دانشکده فنی و مهندسی
کارشناسی ارشد مهندسی کامپیوتر – نرم‌افزار
گروه مهندسی کامپیوتر و فناوری اطلاعات

ساخت پلتفرم های بهینه برای پردازش دیتاهای بزرگ

واحد درسی
سمینار(تتبع)

استاد راهنما
دکتر علی رضوی

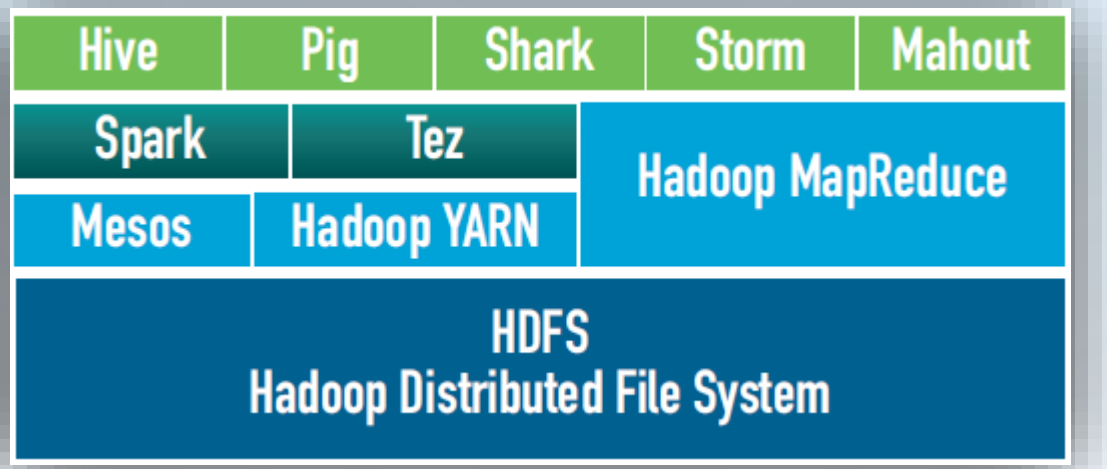
دانشجو ارشد: آرزو درویشی
(ش.د: ۹۹۰۱۲۷۶۲۱)

زمستان 1400

مقدمه

- در دهه های گذشته شاهد تغییر عمده ای در سیستم عامل های پردازش داده ها بوده ایم، زیرا رشد سریع داده های بزرگ نیازمند کاربیشتر برای گسترش در خوشه های بزرگ است. تجزیه و تحلیل داده های بزرگ تقریباً در همه جا از زندگی روزمره ما مانند مراقبت های بهداشتی، تجاری، مالی، کنترل ترافیک، تولید و خرده فروشی استفاده می شود .سیستم های کامپیوتری قدرتمند و کارآمد برای پردازش به موقع داده های بزرگ مورد نیاز است. با این حال، سیستم پایگاه داده سنتی مستقر در یک سرور تنها به دلیل افزایش حجم، تنوع، سرعت و صحت، برای رسیدگی به داده های بزرگ ناکافی است. بنابراین، بسیاری از سیستم عاملهای مقیاس پذیر مبتنی بر خوشه توسعه یافته اند تا به تقاضای روزافزون برای پردازش داده های بزرگ به صورت موازی پاسخ دهند.

شکل زیر خانواده رویکردها و مکانیسم های ظهور سیستم های پردازش داده در مقیاس بزرگ را نشان می دهد. در یک خوشه، داده های ورودی و خروجی در یک سیستم فایل توزیع شده مانند HDFS (سیستم فایل توزیع شده Hadoop) ذخیره می شوند. لایه ذخیره سازی توسط سیستم های مختلف محاسبه داده در مقیاس بزرگ مانند MapReduce/Hadoop، Hadoop YARN، Mesos، Tezو Spark مستقر می شوند. مجموعه ای از سیستم های زیست محیطی برای پردازش انواع مختلفی از داده ها و برنامه ها مانند Hive، Pig، Shark، Storm و Mahout ساخته شده است.



چالش های تحقیق

- **حجم های کاری مختلف:** سیستم های پردازش داده های بزرگ معمولاً انواع مختلفی از برنامه های ارائه شده توسط کاربران مختلف را همزمان ارائه می دهند. هر کار شامل چندین مرحله است و یک عملیات منحصر به فرد در هر مرحله ارائه می شود. انجام وظایف در مراحل مختلف کارهای مختلف نیازمند منابع متفاوتی است و زمان اجرای متفاوتی دارد. به عنوان مثال، یک کار معمولی MapReduce شامل دو مرحله است: Map و Reduce. مرحله Map بر پردازش ورودی/خروجی دیسک متمرکز است و مرحله Reduce وظیفه تجزیه و تحلیل داده ها را بر عهده دارد. بنابراین، وظایف در مرحله Map معمولاً بیشتر به ورودی/خروجی دیسک متکی هستند و وظایف در مرحله Reduce نیاز به منابع بیشتری در حافظه و CPU دارند. حتی در مرحله کاهش، عملیات مختلف به میزان متفاوتی از منابع متکی است.

- **وابستگی مراحل:** وابستگی معمولاً بین مراحل هر برنامه در سیستم های پردازش داده های بزرگ وجود دارد. برای هر دو مرحله متوالی یک کار، داده های خروجی مرحله قبلی داده های ورودی مرحله بعدی است. از یک سو، مرحله بعدی نمی تواند قبل از مرحله قبلی خود به پایان برسد. مدیریت منابع در خوشه باید با چنین وابستگی مطابقت داشته باشد. از سوی دیگر، مرحله بعدی می تواند زودتر از اتمام مورد قبلی شروع شود. اولین جزء در هر مرحله (به استثنای مرحله اول در یک کار) shufflingنامیده می شود. وظیفه انتقال داده های خروجی تولید شده از مرحله قبلی به خود را بر عهده دارد. شروع زودتر از مرحله shufflingمی تواند با تداخل با مرحله shufflingو پایان مرحله قبلی به بهبود عملکرد کمک کند. ما دریافتیم که این دوره منحصر به فرد نقش مهمی در مدیریت منابع ایفا می کند. تعیین مناسب زمان پایان یک مرحله و زمان شروع مرحله بعدی می تواند از منابع بیکار یا داده های انباشته منتظر پردازش جلوگیری کند. با این حال، مراحل مختلف کارهای مختلف اندازه های متفاوتی از داده ها را ایجاد می کند و سرعت انتقال داده ها با توجه به پهنای باند سیستم و تداخل عملیات جابجایی با سایر کارها، همواره در حال تغییر است. بنابراین، برای مدیریت منابع تعیین بهترین زمان برای شروع مراحل بعدی برای کارهای مختلف چالش برانگیز است.

- **عملکرد ناهماهنگ گره های خوشه ای:** در حین اجرای هر خوشه در عمل، عملکرد زمان اجرای هر گره متفاوت است. عوامل زیادی می توانند عملکرد سیستم یک سرور را در خوشه تحت تأثیر قرار دهند، مانند تعداد فرایندهایی که در سرور همزمان اجرا می شود، درصد اشغال حافظه و رقابت عملیات خواندن/نوشتن روی دیسک. این به زمانبند نیاز دارد تا تخصیص منابع را در زمان واقعی با توجه به تغییر عملکرد هر سرور تنظیم کند.
- **خرابی گره در خوشه:** در یک خوشه در مقیاس بزرگ، خرابی گره ها و راه اندازها (سرورهای کند) در عمل طبیعی است. معمولاً تحمل خطا در سیستم های محاسباتی برای رسیدگی به این گونه مسائل اجرا می شود. مکانیسم حدس و گمان یک راه حل رایج برای کاهش تاثیر چنین شکست هایی است.

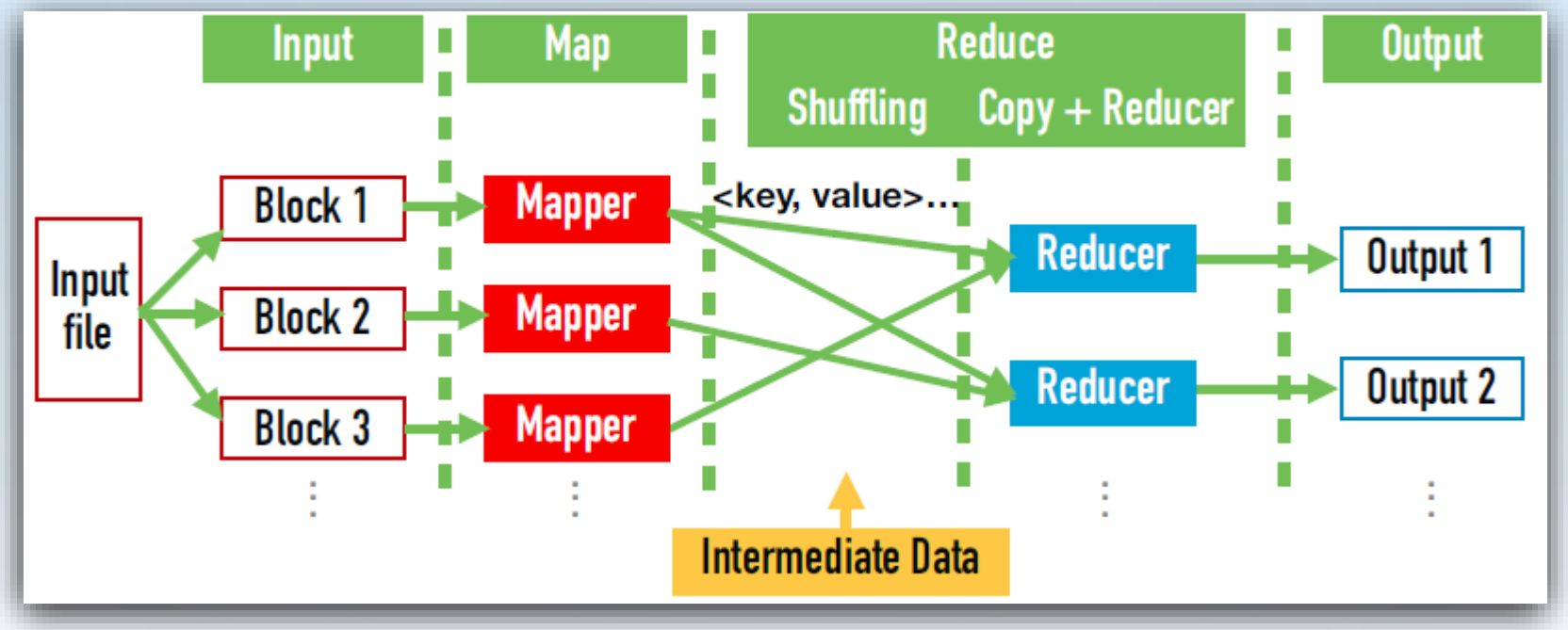
پیش زمینه

- ابتدا، ما مدل برنامه نویسی MapReduce، به ویژه جریان کار دو مرحله در یک برنامه MapReduceرا معرفی می کنیم. دوم، ساختارهای هر دو پلتفرم های Hadoopو Hadoop YARNبه تفصیل توضیح داده شده است. کار ما به عنوان اجزای پلاگین در این سیستم عامل ها اجرا می شود. سرانجام، سیاستهای زمانبندی معمولی درخت، FIFO، Fairو Capacityمعرفی شده است. همه آنها زمانبندی پیش فرض در اکثر سیستم های محاسبه داده بزرگ هستند و می توانند در فایل پیکربندی سیستم تنظیم شوند. ما آنها را به عنوان مبنایی برای مقایسه در ارزیابی عملکرد خود در نظر می گیریم

MAPREDUCE

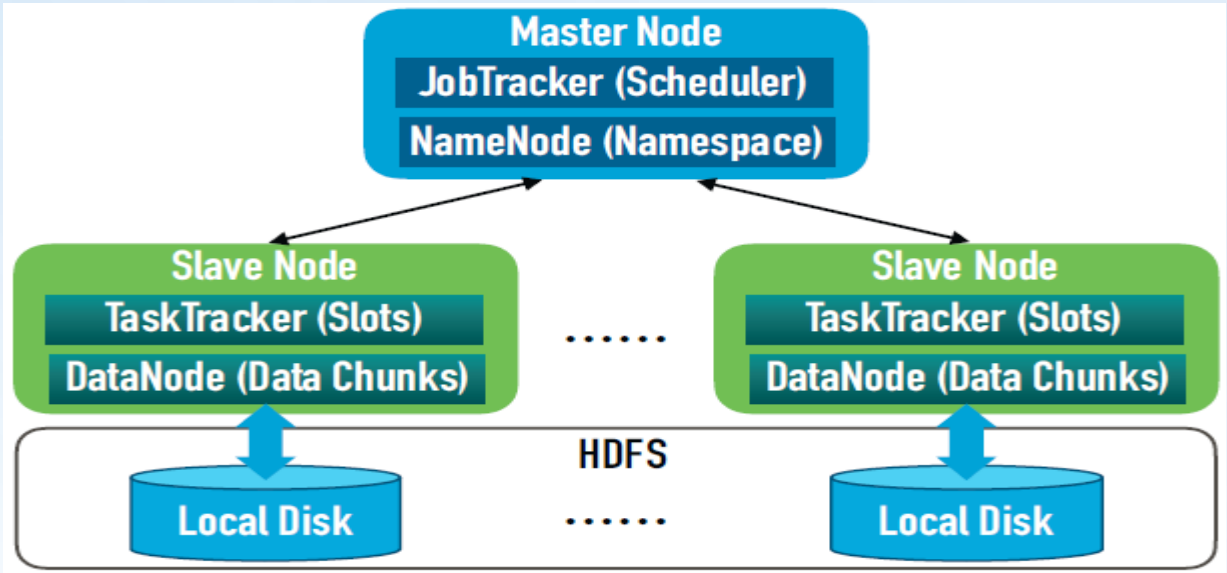
- یک مدل برنامه نویسی است که توسط گوگل برای پردازش مجموعه داده های بزرگ بر روی خوشه های رایانه معرفی شده است .یک کار معمولی MapReduce شامل دو مرحله است: map و Reduce. هر مرحله شامل چندین کار یکسان است. یکی از وظیفه map (mapper)یک بلوک از داده های خام را که در یک سیستم فایل توزیع شده مشترک ذخیره می شود، پردازش می کند و داده های میانی را به شکل <کلید ، مقدار> تولید می کند. سه مرحله در مرحله Reduce وجود دارد: Shuffling،Copy وReduce
- Shuffling . مسئول کپی کردن داده های میانی از مرحله Mapبه مرحله Reduce است. پس از اتمام انتقال همه داده های میانی، مرحله Copy داده ها را جمع آوری می کنند ومرحله Reduce داده های میانی را پردازش می کند و نتایج نهایی را تولید می کند.

طرح پردازش داده های موازی MAPREDUCEرا نشان می دهد



HADOOP MAPREDUCE

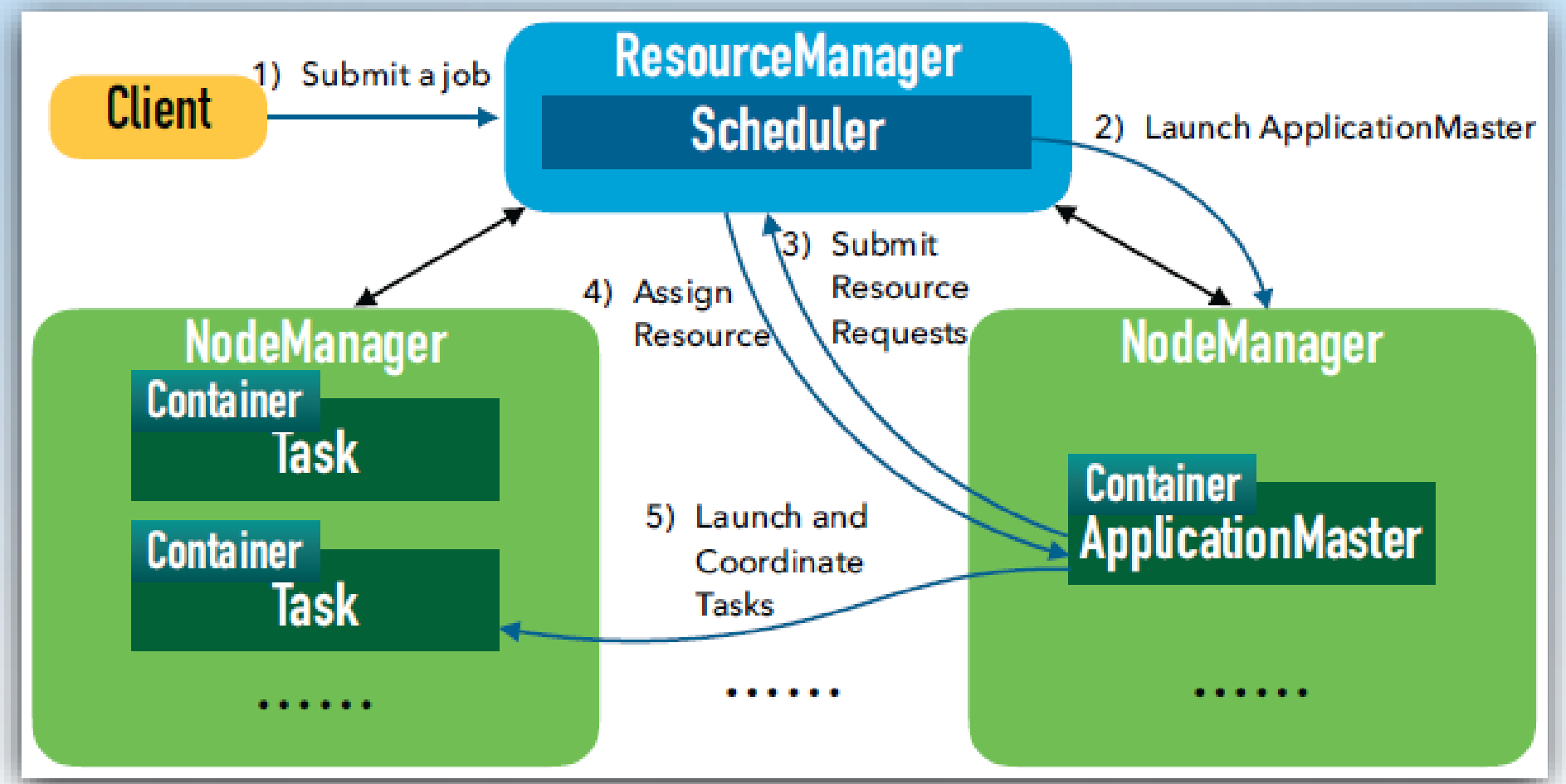
- Apache Hadoop MapReduceیک برنامه منبع باز MapReduceاست. در یک خوشه Hadoop، یک گره اصلی متمرکز و چندین گره slaveتوزیع شده وجود دارد. دو جزء اصلی در Hadoopوجود دارد:
  - سیستم فایل توزیع شده Hadoop (HDFS)و سیستم اجرای کار MapReduce



دیسک های محلی از گره های slaveبه عنوان سیستم فایل توزیع شده با هم ترکیب می شوند. همه داده های ورودی و خروجی به چندین بلوک داده تقسیم شده و جداگانه در آن ذخیره می شوند. هر بلوک داده می تواند چندین کپی اضافی برای تحمل خطا و محل داده داشته باشد. NameNode متمرکز مستقر در گره اصلی مسئول مدیریت HDFSاست. فراداده، نام فایلها و مکانهای بلوک را ذخیره می کند. در هر گره slave، یک ماژول DataNodeداده های ذخیره شده را مدیریت می کند.

HADOOP YARN

- Hadoop YARNنسل بعدی Hadoop است. مشابه Hadoop MapReduce ، در Hadoop YARN، یک گره اصلی متمرکز وجود دارد که ResourceManager را اجرا می کند و چندین گره slave توزیع شده توسط NodeManager روی هر slave مستقر شده است. Hadoop YARNدارای دو تفاوت اصلی با نسل اول Hadoop است. اول، YARNاز مدیریت منابع دانه ریز پشتیبانی می کند. هر گره slave ظرفیت منبع را در قالب هسته و حافظه CPU مشخص می کند. هر وظیفه در هنگام ارسال یک تقاضای منابع را مشخص می کند و ResourceManager با اعطای یک منبع در یک گره slave به تقاضای منابع پاسخ می دهد. تفاوت دوم این است که مدیریت منابع و هماهنگی کاری در YARN از هم جدا هستند. یک جزء جدید ApplicationMaster از هماهنگی کار مراقبت می کند. و ResourceManager فقط مسئول زمانبندی است. در این حالت، YARNمی تواند از چارچوب های مختلف در یک خوشه مانند MapReduce و Spark پشتیبانی کند


Client
1) Submit a job
ResourceManager
Scheduler
2) Launch ApplicationMaster
3) Submit Resource Requests
4) Assign Resource
NodeManager
Container
Task
Container
Task
.....
5) Launch and Coordinate Tasks
.....
NodeManager
Container
ApplicationMaster
.....

سیاستهای برنامه ریزی رایج

- در زمانبندی FIFO، همه فعالیت ها به ترتیب زمان ارسال در صف قرار می گیرند. فعالیتی که ابتدا ارسال می شود ابتدا ارائه می شود. هنگامی که وظایف آن برای اجرا اختصاص داده شد، کار بعدی در صف اجرا می شود. هدف برنامه ریزی Fair اختصاص منابع عادلانه در فعالیت ها در طول زمان است. اولویت ها و وزن ها ممکن است برای کارهای مختلف در تخصیص منابع پیکربندی شوند. برنامه ریزی Capacity، عملکردی مشابه برنامه ریزی Fairرا ارائه می دهد. در قسمت Capacity، صف های کاری متعددی برای کاربران مختلف تعریف شده است. زمانبندی منابع یکسانی را برای هر صف ارائه می دهد در حالی که زمانبندی FIFOبرای کارها در همان صف ارائه شده است.
- سیاست های زمان بندی بالا تنظیمات پیش فرض در اکثر سیستم های محاسباتی مانند Hadoop MapReduce، Hadoop YARNو Sparkاست. با این حال، آنها کارآیی استفاده از منابع سیستم را در نظر نمی گیرند.

تخصیص منابع مبتنی بر کار

- ما یک طرح مدیریت منابع جدید به نام FRESH را پیشنهاد می کنیم. درFRESH، ما یک مؤلفه نظارتی جدید برای شناسایی بارهای کاری زمان اجرا هر مرحله در خوشه ایجاد می کنیم. بر اساس نتایج نظارت، FRESHبه صورت پویا تخصیص منابع را برای مراحل مختلف تنظیم می‌کند تا بهره برداری از منابع سیستم را بهبود بخشد و طول انجام کارهای دسته‌ای را کاهش دهد. علاوه بر این، یک مکانیسم عدالت بهبود یافته در FRESH تضمین می کند که منابع برابر به هر کار در حال اجرا در خوشه اختصاص داده می شود. FRESH در پلتفرم Hadoop MapReduceپیاده سازی شده است اما مکانیسم کلی آن به راحتی در سایر سیستم های محاسباتی قابل استفاده است.

پیشینه و انگیزه

- یکی از چالش های بزرگ برای کاربران Hadoop این است که چگونه سیستم های خود را به درستی پیکربندی کنند. Hadoop به عنوان یک سیستم پیچیده با مجموعه بزرگی از پارامترهای سیستم ساخته شده است. در حالی که انعطاف پذیری برای سفارشی کردن یک خوشه Hadoop برای برنامه های مختلف را فراهم می کند، اغلب برای کاربر درک آن پارامترهای سیستم و تعیین مقادیر بهینه برای آنها دشوار است. در این کار، ما یک پارامتر بسیار مهم Hadoop، پیکربندی اسلات را هدف قرار می دهیم و مجموعه ای از راه حل ها را برای بهبود عملکرد، با توجه به گستردگی مجموعه ای از کارها و عدل را در میان آنها، توسعه می دهیم.
- مقادیر معمولاً با اعداد اکتشافی بدون در نظر گرفتن ویژگی های شغل تنظیم می شوند. چنین تنظیمات ایستا مطمئناً نمی تواند عملکرد بهینه را برای بارهای کاری متفاوت ارائه دهد. بنابراین، هدف اصلی ما رسیدگی به این موضوع و بهبود عملکرد makepan است. علاوه بر طول عمر، عدالت یکی دیگر از معیارهای عملکردی است که ما در نظر می گیریم. عدالت زمانی حیاتی است که چندین کار به طور همزمان در یک خوشه اجرا شوند.
- Makespan, FRESHدو مدل، پیکربندی اسلات استاتیک و پیکربندی اسلات پویا است. در مدل اول، FRESHپیکربندی اسلات را قبل از راه‌اندازی خوشه Hadoopاستخراج می‌کند و از همان تنظیمات در طول اجرا درست مانند Hadoopمعمولی استفاده می‌کند. در مدل دوم، FRESHبه یک شکاف اجازه می دهد تا نوع خود را پس از راه اندازی کلاستر تغییر دهد. هنگامی که یک اسلات وظیفه خود را به پایان می رساند، راه حل ما به صورت پویا اسلات را پیکربندی می کند و وظیفه بعدی را به آن اختصاص می دهد. نتایج تجربی ما نشان می دهد که FRESHبه طور قابل توجهی عملکرد را از نظر ساخت و عدالت در سیستم بهبود می بخشد.

فرمول مسأله

- ما در نظر می گیریم که بطور کلی یک کاربر دسته ای از n شغل، J = {J1, J2, . . . Jn}، را به یک خوشه Hadoop با اسلاتهای S ارسال می کند. هر شغل J_i شامل $n_m(i)$ وظایف نقشه و $n_r(i)$ ساده سازی وظایف است. اجازه دهید s_m و s_r تعداد کل اسلات های نقشه باشند و شکاف ها را در خوشه ساده کنید، یعنی $S = s_m+s_r$. ما فرض می کنیم که یک مکانیسم کنترل پذیرش در این خوشه در حال اجرا است به طوری که یک حد بالا در تعداد کارهایی که می توانند همزمان اجرا شوند وجود دارد. به طور خاص، در این شغل فرض می‌کنیم که در هر زمان، حداکثر k شغل در فاز نقشه و حداکثر k شغل در فاز ساده سازی وجود دارد. بنابراین، حداکثر تعداد شغل های فعال در خوشه k، ۲است. در اینجا، kیک پارامتر مشخص شده توسط کاربر برای ایجاد تعادل بین عدل و استاندارد است. هدف ما این است که زمان ساخت (یعنی طول تکمیل کل) مجموعه شغل J را به حداقل برسانیم و در عین حال به تعادل در بین این کارها نیز دست یابیم.

پیکربندی اسلات استاتیک

- ابتدا، الگوریتم خود را به صورت FRESH برای پیکربندی اسلات استاتیک ارائه می‌کنیم، جایی که تخصیص نقشه‌ها و اسلات‌های ساده سازی در فایل‌های پیکربندی از پیش تنظیم می‌شوند و هنگام راه‌اندازی خوشه Hadoop بارگذاری می‌شوند. هدف ما این است که پیکربندی بهینه اسلات را با توجه به پروفایل های بار کاری مجموعه ای از کارهای Hadoop بدست آوریم.
- ما فرض می کنیم که حجم کار هر شغل به عنوان دانش قبلی در دسترس است. این اطلاعات را می توان از سوابق اجرایی تاریخی یا برآورد تجربی به دست آورد. فرض کنید $\bar{t}_m(i)$ و $\bar{t}_r$ (i) میانگین زمان اجرای یک کار نقشه و یک کار ساده سازی job j_i باشد.
- ما w_m (i) وw_r (i) را به‌عنوان بارهای کاری وظایف نقشه و کاهش وظایف j_i تعریف می‌کنیم، که نشان‌دهنده جمع‌بندی زمان اجرای همه وظایف نقشه و ساده سازی وظایف j_i است. بنابراین، w_m (i) وw_r (i) را می توان به صورت زیر تعریف کرد:

- $wm(i) = nm(i) \cdot \bar{t}m(i)$, $wr(i) = nr(i) \cdot \bar{t}r(i)$.

Algorithm 3.3.1: Static Slot Configuration

1: for $s_m = 1$ to S do
2: $s_r = S - s_m$
3: for $i = 1$ to n do
4: $w_m(i) = n_m(i) \cdot \bar{t}_m(i), \quad w_r(i) = n_r(i) \cdot \bar{t}_r(i)$
5: $W_i = w_m(i)$
6: end for
7: $\mathcal{M} = \{1, 2, \ldots, k\}, \mathcal{R} = \{\}, \mathcal{R}' = \{\}, T = 0$
8: while $\mathcal{M} \bigcup \mathcal{R} \neq \phi$ do
9: $u = \arg\min_{i \in \mathcal{M}} W_i, \quad c_m = \frac{s_m}{|\mathcal{M}|}$
10: $v = \arg\min_{i \in \mathcal{R}} W_i, \quad c_r = \frac{s_r}{|\mathcal{R}|}$
11: if $\frac{W_u}{c_m} < \frac{W_v}{c_r}$ then
12: $\mathcal{M} \leftarrow \mathcal{M} - u, \quad T = T + \frac{W_u}{c_m}, \quad W_u = w_r(u)$
13: DeductWorkload($\mathcal{M}, w_m(u)$)
14: DeductWorkload($\mathcal{R}, \frac{w_m(u)}{c_m} \cdot c_r$)
15: pick a new job from J and add its index to $\mathcal{M}$
16: if $|\mathcal{R}| < k$ then $\mathcal{R} \leftarrow \mathcal{R} + u$
17: else add u to the tail of $\mathcal{R}'$
18: else
19: $\mathcal{R} \leftarrow \mathcal{R} - v, \quad T = T + \frac{W_v}{c_r}$
20: DeductWorkload($\mathcal{R}, W_v$)
21: DeductWorkload($\mathcal{M}, \frac{W_v}{c_r} \cdot c_m$)
22: if $|\mathcal{R}'| > 0$ then move $\mathcal{R}'[0]$ to $\mathcal{R}$
23: end if
24: end while
25: if $T < Opt\_MS$ then $Opt\_MS = T, Opt\_SM = s_m$
26: end for
27: return Opt_SM and Opt_MS

حلقه بیرونی تمام پیکربندی‌های اسلات ممکن را برمی‌شمارد (یعنی s_m و s_r).

ابتدا بارهای کاری نقشه هر کار را محاسبه می کنیم و فازها را کاهش می دهیم، یعنی W_m (i) و w_r (i)، و مقدار اولیه W_i را تنظیم می کنیم

متغیرهای M و R را مقداردهی اولیه می کنیم، R نشان دهنده لیست مرتبی از کارهای معلق است که فاز نقشه خود را به پایان رسانده اند، اما هنوز وارد فاز ساده سازی خود نشده اند، و T، مدت زمان مجموعه کارها را ثبت می کند. مؤلفه اصلی الگوریتم حلقه while است

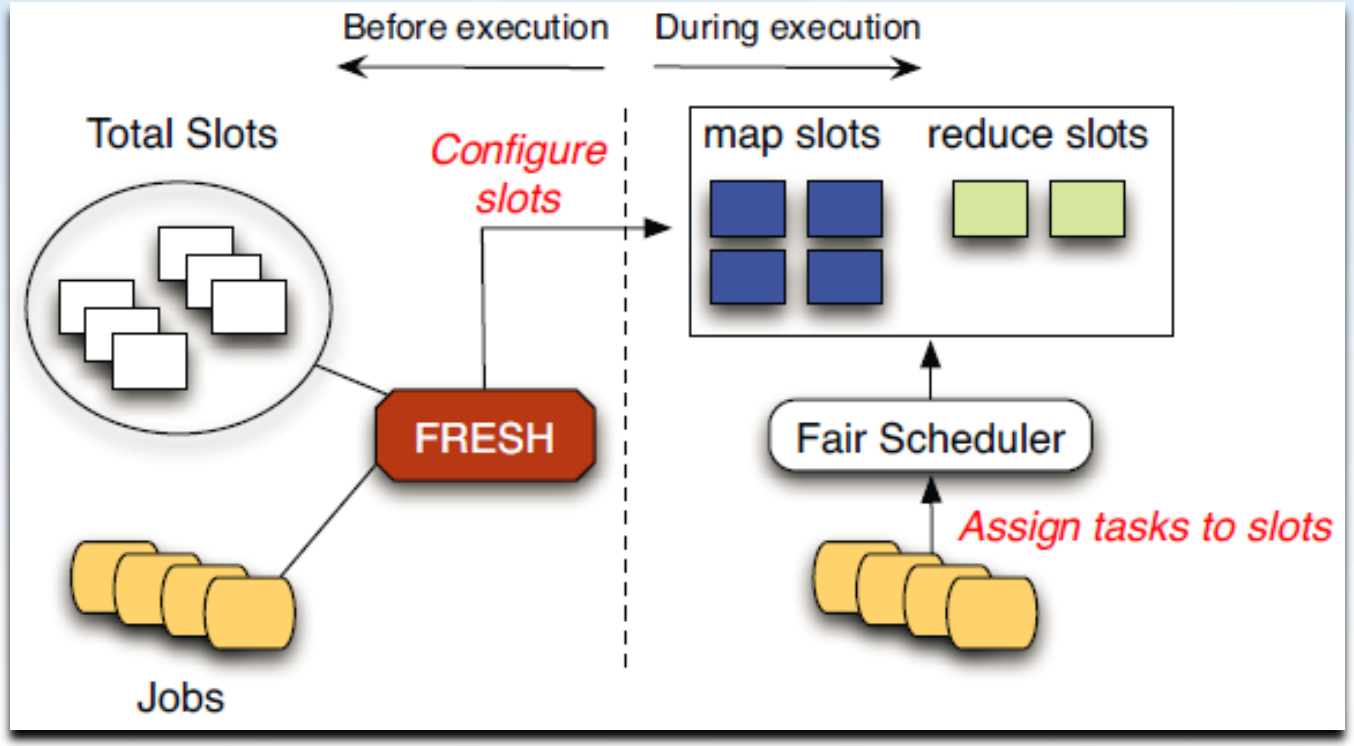
فاصله زمانی را محاسبه می کند و زمانی که M و R هر دو خالی باشند به پایان می رسد. در این حلقه، الگوریتم ما ترتیب اجرای همه کارها را تقلید می کند. M و R هر دو دست نخورده باقی می مانند تا زمانی که یکی از کارهای در حال اجرا فاز نقشه فعلی (یا ساده سازی) خود را به پایان برساند. در هر دور اجرا در حلقه while، الگوریتم ما اولین کاری را که وضعیت را تغییر می‌دهد، پیدا می‌کند و سپس مجموعه‌های کار را متناسب با آن به‌روزرسانی می‌کند. این کار هدف بسته به حجم کار باقیمانده و تعداد اسلات های اختصاص داده شده به هر کار می تواند در فاز نقشه یا ساده سازی باشد.

دو شغل (J_u و J_v) را پیدا می‌کنند که کمترین حجم کاری باقیمانده را به ترتیب در M و R دارند. این دو شغل کاندیدایی هستند که ابتدا مراحل فعلی خود را به پایان برسانند.

اگر J_u ابتدا فاز نقشه‌اش را تمام کند (مورد خطهای 10-16)، سپس u را از M حذف می‌کنیم

الگوریتم یک کار جدید را برای ورود به فاز نقشه خود در خط 14 انتخاب می کند. در نهایت، u را به R اضافه می کنیم تا در صورت عدم رسیدن به محدودیت ظرفیت R، فاز ساده سازی آن شروع شود. در غیر این صورت، u به دنباله لیست معلق R اضافه می شود

پیکربندی اسلات استاتیک.



پیکربندی اسلات داینامیک

- هدف اصلی این مدل، فعال کردن یک شکاف برای تغییر نوع آن (یعنی نقشه برداری یا ساده سازی) پس از راه اندازی خوشه است. برای انجام آن، ما راه حل هایی را برای پیکربندی اسلات ها و اختصاص وظایف به اسلات ها توسعه می دهیم. علاوه بر این، عادلانه مصرف منابع را در میان کارها بازتعریف می کنیم. بنابراین، هدف ما این است که ضمن دستیابی به بهترین عدل، بدون تنزل عملکرد ساخت، طول کار را به حداقل برسانیم.

Algorithm 3.3.2: Configure a Free Slot

1: if a map task of job J_i is finished then
2: update $\bar{t}_m(i)$, $n'_m(i)$, $w'_m(i)$ and RW_m
3: end if
4: if a reduce task of job J_i is finished then
5: update $\bar{t}_r(i)$, $n'_r(i)$, $w'_r(i)$ and RW_r
6: end if
7: $expSm$ =CalExpSm()
8: if $expSm > s_m$ then set the slot to be a map slot
9: else set the slot to be a reduce slot

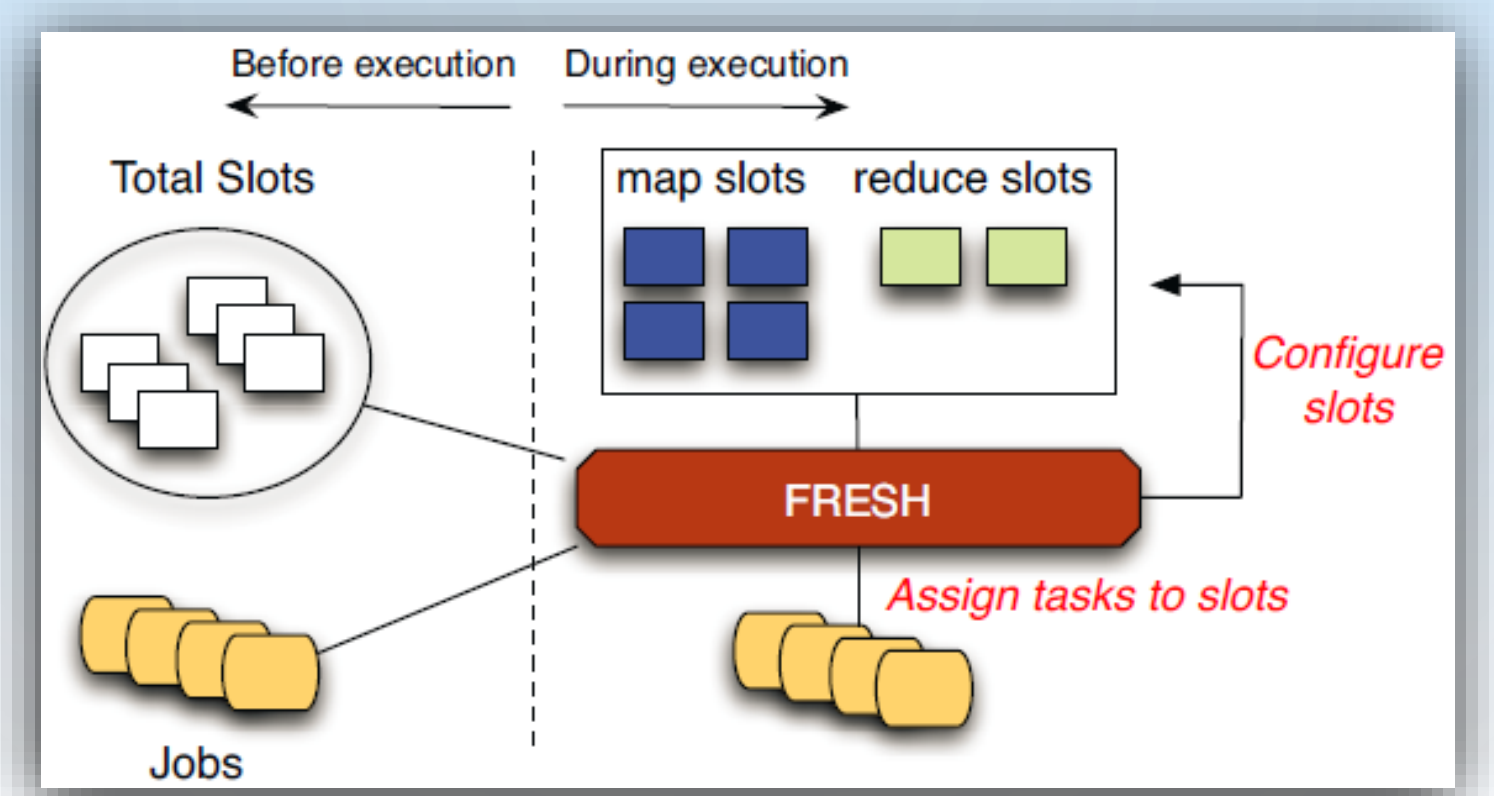
ما از این الگوریتم برای استخراج پیکربندی بهینه اسلات در حالت آنلاین توسعه می‌دهیم. ما از اصل طراحی اولیه با یک کنترل مبتنی بر آستانه پیروی می کنیم تا اثرات منفی ناشی از تغییرات ناگهانی در نقشه را کاهش دهیم / بار کاری را کاهش دهیم. هنگامی که یک کار نقشه/کاهش به پایان رسید، الگوریتم زمان اجرای کار را جمع‌آوری می‌کند و مجموعه‌ای از اطلاعات آماری شامل میانگین زمان اجرای کار، تعداد وظایف باقی‌مانده، حجم کار باقی‌مانده از کار j_i، و کل بارهای کاری باقی‌مانده را به‌روزرسانی می‌کند. خطوط ۱-۴). به دنبال آن، الگوریتم تابعی به نام CalExpSm را فراخوانی می کند تا تعداد مورد انتظار اسلات نقشه ($expS_m$) را بر اساس اطلاعات آماری فعلی محاسبه کند. اگر انتظار بیشتر از تعداد اسلات های فعلی نقشه (S_m) باشد، این اسلات رایگان به اسلات نقشه تبدیل می شود. در غیر این صورت، ما آن را به عنوان یک اسلات کاهش تنظیم می کنیم.

Algorithm 3.3.3: Function $CalExpSm()$

1: $\theta_{cur} = \frac{RW_m}{RW_m + RW_r}$
2: $a = \arg\min_{i \in \mathcal{M}} w'_m(i)$
3: **if** the progress of job $J_a < \tau_1$ **then**
4: return $\theta_{cur} \cdot S$
5: **else**
6: Let job J_b be the next that will start its map phase
7: $RW'_m = RW_m - w'_m(a) \cdot k + w_m(b)$
8: $RW'_r = RW_r + w_r(a)$
9: $\theta_{exp} = RW'_m / (RW'_r + RW'_m)$
10: calculate η using Theorem 3.3.1
11: $s'_m = s_m + \theta_{cur} \cdot \eta$
12: **if** $\frac{|\frac{s'_m}{S} - \theta_{exp}|}{\theta_{exp}} > \tau_2$ **then return** $\theta_{exp} \cdot S$
13: else return $\theta_{cur} \cdot S$
14: **end if**

زمانی که Ja حجم کار نقشه باقیمانده خود را w’ m(a) تمام می کند، فرض می کنیم کارهای دیگر در M تقریباً همان پیشرفت را داشته اند. بنابراین حجم کار نقشه کل به میزان w’ m(a) · k کاهش می یابد. سپس یک کار جدید به مجموعه M می‌پیوندد، اجازه دهید job Jb باشد، و wm(b) به کل حجم کار باقی‌مانده RWm اضافه می‌شود (خط ۷). در همین حال، Ja به مجموعه R یا R’ تعلق دارد و حجم کاری کاهش یافته wr(i) به کل RWr کاهش بار باقیمانده اضافه می شود (خط ۸). متغیر θexp در خط ۹ نشان دهنده نسبت مورد انتظار اسلات نقشه در آن نقطه است. در مرحله بعد، زمانی که Ja فاز نقشه خود را به پایان می رساند، تابع تعداد اسلات های نقشه را به دنبال نسبت پیکربندی θcur تخمین می زند. این شامل تعداد اسلات های آزاد شده از نقطه فعلی است. واضح است که هیچ اسلات نقشه دیگری بر اساس تعریف Ja منتشر نشده است، بنابراین ما فقط باید تعداد اسلات های کاهش موجود را در این دوره تخمین بزنیم. در الگوریتم ۳.۳.۳، ما از η برای نشان دادن این عدد و تخمین مقدار آن بر اساس قضیه ۳.۳.۱ زیر استفاده می کنیم. در خط ۱۱، تعداد اسلات های نقشه s’ m را با استفاده از نسبت پیکربندی فعلی پیش بینی می کنیم. در نهایت، تابع نسبت تخمینی را با نسبت مورد انتظار در خط ۱۲ مقایسه می کند. اگر اختلاف بیش از آستانه τ2 باشد، نسبت مورد انتظار آینده θexp را در نظر خواهیم گرفت. در غیر این صورت، ما به استفاده از نسبت پیکربندی فعلی θcur ادامه خواهیم داد.

- پیکربندی اسلات داینامیک.



وظایف را به اسلات ها اختصاص دهید

- هنگامی که نوع اسلات آزاد شده مشخص شد، FRESH وظیفه ای را به آن اسلات اختصاص می دهد. اساساً، ما باید یک کار فعال را انتخاب کنیم و اجازه دهیم اسلات نقشه/کاهش موجود، یک کار نقشه/کاهش از آن کار را انجام دهد. در FRESH، ما از ایده اصلی در زمان‌بندی منصفانه پیروی می‌کنیم، اما به جای آن از معیار انصاف کلی جدید استفاده می‌کنیم: مصرف منابع را برای هر شغل بر اساس بیشترین کمبود انصاف کلی انتخاب کنید.

پیاده سازی

- ما FRESH را روی Hadoop نسخه ۰.۲۰.۲ پیاده سازی کرده ایم. برای FRESH-static، ما یک برنامه خارجی برای استخراج بهترین تنظیمات اسلات و اعمال آن در فایل پیکربندی Hadoop ایجاد می کنیم. خود سیستم Hadoop برای FRESHstatic اصلاح نشده است. برای پیاده سازی FRESH-dynamic، چند کامپوننت جدید به Hadoop اضافه کرده ایم. ابتدا، ما سیاست کنترل پذیرش را با پارامتر k پیاده‌سازی می‌کنیم، به‌عنوان مثال، حداکثر k کار مجاز است به طور همزمان در فاز نقشه یا در فاز کاهش اجرا شود. دوم، ما دو ماژول جدید در JobTracker ایجاد می کنیم. یک ماژول اطلاعات اماری مانند میانگین زمان اجرای یک کار را به روز می کند و حجم کار باقی مانده از هر کار فعال را تخمین می زند. ماژول دیگر به گونه ای طراحی شده است که یک اسلات آزاد را به عنوان یک اسلات نقشه یا یک شکاف کاهشی پیکربندی کند و طبق الگوریتم های بخش ۳.۳.۲، وظیفه ای را به آن اختصاص دهد. دو پارامتر آستانه در الگوریتم ۳.۳.۳ به صورت $\tau1 = 0.8$ و $\tau2 = 0.6$ تنظیم شده است. ما با مقادیر مختلف آزمایش کرده‌ایم و دریافتیم که عملکرد زمانی نزدیک است که $\tau1 \in [0.7$ و $0.9]$، $\tau2 \in [0.5$ و $0.7]$. به دلیل محدودیت صفحه، بحث در مورد این دو مقدار اکتشافی را حذف می کنیم. علاوه بر این، پروفایل های شغلی (زمان اجرای وظایف) بر اساس نتایج تجربی زمانی که یک کار به صورت جداگانه اجرا می شود، تولید می شود. با این حال، ما به طور تصادفی ۳۰٪± سوگیری را برای زمان اجرای اندازه‌گیری شده معرفی می‌کنیم و از آنها به عنوان پروفایل‌های شغلی که تخمین‌های تقریبی را نشان می‌دهند، استفاده می‌کنیم.

ارزیابی عملکرد

- در این بخش، عملکرد FRESH را ارائه کرده و با راه حل های دیگر مقایسه می کنیم. با توجه به دسته‌ای از کارهای MapReduce، معیارهای اصلی عملکرد ما زمان ساخت است، یعنی زمان پایان آخرین کار، و انصاف در بین همه کارها. ما عمدتاً با سیستم Hadoop معمولی با زمان‌بندی Fair و پیکربندی اسلات استاتیک مقایسه می‌کنیم. در تنظیمات ما، هر اسلیو دارای ۴ اسلات است، بنابراین در Hadoop معمولی سه تنظیمات ثابت وجود دارد، ۱

- اسلات نقشه / ۳ اسلات کاهش،
- ۲ اسلات نقشه / ۲ اسلات کاهش،
- ۳ اسلات نقشه / ۱ اسلات کاهش.

ما به ترتیب از Fair-1:3، Fair-2:2 و Fair-3:1 برای نمایش این سه تنظیمات استفاده می کنیم.

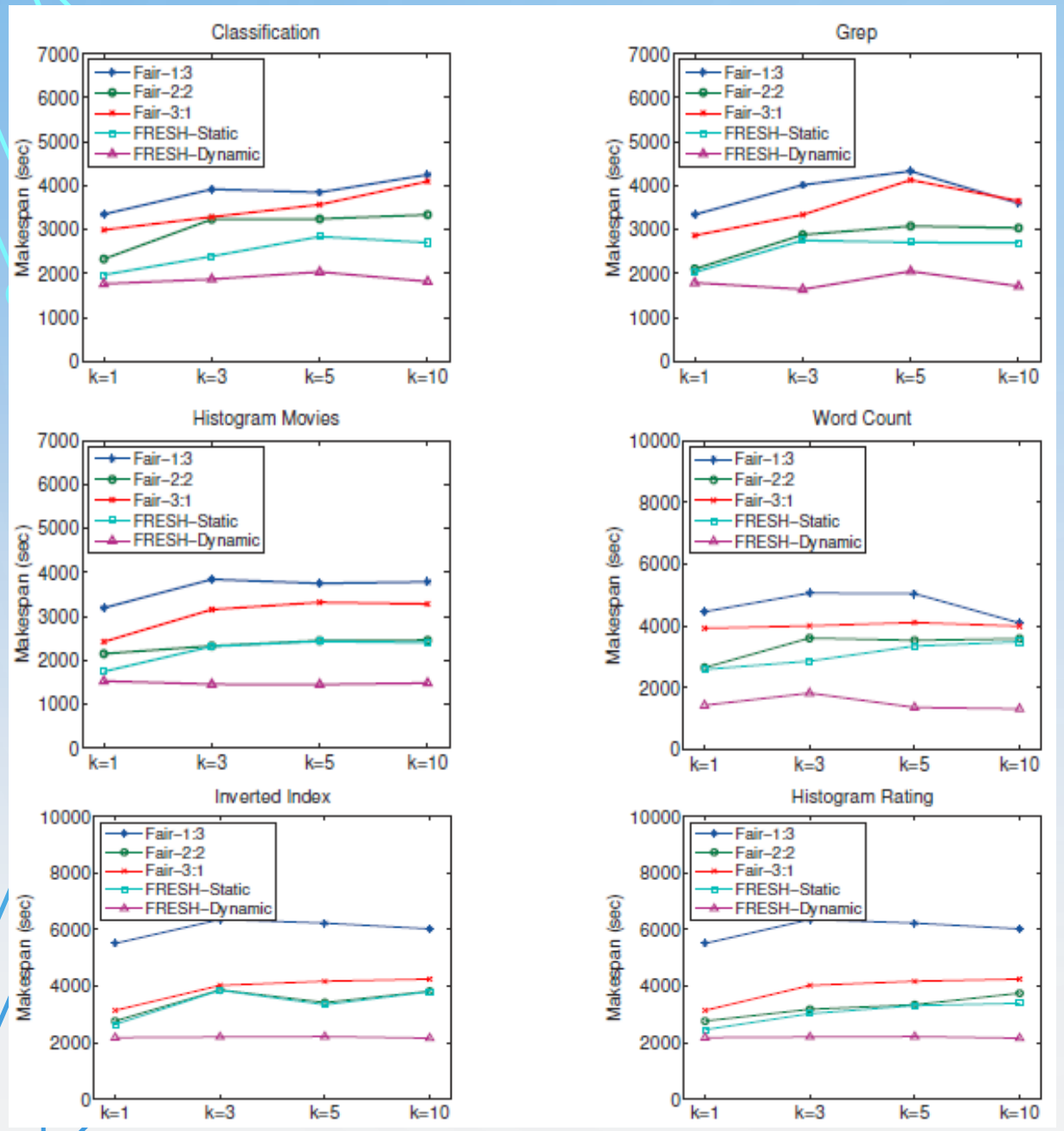
- ما دو دسته از آزمایش‌ها را با بار کاری متفاوت انجام داده‌ایم، حجم‌های کاری ساده از یک نوع کار (انتخاب شده از شش معیار MapReduce) و بارهای کاری ترکیبی مجموعه‌ای از کارهای ترکیبی را نشان می‌دهند. در بازنشانی این بخش، نتایج ارزیابی را با این دو دسته از بار کاری به طور جداگانه ارائه می کنیم.

ما شش معیار HADOOPزیر را از PURDUE MAPREDUCE BENCHMARKS SUITE [26] برای ارزیابی عملکرد انتخاب می‌کنیم.

- طبقه بندی: داده های رتبه بندی فیلم را به عنوان ورودی در نظر بگیرید و فیلم ها را بر اساس رتبه بندی آنها طبقه بندی کنید.
- نمایه معکوس: فهرستی از اسناد ویکی‌پدیا را به‌عنوان ورودی بگیرید و نمایه‌سازی کلمه به سند ایجاد کنید.
- Wordcount: فهرستی از اسناد ویکی پدیا را به عنوان ورودی بگیرید و تعداد تکرار هر کلمه را بشمارید.
- Grep: فهرستی از اسناد ویکی‌پدیا را به عنوان ورودی بگیرید و یک الگو را در فایل‌ها جستجو کنید.
- رتبه بندی هیستوگرام: یک هیستوگرام از داده های رتبه بندی فیلم (با ۵ bins) ایجاد کنید.
- فیلم های هیستوگرام: یک هیستوگرام از داده های رتبه بندی فیلم (با ۸ bins) ایجاد کنید.

عملکرد با بارهای کاری ساده

- برای آزمایش بارهای کاری ساده، 10 کار Hadoop برای هر یک از 6 معیار بالا ایجاد می کنیم. هر مجموعه از 10 کار، مجموعه داده های ورودی یکسانی را به اشتراک می گذارد و به طور متوالی با فاصله زمانی 2 ثانیه به خوشه Hadoop ارسال می شود. علاوه بر این، ما مقادیر مختلف k را برای نشان دادن تأثیر سیاست کنترل پذیرش، به ویژه k = 1,3,5,10 آزمایش کرده‌ایم.



شکل روبرو عملکرد makespan را نشان می دهد. اول، مشاهده می‌کنیم که در هادوپ معمولی، بهترین عملکرد بیشتر زمانی به دست می‌آید که k=1 باشد، یعنی فقط یک کار در نقشه و فاز کاهش که معادل زمان‌بندی FIFO (First-In-First-Out) است. این نشان می‌دهد که در حالی که انصاف را بهبود می‌بخشد، زمان‌بندی نمایشگاه، طول عمر مجموعه‌ای از کارها را قربانی می‌کند. علت اصلی اختلاف منابع بین کارها هست که اجرای هر کار را طولانی می کند. راه حل ما، FRESH-static بدتر از بهترین تنظیمات با زمانبندی Fair نیست. در برخی از حجم کار مانند "طبقه بندی"، بهبود کافی است. علاوه بر این، FRESH-dynamic همیشه بهبود قابل توجهی را در تمام تنظیمات آزمایش شده به همراه دارد.

- در بیشتر موارد، ارزش منصفانه یک تابع افزایشی بر روی k است. به خصوص هنگامی که k = 10، که در آن همه مشاغل مجاز به اجرای همزمان هستند (بدون کنترل پذیرش)، تقریباً همه تنظیمات ارزش انصاف خوبی را به دست می آورند. از آنجایی که همه کارها در حجم کاری ساده از یک معیار هستند، فازهای نقشه خود را به صورت موجی تمام می کنند و تقریباً در همان زمان وارد فاز کاهش می شوند. بنابراین زمانبندی منصفانه در این مورد به خوبی عمل می کند (k=10). به طور کلی، FRESH-Dynamic از همه طرح‌های دیگر بهتر عمل می‌کند و حتی زمانی که k = 3 یا k = 5 باشد، به ارزش عادلانه بسیار نزدیک به 1 دست می‌یابد.

عملکرد با بارهای کاری مختلط

- زمانی که حجم کار در نقشه و فازهای کاهشی نامتعادل است، FRESH ٪24.47 از زمان ساخت را در مقایسه با FIFO بهبود می بخشد. به طور کلی، FRESH با مجموعه کارهای مختلف به یک عملکرد عالی و پایدار در طول تولید می‌رسد. وقتی تعداد کارهای در حال اجرا همزمان k بیشتر از 1 باشد، عملکرد FRESH از زمانبندی Fair با مقادیر عادلانه بالای 0.95 در همه موارد آزمایش شده بهتر عمل می کند. ما می‌توانیم افزایش عملکرد ثابتی را از FRESH مانند خوشه کوچکتر 10 گره اسلیو مشاهده کنیم. FRESH در مقایسه با زمان‌بندی منصفانه، 31.1% طول زمان ساخت را کاهش می‌دهد. به طور خلاصه، FRESH در هر دو بار کاری ساده و مختلط، بهبود قابل توجهی در ساخت و انصاف ایجاد می کند.

- یکی دیگر از جهت‌گیری‌های مهم برای بهبود عملکرد در Hadoop، زمان‌بندی آگاهانه از منابع است. هدف RAS [22] بهبود استفاده از منابع در سراسر ماشین‌ها و رسیدن به مهلت تکمیل کار است. MROrchestrator [23] رویکردی را برای شناسایی استفاده از منبع وظیفه در هر TaskTracker به عنوان یک مدیر منبع محلی و تخصیص منابع به وظایف در JobTracker به عنوان یک مدیر منبع جهانی معرفی می کند. علاوه بر این، برخی از کارهای دیگر بر روی محیطهای ناهمگن متمرکز شده‌اند. م. زهاریا و همکارانش. یک زمانبندی LATE [35] را برای متوقف کردن اجرای حدس و گمان غیرضروری در یک خوشه Hadoop ناهمگن پیشنهاد می کند. LsPS [36] از الگوهای اندازه کار ناهمگن فعلی برای تنظیم طرح‌های زمان‌بندی استفاده می‌کند.

خلاصه

- این کار بر روی مشکل تخصیص منابع به مراحل مختلف مشاغل متعدد تمرکز دارد. ما Hadoop MapReduce را به عنوان نماینده انتخاب می کنیم. ما یک خوشه Hadoop را مطالعه می کنیم که دسته ای از مشاغل MapReduce را ارائه می دهد. ما مشکلات پیکربندی اسلات و زمان‌بندی کار را هدف قرار می‌دهیم. ما FRESH، یک نسخه پیشرفته از Hadoop را توسعه می‌دهیم که از پیکربندی‌های اسلات استاتیک و پویا پشتیبانی می‌کند. علاوه بر این، ما یک تعریف جدید از عدالت کلی ارائه می دهیم. راه حل ما می تواند در حالی که انصاف را نیز به دست می آورد، makespan خوبی به همراه داشته باشد. ما آزمایش های گسترده ای را با حجم های کاری و تنظیمات مختلف انجام داده ایم. FRESH در مقایسه با یک سیستم Hadoop معمولی، بهبود قابل توجهی را در هر دو جنبه ساخت و انصاف نشان می دهد.