر قابل اعتماد (به دلیل به مقیاس محض خود) است، و خدمات با کارایی بالا را برای تعداد زیادی از مشتریان ارائه می دهد. Bigtable سیستم ذخیره سازی توزیع شده بر اساس GFS برای مدیریت داده های ساخت یافته است. این سیستم فراهم کننده چکیده نگاشت سه بعدی به برنامه ها است، و با موفقیت در بسیاری از محصولات مستقر گوگل به کار گرفته شده است. MapReduce یک مدل برنامه نویسی با پیاده سازی مربوط به پردازش داده های عظیم است. MapReduce یک چکیده را با تعریف "نگاشت کننده" و "کاهنده" فراهم می کند. "نگاشت کننده" برای هر جفت ورودی کلید / مقدار برای تولید تعداد دلخواه جفت کلید / مقدار میانی اعمال می شود. "کاهنده" برای به تمام مقادیر مرتبط با همان کلید میانی برای تولید جفت خروجی کلید / مقدار اعمال می شود. MapReduce یک مدل برنامه نویسی آسان برای استفاده است، و دارای توانایی بیان کافی

برای پشتیبانی از الگوریتم های بسیاری از دنیای واقعی و این وظایف است. سیستم MapReduce می تواند با داده های ورودی پارتیشن بندی شود، اجرای برنامه در سراسر مجموعه برنامه ماشینها را زمانبندی نماید، وقوع خرابی های ماشین را هدایت نماید ، و ارتباط بین ماشینی را مدیریت نماید.

به تازگی ، بسیاری از سیستم های مشابه توسعه یافته اند. 3] KosmosFS] منبع باز مانند GFS - سیستم ، است که از رابط سخت POSIX پشتیبانی می نماید. Hadoop [2] پروژه منبع باز جاوا فعال است. با پشتیبانی از یاهو ، Hadoop پیشرفت زیادی را در این دو سال به دست آورد. این مورد در سیستمی بزرگ با ۴۰۰۰ گره مستقر شده است و در بسیاری از وظایف پردازش داده های مقیاس بزرگ مورد استفاده قرار گرفته است.

در اکتبر ۲۰۰۷ ، گوگل و آی بی ام برنامه " ابتکار ابر رایانه " را برای دانشگاه ها در ترویج آموزش و پژوهش مربوط به کار بر روی محاسبات در مقیاس بزرگ به طور فزاینده مردمی راه اندازی نمودند. بعدها در ماه ژوئیه ۲۰۰۸ ، HP ، اینتل و یاهو یک ابتکار عمل مشابه را برای ترویج و توسعه ابر محاسبات تحقیقات و آموزش و پرورش راه اندازی نمودند.. چنین پروژه ابرمحاسباتی نه تنها می تواند محاسبات موازی آموزش و پرورش را بهبود دهد، بلکه کار پژوهش مانند مدیریت داده ها در مقیاس اینترنت ، پردازش و محاسبات علمی را ترویج می دهد. با الهام از این روند و با انگیزه نیاز به ارتقاء کار موجود، ما به پیاده سازی زیرساخت های عملی وب به عنوان پلت فرم ابر محاسبات پرداختیم که می تواند برای ذخیره داده ها در مقیاس وسیع وب و ارائه قابلیت پردازش عملکرد بالا مورد استفاده قرار گیرد. در دهه گذشته ، تمرکز تحقیقات ما و توسعه سیستم روی جستجو در وب ، وب هاست (هاستینگ) بوده است و ما دو سیستم وب سایت های عمومی، به عنوان مثال موتور جستجوی Tianwang [۴] و وب infomall سایت سیستم بایگانی [1] را و نگهداری توسعه داده ایم, همانطور که در شکل ۱ نشان داده شده است.

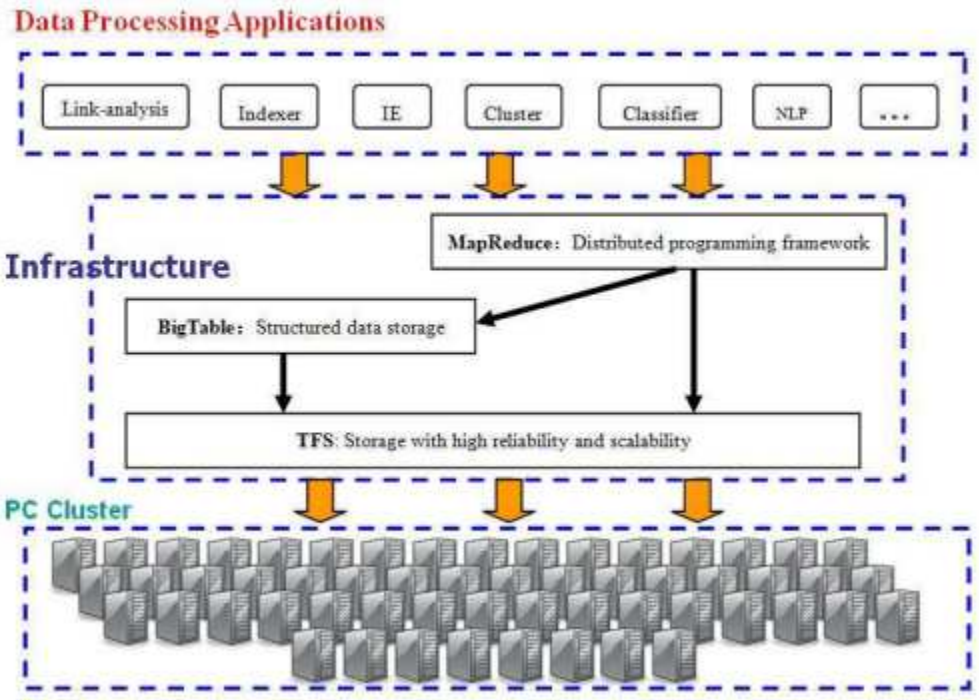

(a) Tianwang (b) Web infomall

شکل ۱ : موتور جستجو و بایگانی وب Chines توسعه یافته در گروه SEWM PKU

در طی این مدت ، ما بیش از ۵۰ داده وب TB را جمع نمودیم، یک خوشه PC شامل ۱۰۰ + رایانه های شخصی را ساختیم و نرم افزار های وب سایت های مختلف نرم افزار از قبیل تجزیه و تحلیل و پردازش متن صفحه وب را طراحی نمودیم. با افزایش حجم داده ها و حجم کار محاسبات در سیستم ، ما ابر فن آوری محاسبات را یافتیم که رویکردی نویدبخش برای بهبود مقیاس پذیری و بهره وری از سیستم برای ارائه خدمات وب است. از سال ۲۰۰۷ ، ما شروع به طراحی و توسعه وب سیستم زیرساخت ها خود، به نام "Tplatform" ، شامل سیستم فایل مانند GFS "TFS" [10] و محیط محاسبات MapReduce نمودیم. ما اعتقاد داریم که عمل ما از پیاده سازی پلت فرم ابر محاسبات می تواند مرجع خوبی برای محققان و مهندسین باشد که به این زمینه علاقه مند هستند.

## ۲ TPlatform : پلت فرم ابر محاسبات

در این بخش ، ما به طور خلاصه پیاده سازی و اجزای پلت فرم ابر محاسبات خود به نام "Tplatform" را معرفی می کنیم. ما برای اولین بار مروری بر سیستم را پس از پیاده سازی سیستم دقیق و برخی از مسائل عملی ارائه می دهیم.




شکل ۲ : چارچوب سیستم Tplatform

شکل ۲ نشان دهنده چارچوب کلی سیستم "Tplatform" است، که متشکل از سه لایه، به عنوان مثال ، خوشه PC، زیرساخت ها برای پلت فرم ابر محاسبات و پردازش داده لایه کاربردی است. لایه خوشه PC سخت افزار و ابزارهای ذخیره سازی برای پردازش داده ها در مقیاس بزرگ را فراهم می کند. لایه کاربرد, خدمات برای کاربران را فراهم می کند که در آن کاربران می توانند برنامه های کاربردی خود، از جمله وب سایت تجزیه و تحلیل داده ها ، پردازش زبان ، خوشه و طبقه بندی ، و غیره را توسعه دهند. لایه دوم تمرکز اصلی کار ما است، که متشکل از TFS سیستم فایل توزیع, مکانیسم های ذخیره سازی داده های توزیع شده BigTable ، و مدل برنامه نویسی MapReduce است. پیاده سازی BigTable مشابه روش ارائه شده در [۶] است، و از این رو ما بحث مفصل را در اینجا حذف می کنیم.

### ۲,۱ پیاده سازی سیستم فایل

سیستم فایل از اجزای مهم این سیستم برای پشتیبانی از ذخیره سازی و مدیریت اطلاعات گسترده است. طراحی TFS مقیاس پذیر، سیستم فایل توزیع شده ، و هر خوشه TFS متشکل از یک مرع مجرد و سرورهای متعدد Chunk ای است و می تواند توسط مشتریان متعدد قابل دسترسی باشد.

### ۲,۱,۱ معماری TFS

در TFS ، فایل ها به Chunk هایی با اندازه متغیر تقسیم شده اند. هر Chunk توسط Chunk تغییر ناپذیر ۶۴ بیتی منحصر به فرد در سطح کلی و اختصاص داده شده توسط مستر در زمان خلق Chunk شناخته می شود. سرورهای Chunk ذخیره Chunk را روی دیسک های محلی و داده های Chunk خواندن / نوشتن مشخص شده توسط هدایت Chunk و دامنه بایت ذخیره می کنند. برای قابلیت اطمینان داده ، هر Chunk بر روی سرورهای چند Chunk تکرار شده است. به طور پیش فرض ، ما سه کپی را در سیستم حفظ می کنیم، هر چند کاربران می توانند سطوح تکرار متفاوت را برای فایل های مختلف تعیین نمایند.

Master ابرداده ها را از سیستم فایل نگهداری می کند، که شامل فضای نامی ، کنترل دسترسی به اطلاعات ، نگاشت فایل ها به Chunk ها ، و مکان فعلی از Chunk ها می شود. همچنین فعالیت های سیستم مانند جمع آوری زباله از Chunk های تنها، و مهاجرت Chunk بین سرورها Chunk را کنترل می کند. هر سرور Chunk به صورت دوره ای با Master در پیام HeartBeat برای گزارش حالت خود ارتباط برقرار می نماید و دستورالعمل را بازیابی می نماید.

ماژول مشتری TFS که با هر نرم افزار با یکپارچه سازی فایل سیستم API مرتبط است، که می تواند هراه با Master و سرورهای chunk برای خواندن و یا نوشتن داده ها به نمایندگی از برنامه ارتباط برقرار نماید. مشتریان با Master برای انجام عملیات های ابرداده ارتباط برقرار می نمایند، اما تمام ارتباطات تحمل داده ها به طور مستقیم به سرورهای chunk می رود.

این سیستم برای به حداقل رساندن دخالت Master در عملیات دسترسی به فایل طراحی شده است. ما API POSIX را فراهم نمی کنیم. علاوه بر فراهم نمودن عملیات خواندن و نوشتن عادی ، مانند GFS ، ما نیز سابقه

اتمی عمل اضافه کردن را ارائه می کنیم به طوری که چند سرویس گیرنده به صورت همزمان می تواند به یک فایل بدون هماهنگی اضافی در میان آنها متصل شوند. در پیاده سازی سیستم ، مشاهده می کنیم که عملیات اضافه کردن رکورد, عملیات کلیدی برای عملکرد سیستم است. ما مکانیزم تعامل سیستم را طراحی می کنیم که متفاوت از GFS و دارای بازدهی بهتر در عملکرد اضافه کردن رکورد است.

### ۲,۱,۲ حجم Chunk متغیر

در GFS ، یک فایل به Chunk های ثابت (به عنوان مثال ، ۶۴ MB) تقسیم می شود. هنگامی که یک مشتری از عملیات اضافه کردن ثبت برای الحاق داده استفاده ، سیستم چک می کندکه آیا اضافه کردن رکورد به آخرین Chunk از یک فایل خاص ممکن است Chunk سرریز را ایجاد نماید، به عنوان مثال، از اندازه حداکثر تجاوز نماید. اگر چنین باشد، تمام المثنی Chunk را در حداکثر اندازه پوشش می دهد، و به مشتری اطلاع می دهد که این عملکرد باید بر روی Chunk جدید ادامه یابد. (اضافه کردن رکورد اغلب محدود به یک چهارم اندازه Chunk برای حفظ بدترین حالت را در سطح قابل قبول می شود)/ در صورت نوشتن وقوع خرابی، این رویکرد ممکن است منجر به تکرار سوابق و ثبت های ناقص شود.

در طرح TFS ما ، Chunk های یک فایل مجاز به داشتن اندازه متغیر هستند. با مکانیزم تعامل سیستم پیشنهادی، این استراتژی باعث عملیات اضافه کردن رکورد کارآمد تر می شود. داده های عمل توسعه ، سابقه قطعات و دوبرابر نمودن ثبت در سیستم ما ضروری نیست. اگرچه این رویکرد برخی از هزینه های زیادی را به همراه می آورد به عنوان مثال ، هر ساختار داده Chunk نیاز به یک ویژگی اندازه Chunk دارد، عملکرد کلی میزان قابل توجهی بهبود می یابد، زمانی که عملیات خواندن و اضافه کردن ثبت عملیات مسلط در سیستم ما هستند و می تواند از این انتخاب طراحی بهره ببرند.

### ۲,۱,۳ عملیات فایل

در حال حاضر ما عملیات فایل های مختلف را برای TFS, از قبیل خواندن, ثبت و نوشتن را طراحی نموده ایم. از آنجا که ما اندازه های Chunk متغیر را در TFS فراهم نموده ایم، استراتژی عملیات متفاوت از GFS است. در اینجا ما در حال حاضر جزئیات اجرای عملیات خواندن را برای نشان دادن تفاوت رویکرد خود ارائه می دهیم.

برای خواندن یک فایل ، تبادل پیام مشتری با Master ، موقعیت Chunk هایی را می گیرد که می خواهد از آنها بخواند، و سپس با سرور Chunk برای بازیابی داده ها ارتباط برقرار نماید. از آنجا که GFS از اندازه Chunk های ثابت استفاده می کند، مشتری فقط نیاز به ترجمه نام فایل و افست بایت به شاخص Chunk در داخل فایل دارد و Master یک درخواست حاوی نام فایل و شاخص Chunk را می فرستد. Master با هدایت Chunk مربوطه و محل کپی ها پاسخ می دهد. سپس مشتری یک درخواست برای یکی از کپی ها، به احتمال زیاد نزدیکترین آنها را می فرستد. این درخواست هدایت Chunk و یک محدوده بایت در آن Chunk را مشخص می کند. خواندن های بیشتر از یک Chunk نیاز به تعامل بیشتر تعامل Master کاربر ندارد, مگر آن که اطلاعات منقضی شود یا فایل بازگشایی شود.

در سیستم TFS ما ، داستان با توجه به استراتژی Chunk اندازه متغیر متفاوت است. مشتری نمی تواند افست بایت را به طور مستقیم به شاخص Chunk ترجمه نماید. باید تمام اندازه های Chunk را در فایل بداند قبل از تصمیم گیری که Chunk باید خوانده شود. راه حل آن بسیار ساده است ، هنگامی که یک مشتری ، یک فایل را با استفاده از حالت خوانده شده باز می کند، تمام Chunk های اطلاعات را از Master می گیرد، از جمله هدایت Chunk ، اندازه Chunk و مکان ، و از اینها برای دریافت اطلاعات برای Chunk مناسب استفاده می کند. اگر چه این استراتژی این است با واقعیت از اندازه Chunk متغیر تعیین می شود، مزیت آن این است که مشتری فقط نیاز به برقراری ارتباط با Master برای یک بار به خواندن کل فایل دارد، که بسیار کارآمد تر از طرح اصلی GFS است.اشکال آن این است که هنگامی که یک مشتری یک فایل را برای خواندن باز کرده باشد, داده های بعداً اضافه شده توسط مشتریان دیگر برای این مشتری نامرئی است. اما ما اعتقاد داریم که این مشکل نیز قابل اغماض است، زمانی که معمولا اکثر فایل ها در برنامه های کاربردی وب ایجاد و اضافه می شوند، و خواندن توسط کاربردهای

پردازش داده ها چند بار بدون تغییر صورت می گیرد. اگر در هر موقعیتی این مشکل بحرانی شود، به آسانی می تواند بر مجموعه ای از برچسب های زمانی منقضی برای اطلاعات Chunk غلبه و آنرا هنگامی که نامعتبر است تازه نماید.

TFS نشان دهنده تلاش های ما برای ساخت زیرساخت های پردازش داده ها برای مقیاس بزرگ است. اگر چه سیستم ما دارای فرضیات مشابه و معماری به صورت GFS است، تفاوت کلیدی این است که اندازه Chunk متغیر است، که باعث می شود سیستم ما قادر به اتخاذ تعامل سیستم های مختلف برای عملیات اضافه کردن رکورد باشد. عملیات اضافه کردن رکورد ما بر اساس سطح Chunk و در نتیجه عملکرد اضافه کردن رکورد جمع شده دیگر محدود به پهنای باند شبکه سرور های Chunk نمی شود که آخرین Chunk فایل را ذخیره می کنند. ارزیابی تجربی ما نشان می دهد که رویکرد ما به طور قابل توجهی عملکرد اضافه کردن رکورد همزمان برای فایل تنها را تا ۲۵ ٪ بهبود می بخشد. نتایج بیشتر در TFS گزارش شده است[۱۰]. ما باور داریم طراحی می تواند برای دیگر زیرساخت های پردازش داده های مشابه اعمال شود.

### ۲,۲ پیاده سازی MapReduce

سیستم MapReduce یکی دیگر از بخش های عمده از پلت فرم ابر محاسبات است ، و اخیرا بیشتر توجه [۹ ، ۷ ، ۱۱] را به خود جذب کرده است. معماری اجرای ما مشابه با [2] Hadoop است که ساختار Master کارگر معمولی است. سه نقش در این سیستم وجود دارد : Master ، کارگر و کاربر. Master کنترل کننده مرکزی از سیستم است که مسئول پارتیشن بندی داده ها ، برنامه ریزی وظیفه ، حفظ تعادل بار و پردازش تحمل خطا است. کارگران, وظایف مشخص از پردازش داده ها و محاسبات را اجرا می کنند. بسیاری از کارگران در این سیستم وجود دارد که وظایف را از Master واکشی می نمایند، وظایف را اجرا می نمایند و برای انتقال داده ها ارتباط با یکدیگر برقرار می نمایند. کاربر, مشتری سیستم است، Map را پیاده سازی می کند و توابع برای محاسبات کار را کاهش می دهد و جریان محاسبات را کنترل می نماید.

### ۲,۲,۱ اجرای بهبود

ما سه توسعه را برای بهبود عملکرد theMapReduce در سیستم خود ساخته ایم. اول ، ما با انتقال میانه داده به عنوان یک کار مستقل برخورد می کنیم. هر وظیفه محاسبات شامل نگاشت و کاهش وظایف فرعی می شود. در پیاده سازی نمونه مانند Hadoop ، وظیفه کاهش, انتقال داده های میانی را شروع می کند که داده ها را از تمام ماشین های انجام دهنده وظایف نگاشت واکشی می نماید. این یک ارتباط غیر قابل کنترل به تمام است که ممکن است تراکم شبکه را ایجاد نماید، و از این رو به تنزل کارایی سیستم منجر شود. در طراحی خود، ما وظیفه انتقال را از کاهش وظیفه تقسیم می کنیم، و " ماژول انتقال داده ها " را برای اجرا و زمانبندی وظیفه انتقال داده ها به طور مستقل پیشنهاد می کنیم. با استفاده مناسب از الگوریتم زمان بندی، این روش می تواند احتمال تراکم شبکه را کاهش دهد. هر چند این رویکرد ممکن است حجم کار Master را تشدید نماید زمانی که تعداد وظایف انتقال زیاد است, این مشکل می تواند با تنظیم دانه دانه از وظیفه انتقال و وظایف یکپارچه سازی انتقال داده ها با همین منبع و پرداختن به هدف اختصاص داده شود. در عمل، رویکرد جدید ما می تواند به میزان قابل توجهی عملکرد انتقال داده را بهبود بخشد.

دوم، برنامه ریزی وظیفه یکی دیگر از نگرانی های بر روی سیستم MapReduce است، که کمک می کند تا تعهد منابع بین انواع وظایف و برنامه برای دستور انجام وظیفه صورت گیرد. برای بهینه سازی استفاده از منابع سیستم، ما چند سطح بازخورد صف الگوریتم زمان بندی را در طراحی خود اتخاذ می کنیم. چند صف برای تخصیص وظایف همزمان مورد استفاده قرار می گیرند، و هر یک از آنها را با یک اولویت خاص اختصاص داده می شوند، که ممکن است برای وظایف مختلف با توجه به منابع درخواستی متفاوت باشد. الگوریتم ما به صورت پویا می تواند اولویت کار در حال اجرا را تنظیم نماید که بین حجم کار سیستم ایجاد بالانس می نماید و توان کلی را بهبود می بخشد.

بهبود سوم سریال سازی داده است. در چارچوب MapReduce ، یک وظیفه محاسبات شامل چهار مرحله می شود: نگاشت ، پارتیشن ، گروه و کاهش. داده ها در عملیات نگاشت خوانده می شوند, داده های میانی ایجاد می شوند و در این سیستم منتقل می شوندو در نهایت نتایج به دست آمده توسط عملیات کاهش صادر می شوند. تبادل

مکرر داده ها بین حافظه و دیسک وجود دارد که به طور کلی توسط سریال سازی داده انجام می شود. در اجرای سیستم MapReduce، ما مشاهده کردیم که نوع داده ساده و بومی غالبا در بسیاری از پردازش برنامه های کاربردی داده ها استفاده می شود. از آنجایی که بافر حافظه به طور گسترده استفاده می شود ، بیشتر داده ها قبلاً در حافظه می ماند قبل از اینکه آنها سریال سازی شی داده ای جدید غیرفعال شود. به عبارت دیگر ، ما باید از عملیات غیرفعال نمودن سریال سازی گران اجتناب نماییم که حجم زیادی از فضای حافظه را مصرف می کند و به تنزل کارایی سیستم منتج می شود. برای حل این مشکل ، نوع داده را برای کلید و مقدار به عنوان اشاره گر خالی (void) تعریف می کنیم. اگر ما بخواهیم غیرفعال نمودن سریال سازی داده ها با نوع داده های بومی صورت گیرد، عملیات تخصیص اشاره گر ساده می تواند جایگزین عملیات غیرفعال نمودن سریال سازی، شود که بسیار کارآمد تر هست. با استفاده از این بهینه سازی، ما همچنین می توانیم داده ها را مستقیما در حافظه با غیرفعال نمودن سریال سازی داده ها ترتیب بندی نماییم. این مکانیزم به طور قابل توجهی می تواند عملکرد MapReduce را بهبود بخشد هر چند برخی از سربارهای هزینه را برای مدیریت بافر معرفی می نماید.

### ۲,۲,۲ ارزیابی عملکرد در MapReduce

با توجه به فقدان معیار که می تواند برنامه های کاربردی معمولی را ارائه دهد، ارزیابی عملکرد بر روی سیستم MapReduce نشاندهنده یک کار بی اهمیت نیست. ما برای اولین بار PennySort را به عنوان معیار ساده استفاده می کنیم. نتیجه نشان می دهد که عملکرد انتقال داده های میانی در فاز ترکیب تنگنای سیستم است که در واقع به ما برای بهینه سازی ماژول انتقال داده ها در MapReduce انگیزه می دهد. علاوه بر این ، ما نیز یک برنامه واقعی را برای معدن متن بررسی می کنیم، که آمار فرکانس کلمه چینی در صفحات وب را جمع می کند. ما برنامه را در مجموعه وب سایت چینی GB۲۰۰ اجرا می کنیم. تابع نگاشت به تجزیه و تحلیل محتوای صفحه وب می پردازد و هر کلمه منفرد چینی را به عنوان مقدار کلیدی تولید می کنیم. تابع کاهش همه مقادیر جمع شده را جمع می کند و فرکانس را صادر می کند. در بستر ازمایش ما با ۱۸ گره ، کار را به وظایف نگاشت ۳۳۸۵ ، ۳۰ وظیفه

کاهش وظایفی و ۱۰۱۵۵۰ وظایف انتقال داده تقسیم می کنیم, کل کار را با موفقیت در حدود ۱۰ ساعت کامل شد ، که بسیار موثر است.

### ۲,۳ مسائل عملی برای پیاده سازی سیستم

ذخیره سازی داده ها و قابلیت محاسبات از عوامل عمده پلت فرم ابر محاسبات است که تعیین می کند که چگونه زیرساخت ها به خوبی می تواند خدمات را به کاربران نهایی ارائه دهند. ما با برخی از مشکلات فنی و مهندسی در طول پیاده سازی سیستم برخورد می کنیم. در اینجا در مورد برخی از مسائل عملی در کار خود بحث می کنیم.

#### ۲,۳,۱ معیارهای طراحی سیستم

در طراحی سیستم ، هدف ما برای توسعه یک سیستم است که مقیاس پذیر ، قدرتمند ، با عملکرد بالا و آسان برای حفظ می باشد. با این حال ، برخی از مسائل طراحی سیستم ممکن است متناقض باشند و که ما را در یک معضل در بسیاری از موارد قرار دهد. به طور کلی ، ما سه معیار اصلی را برای طراحی سیستم در نظر می گیریم: ۱) برای یک راه حل خاص، تنگنای این روش چیست که ممکن است عملکرد سیستم را منحط نماید؟ ۲) کدام راه حل دارای مقیاس پذیری و انعطاف پذیری بهتر برای تغییر آینده است؟ ۳) از آنجا که پهنای باند شبکه از منابع کمیاب از سیستم است، چگونه به طور کامل منابع شبکه را در پیاده سازی استفاده می کنیم؟ در زیر ، ما مثالی برای نشان دادن ملاحظات ما در پیاده سازی ارائه می نماییم.

در سیستم MapReduce ، تحمل خطا را می توان توسط هر دو مقطع Master یا کارگران انجام داد. Master نقش کنترل کننده کلی را می گیرد، اطلاعات را از کل سیستم حفظ می کند و می تواند به راحتی تصمیم بگیرد که آیا شکست خورده وظیفه باید عمل دوباره اجرا شود ، و کی و کجا دوباره اجرا شود. کارگران تنها اطلاعات محلی را حفظ می کنند و مسئول گزارش وضعیت وظایف در حال اجرا برای Master هستند. طرح ما ترکیبی از مزایای استفاده از این دو عامل است. کارگران می تواند یک کار ناموفق را برای تعداد معینی از دفعات، دوباره اجرا کنند و حتی مجاز به گذر از برخی از سوابق بد داده ها هستند که باعث خرابی می شود. این استراتژی توزیع شده است

قوی تر و مقیاس پذیرتر از از ساز و کار متمرکز است، یعنی ، فقط برنامه ریزی دوباره وظایف شکست خورده در سمت Master.

### ۲,۳,۲ پیاده سازی ارتباطات بین ماشین

از آنجا که اجرای پلت فرم ابر محاسبات بر اساس خوشه PC است، چگونگی طراحی پروتکل های ارتباطی بین ماشین، موضوع کلیدی برنامه نویسی در محیط توزیع شده است. روش از راه دور تماس با (RPC) کالای میانه یک الگوی مردمی برای اجرای مدل کاربر-سرور از محاسبات توزیع شده است که فناوری های ارتباطی درون فرایندی است که به یک برنامه کامپیوتری اجازه می دهد تا سبب یک زیرروال یا روش برای اجرا بر روی یک کامپیوتر دیگر در یک خوشه PC بدون برنامه نویسی صریح جزئیات برای این تعامل از راه دور شود. در سیستم ما ، همه خدمات و پروتکل های ضربان-قلب تماس های RPC هستند. ما از موتور ارتباطات اینترنت (ICE) بهره برداری می کنیم، که یک میان افزار شی گرا است که RPC ، شی گرا را برای پیاده سازی چارچوب RPC فراهم می کند. رویکرد ما به خوبی در زیر سیستم اجرا می شود و  و می تواند از مدل ارتباطی ناهمگام حمایت نماید. عملکرد شبکه ارتباطات برای ما سیستم با ICE است با پروتکل های ناهمزمان ویژه با برنامه نویسی سوکت قابل مقایسه است، که اجرای آن خیلی پیچیده است.

### ۲,۳,۳ عیب یابی و اشکال زدایی سیستم

عیب یابی و اشکال زدایی سیستم در محیط توزیع یک چالش بزرگ برای محققان و مهندسان است. سیستم کلی متشکل از مراحل مختلف توزیع شده در شبکه است و این پروسه ها با یکدیگر برای اجرای یک کار پیچیده ارتباط برقرار می کنند. به علت برقراری ارتباطات همزمان چنین در سیستمی، بسیاری از خطاها به طور کلی آسان موقعیت یابی نمی شوند، و از این رو به سختی می تواند اشکال زدایی نمود. بنابراین ، ما ارتباط سیستم کامل را در سیستم خود ضبط می کنیم. در طرف سرور و کاربر ، مرزهای مهم نرم افزار مانند واسط های API و RPC همه وصل می شوند. به عنوان مثال ، ورود برای پیام RPC برای بررسی انسجام پروتکل می تواند مورد استفاده قرار گیرد ، ورود

برای انتقال داده ها می تواند برای اثبات صحت انتقال مورد استفاده قرار گیرد. علاوه بر این ، ورود عملکرد برای تنظیم عملکرد را ضبط می کنیم. در سیستم Mapeduce ما ، اتصال در سمت سرویس گیرنده, جزئیات اطلاعات برای داده ها در زمان خواندن و در زمان نوشتن را برای تمام وظایف، هزینه زمان عملیات مرتب سازی در وظیهف کاهش را که عوامل طراحی سیستم ما را تنظیم می کند ضبط می کند.

در کار ما ، ورود ثبت شده به سیستم نه تنها به ما در تشخیص مشکلات برنامه ها کمک می کند، بلکه کمک می کند تنگنا عملکرد سیستم یافت شود و از این رو ما می توانیم پیاده سازی سیستم را بهبود بخشیم. با این حال, اشکال زدایی و تشخیص توزیع شده, هنوز دارای کارایی کم است و نیاز به کار زیاد دارد. ما انتظار ابزار بهتر و روش ها برای بهبود اثربخشی و بهره وری از اشکال زدایی و تشخیص در مقیاس بزرگ سیستم توزیع شده پیاده سازی را داریم.

## ۳ نتیجه گیری

بر اساس تجربه ما با Tplatform ، ما مسائل مختلف عملی در اجرای پلت فرم ابر محاسبات را به دنبال مدل گوگل مورد بحث قرار دادیم. مشاهده شده است که در حالی که GFS / MapReduce / BigTable ارائه یک چارچوب مفهومی برای هسته اصلی نرم افزار از یک ابر را می دهد و Hadoop مخفف محبوب ترین پیاده سازی منبع آزاد است, هنوز هم بسیاری از مسائل مربوط به اجرای مورد نظر برای بررسی وجود دارد. سه تا از آنها در این مقاله مشخص شده است.

- اندازه chunk برای یک فایل در GFS می تواند به جای ثابت متغیر باشد. با اجرای دقیق, این تصمیم طراحی, عملکرد بهتر برای خواندن و الحاق عملیات ارائه می گردد.
- انتقال داده در میان گره های مشارکتی در مرحله کاهش می تواند "قابل برنامه ریزی" باشد به جای "غیر قابل کنترل" بودن. مکانیزم جدید فراهم کننده فرصتی برای جلوگیری از تراکم های شبکه است که عملکرد را تنزل می دهد.

- داده ها با انواع بومی نیز می تواند به طور موثر برای دسترسی به داده ها در نگاشت و توابع کاهش مرتب شوند، که احتمالا باعث بهبود عملکرد در برخی موارد می شود.

در حالی که Tplatform به عنوان یک کل هنوز هم در حال پیشرفت است، پیاده سازی BigTable, بخش های ادامه یافته (TFS و MapReduce) و در حال حاضر مفید می باشد. برنامه های کاربردی متعدد, امکان سنجی و مزایای استفاده از روش های پیاده سازی جدید ما را نشان داده اند. کد منبع از Tplatform از [۵] در دسترس است.

## قدردانی

این کار توسط CB310902۲۰۰۷ شماره ۹۷۳ پروژه ، آی بی ام ۲۰۰۸ SUR گرانت برای PKU ، و ملی بنیاد علوم طبیعی چین تحت گرانت No.60603045 و ۶۰۸۷۳۰۶۳ مورد حمایت قرار گرفت.

## References

[1] *China Web InfoMall*. http://www.infomall.cn, 2008.

[2] *The Hadoop Project*. http://hadoop.apache.org/, 2008.

[3] *The KosmosFS Project*. http://kosmosfs.sourceforge.net/, 2008.

[4] *Tianwang Search*. http://e.pku.edu.cn, 2008.

[5] *Source Code of Tplatform Implementation*. http://net.pku.edu.cn/~ webg/tplatform, 2009.

[6] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: a distributed storage system for structured data. In *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 15–15, 2006.

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI '04: Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation*, pages 137–150, 2004.

[8] G. Sanjay, G. Howard, and L. Shun-Tak. The google file system. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles*, pages 29–43, 2003.

[9] H. Yang, A. Dasdan, R. Hsiao, and D. S. Parker. Map-reduce-merge: simplified relational data processing on large clusters. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1029–1040, 2007.

[10] Z. Yang, Q. Tu, K. Fan, L. Zhu, R. Chen, and B. Peng. Performance gain with variable chunk size in gfs-like file systems. In *Journal of Computational Information Systems*, pages 1077–1084, 2008.

[11] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica. Improving mapreduce performance in heterogeneous environments. In *OSDI '07: Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation*, pages 29–42, 2007.