Hadoop 框架下的情报分析大数据调度超时预测方法

蒋苏蓉 蓝江桥 杨玉海
(空军预警学院 武汉 430019)

摘 要 “大数据”是信息科技领域出现的一个研究热点。军用情报数据具备典型的“大数据”特征。Hadoop 是一个基于 java 的分布式密集数据处理和数据分析的软件框架。为了使情报大数据存取满足实时性要求,从军事应用需求出发,对 Hadoop 框架下的情报分析大数据存取调度算法超时现象进行分析研究,实现了一种可以进行超时预测的调度算法。

关键词 Hadoop,HDFS,调度算法,云存储
中图法分类号 TP393 **文献标识码** A

Timeout Prediction of the Schedule Method for Big Data of the Intelligence Analysis Based on Hadoop

JIANG Su-rong LAN Jiang-qiao YANG Yu-hai
(Institute of the Air Force Early Warning, Wuhan 430019, China)

Abstract "big data" is a hot research field of information science and technology. Military intelligence data has typical "big data" features. Hadoop is a software framework based on Java for distributed density data processing and analysis. In order to make the intelligence information and data access meet the real-time requirements, improved the scheduling algorithm under the framework of Hadoop. Through a simple classification way, we realized the real-time intelligence data access request scheduling in a laboratory environment.

Keywords Hadoop, HDFS, Schedule method, Cloud storage

2010 年,我们建设了“云计算军事应用实验室”,并将实验室扩建为国家重点实验室。建设的实验云由 4 部分组成:控制管理中心、计算池、分布式存储池以及交换网络,如图 1 所示。

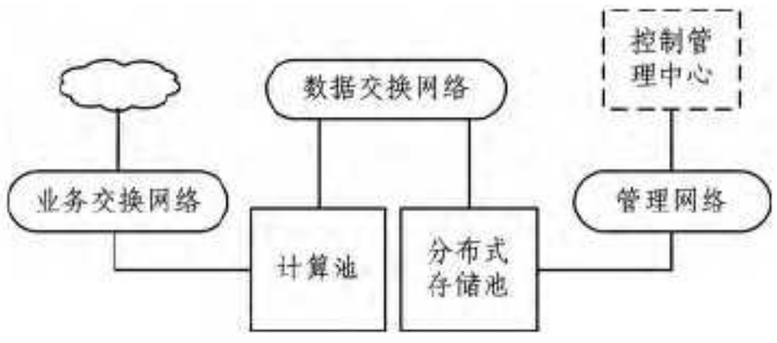

图 1 实验云的拓扑结构

控制管理中心负责对整个云计算数据中心资源进行管理监控,是整个系统的管理端。具体功能包括对基础软硬件进行状态监控和性能监控,对云计算中心运营策略进行设置管理,对基础软硬件异常情况触发报警,提醒用户及时维护问题设备,对基础软硬件资源进行长期的统计分析,为高层次的资源调度提供决策依据。

计算池主要提供对应用透明的计算资源,为应用软件提供按需配置的运行环境。整合并提高物理服务器资源利用率,并保证业务应用的可靠性运行。实现服务器计算资源的按需分配,将虚拟机合理均衡地分配到现有的物理服务器上。实现存储资源的按需分配和弹性伸缩,支撑大规模的存储聚合与数据访问。分存式存储池采用分布式的架构组织底层各种异构的存储资源,形成一个 HDFS 分布式文件系统,用以支撑上层应用尤其是虚拟机磁盘镜像的存储需求。

分存式存储池主要分为主服务器、备份服务器、块服务器和客户端。

主服务器是用于存储元数据的服务器。备份服务器是用于备份元数据的服务器。块服务器是用于存储数据的服务器。客户端是使用分布式文件系统的节点,通过挂载,可以像使用本地文件系统一样使用分布式文件系统。

1 算法基本思想

HDFS 数据存取系统在真正传输数据时,根据所采用的硬件,传输速率可以被认为是恒定的。因此为了提高 HDFS 数据存取系统的吞吐率和带宽利用率,必须对 HDFS 数据存取请求进行有效调度,从而将包括选择 DataNode 节点、发送和接收情报数据在内的整体存取时间降至最短,尽量提高情报数据用户的体验满意度。

为了保证 HDFS 数据存取系统有足够快的响应时间,当一个实时请求到来时,在满足截止期限以及选择最近 DataNode 的情况下,尽量将其插入准备好的队列的后部;当一个非实时请求到来时,在不导致实时请求超时的条件下,尽量选取存放最近的 DataNode,从而使得服务时间最短。

为了实现这一思想,我们将准备好的队列按照请求的 DataNode 号分区。区中请求的 DataNode 号,按照从小到大

本文受空军装备部专项资金资助。
蒋苏蓉(1978—),女,博士生,讲师,主要研究方向为军事情报分析,E-mail:yyhandy@yeah.net。

或从大到小的顺序排序。对于处于同一区中的请求，序列移动方向是一致的。两个相邻的区之间序列移动方向是不一致的，不满足顺序方向，如图 2 所示。

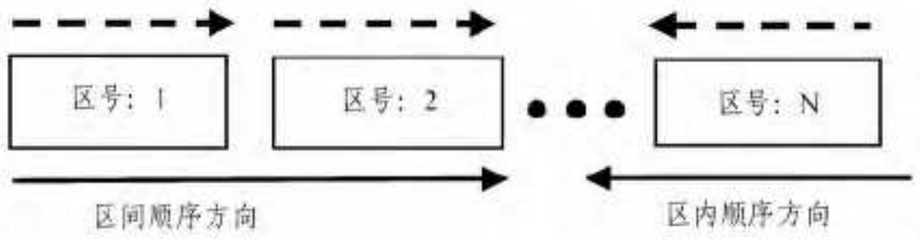

图 2　顺序的位置关系

每个区中采用顺序调度，并且尽量使各区之间也采用顺序调度。当一个非实时请求到达时，为了减少整体响应时间，在不引起实时请求超时的情况下，将该非实时请求插入在准备好的队列的队尾；当一个实时请求到达时，在不会引起该请求超时以及其它实时请求超时的情况下，使其最早开始执行时间接近其最迟执行时间，并且其插入的位置应该符合顺序的位置，将该请求插在准备好的队列的后面，从而使得后来的截止期限小的请求能够有较大的机会在截止期限前完成。

2　数据结构和组织

当将一个请求 T 插入区 P 中（将一个请求 T 从区 P 中删除），在 P 之前的区中请求的服务时间和剩余时间是不变的。而在 P 和 P 之后的区中请求的最早开始服务时间和剩余时间由于新的请求插入导致最早服务时间和剩余时间发生改变。

在区队列中的各个节点记录了对应区的最小剩余时间。因此对于位于 P 之后区中的请求，只需要判断该区的最小剩余时间减去由于该请求的插入而增加的额外服务时间是否小于 0，如果小于 0，说明该请求会引起请求超时；如果大于 0，则说明该请求不会引起请求超时。对于 P 中请求，则根据各自的最早开始执行时间或者剩余时间，判断是否会引起 P 中请求超时。由于最坏情况下，P 等于准备好的队列，因此若采用逐个判断的方法，时间复杂度会比较大。为了解决该问题，每个请求 r 都有一个记录 $min\_st(r)$，用于记录在一个区中，从请求 r 开始到区队列尾请求的最小的剩余时间。这样，当一个请求插入到区 P 中时，只需判断其直接后继的记录 $min\_st(r)$ 减去由于该请求的插入而增加的额外服务时间是否小于 0。

因此判断超时的关键是计算请求的最早开始执行时间、剩余时间和区的最小剩余时间 min_st。

情报数据存取请求的服务时间并不是独立的，与前一个请求是相关的，前一个请求的完成时间决定了后一请求的最早开始执行时间。将准备好的队列分为多个区，可以减少最早开始执行服务时间和剩余时间的计算量。当一个区中只有一个请求时，其 Δ 为 0。而当有请求插入该区或该区之前的区时，Δ 也随之变化。

以 $Tprevious$ 表示准备好队列中 T 的直接前驱，$Tsuccesor$ 表示准备好队列中 T 的直接后继，并且 $Tprevious$ 的 $DataNode$ 号为 $C1$，$Tsuccesor$ 的 $DataNode$ 号为 $C2$，T 的 $DataNode$ 号为 $C3$。

以 $Ppreviousi(i=1,2,\cdots,m)$ 表示区队列中 P 的前驱，$Psuccesorj(j=1,2,\cdots,n)$ 表示区队列中 P 的后继，其中 $Ppervious1$ 表示位于 P 之前的区队列中第一个区，$Pperviousm$ 表示 P 的直接前驱，$Psuccessor1$ 表示 P 的直接后继，$Psuccessorn$ 表示在区队列中位于 P 之后的最后一个区。

以 r 表示 P 中位于 T 之后的一个请求，q 表示 P 中位于 T 之前的一个请求。

T 的插入和删除都会引起请求的最早开始执行时间和剩余时间的变化。在准备好的队列中，T 的前驱请求的最早开始执行时间、完成时间以及剩余时间都不会发生变化；T 的后继请求的最早开始时间、完成时间以及剩余时间都会由于 T 的插入（或删除）而会发生变化。

根据 T 的插入和删除，超时判断分为两种情况。

2.1　插入请求

插入请求 T，其后继请求的最早开始执行时间将推迟。以 et 表示由于该请求 T 的插入其后继请求的推迟执行时间。

$Tprevious$、T、$Tsuccesor$ 的位置关系有如图 3 所示的 6 种基本情况。图 3 中带箭头的实线表示序列的移动方向，例如在图 3(a) 中，序列先从 $C1$ $DataNode$ 移动到 $C3$ $DataNode$，服务完请求 T 后，从 $C3$ $DataNode$ 移动到 $C2$ $DataNode$，再服务请求 $Tsuccesor$。

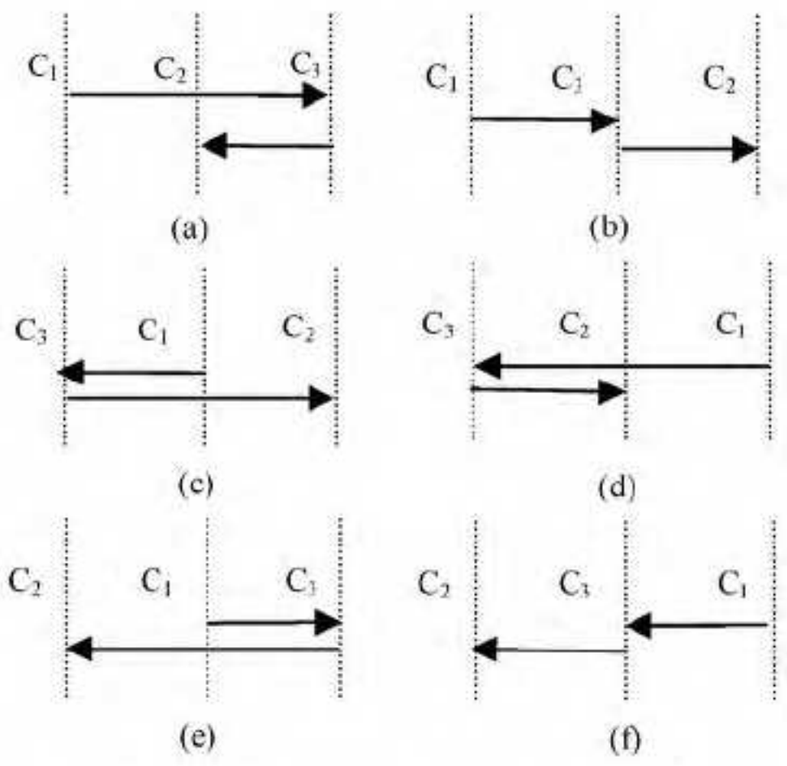

图 3　$Tprevious$、T、$Tsuccesor$ 的位置关系

设在没有插入 T 时，$Tsuccessor$ 的剩余时间为 ST，最早开始执行时间为 $EST(Tsuccessor)$，则 $EST(T)=EFT(Tprevious)$。

推后时间 et 的计算方法如下：

插入 T 之后，T 的最早开始时间为 $EST(T)=EFT(Tprevious)$。当 T 为实时请求时，T 的剩余时间 $ST=dT-EFT(T)$；T 为非实时请求时，ST 取系统允许的最大正整数。设 $Tsuccesor$ 的最早开始执行时间和剩余时间分别为 $EST(Tsuccesor)$ 和 $STsuccessor$，插入 T 之后，$Tsuccesor$ 的最早开始执行时间为 $EST(Tsuccesor)'$ 和 $STsuccessor'$，其分别等于 $EST(Tsuccesor)'=EFT(T)$，$STsuccessor'=dTsueccessor-EFT(Tsuccesor)'$。

则推迟时间 $et=EST(Tsuccesor)'-EST(Tsuccesor)=STsuccessor-STsuccessor'$。$et$ 的计算与其直接前驱 $Tprevious$、直接后继请求 $Tsuccesor$ 和 T 的相对位置有关。

1) $DataNode$ 关系如图 3(a) 所示，3 个请求的 $DataNode$ 关系为 $C1\leqslant C2\leqslant C3$。此时，序列从 $C1DataNode$ 经过 $C2DataNode$ 到 $C3DataNode$，服务完 T 后，序列回到 $C2DataNode$，再服务 $Tsuccesor$。$Tsuccesor$ 的最早开始执行时间将推迟 $2|C3-C2|*DF+cT$，即 $et=2|C3-C2|*DF+cT$。

2)$DataNode$ 关系如图 3(b)所示，3 个请求的 $DataNode$ 关系为 $C1 \leqslant C3 \leqslant C2$。此时，由于 T 的插入而引起 $Tsuccesor$ 的最早开始执行时间推后，推后时间为 $et=cT$。

3)$DataNode$ 关系如图 3(c)所示，3 个请求的 $DataNode$ 关系为 $C3 \leqslant C1 \leqslant C2$。此时，由于 T 的插入而引起 $Tsuccesor$ 的最早开始执行时间推后，推后时间为 $et=2|C3-C1| * DF+cT$。

4)$DataNode$ 关系如图 3(d)所示，3 个请求的 $DataNode$ 关系为 $C3 \leqslant C2 \leqslant C1$。此时，由于 T 的插入而引起 $Tsuccesor$ 的最早开始执行时间推后，推后时间为 $et=2|C3-C2| * DF+cT$。

5)$DataNode$ 关系如图 3(e)所示，3 个请求的 $DataNode$ 关系为 $C3 \leqslant C2 \leqslant C1$。此时，由于 T 的插入而引起 $Tsuccesor$ 的最早开始执行时间推后，推后时间为 $et=2|C3-C1| * DF+cT$。

6)$DataNode$ 关系如图 3(f)所示，3 个请求的 $DataNode$ 关系为 $C2 \leqslant C3 \leqslant C1$。此时，由于 T 的插入而引起 $Tsuccesor$ 的最早开始执行时间推后，推后时间为 $et=cT$。

可以总结出如下规律：当 $C1 \leqslant C3 \leqslant C2$ 或 $C2 \leqslant C3 \leqslant C1$ 时，$et=cT$。当 $C1 \leqslant C2 \leqslant C3$ 或 $C3 \leqslant C2 \leqslant C1$ 时，$et=2|C3-C2| * DF+cT$。当 $C3 \leqslant C2 \leqslant C1$ 或 $C3 \leqslant C1 \leqslant C2$ 时，$et=2|C3-C1| * DF+cT$。

T 的插入(删除)并不影响在 $Pprevious$ 中的请求的最早开始执行时间和剩余时间，因此只考虑 P 和 $Psuccessorj(j=1,2,\cdots,n)$ 中的请求。

T 的插入将引起区 $Psuccesorj(j=1,2,\cdots,n)$ 中请求推迟其最早开始执行时间。

$Psuccesorj$ 中请求超时判断方法：

若 $min\text{-}st(Psuccessorj)-et<0$，则说明有请求超时，不能将 T 插入到 $Psuccesorj$ 之前；否则可以将 T 插入到 $Psuccesorj$ 之前。

P 中请求超时判断方法：

若 $min\text{-}st(Tsuccesor)-(\Delta(P)+et)<0$ 或 $ST<0$，则说明 P 中有请求超时，不能将 T 插入到 P 中；否则可以将 T 插入到 P 中。

T 插入之后，需完成以下工作：

修改 $\Delta(Psuccesorj)$、$min\text{-}st(Psuccesorj)$ 以及 $\Delta(P)$、$min\text{-}st(P)$ 和 P 中请求 r 的最早开始执行时间和剩余时间以及 $min\text{-}st(r)$。

$\Delta(Psuccesorj)$、$min\text{-}st(Psuccesorj)$ 的修改：

$\Delta(Psuccessorj)=\Delta(Psuccessorj)+et$；

$min\text{-}st(Psuccessorj)=min\text{-}st(Psuccessorj)-et$。

由于判断位于 P 后继区中的实时请求是否会超时，只需要判断该请求所在区的最小剩余时间是否小于零。此时并不修改在 P 的后继区中请求的最早开始执行时间和剩余时间。只有当区中有请求插入(或删除)时，才修改该区中请求的最早开始执行时间、剩余时间、$min\text{-}st(r)$ 和该区的 $min\text{-}st(P)$。

$\Delta(P)$ 和 $min\text{-}st(P)$ 以及 P 中请求 r 的最早开始执行时间和剩余时间的修改以及 $min\text{-}st(r)$ 修改方法如下：

T 在一个新建区 P 中，则 $\Delta(P)=0$，$min\text{-}st(P)=ST$，$EST(T)=EFT(Tprevious)$。

当 T 在一个已存在区 P 中时，需修改 P 中请求的最早开始执行时间和剩余时间。修改步骤有如下 4 步：

1)$min\text{-}st(P)$ 取 $min\text{-}st(P)-et$ 与 ST 的最小值，即 $min\text{-}st(P)=min(min\text{-}st(P)-et,ST)$。

2)修改在 P 中 T 所有后继的最早开始执行时间和剩余时间。在 P 中 T 的一个后继 r 的最早开始执行时间、剩余时间、$min\text{-}st(r)$ 分别修改为：$EST(r)=EST(r)+(\Delta(P)+et)$；$Sr=Sr-(\Delta(P)+et)$；$min\text{-}st(r)=min\text{-}st(r)-(\Delta(P)+et)$。

3)修改在 P 中 T 所有前驱的最早开始执行时间和剩余时间。在 P 中 T 的一个前驱 q 的最早开始执行时间修改为：$EST(q)=EST(q)+\Delta(P)$；q 的剩余时间修改为 $Sq=Sq-\Delta(P)$。

4)令 $\Delta(P)=0$。

2.2 删除请求

删除一个请求，其后继请求将会提前执行。以 lt 表示由于该请求 T 的删除其后继请求的提前执行时间。

lt 的计算：

与插入请求类似，lt 也与其前驱请求 $Tprevious$、后继请求 $Tsuccesor$ 和 T 的相对位置有关，如图 4 所示。图 4 中带箭头的实线表示序列的移动轨迹，而虚线表示在未删除 T 时，序列的移动轨迹。

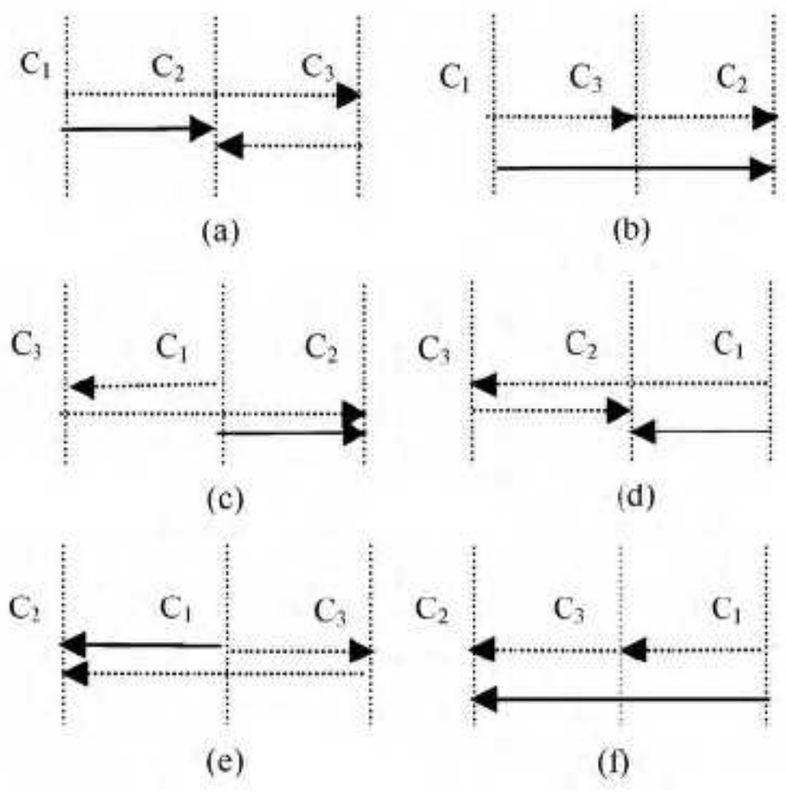

图 4 $DataNode$ 关系图

1)$DataNode$ 关系如图 4(a)所示，3 个请求的 $DataNode$ 关系为 $C1 \leqslant C2 \leqslant C3$。当未删除 T 时，序列先从 $C1$ $DataNode$ 移动到 $C3$ $DataNode$，服务完请求 T 后，从 $C3$ $DataNode$ 移动到 $C2$ $DataNode$，再服务请求 $Tsuccesor$。删除 T 之后，序列服务完 $Tprevious$，直接从 $C1$ 移动到 $C2$，因此删除 T 之后，后继请求能够提前执行的时间为 $lt=2|C3-C2| * DF+cT$。

2)$DataNode$ 关系如图 4(b)所示，3 个请求的 $DataNode$ 关系为 $C1 \leqslant C3 \leqslant C2$。删除 T 后，后继请求能提前执行的时间为 $lt=cT$。

3)$DataNode$ 关系如图 4(c)所示，3 个请求的 $DataNode$ 关系为 $C3 \leqslant C1 \leqslant C2$。删除 T 后，后继请求能提前执行的时间为 $lt=2|C3-C1| * DF+cT$。

4)$DataNode$ 关系如图 4(d)所示，3 个请求的 $DataNode$ 关系为 $C3 \leqslant C2 \leqslant C1$。删除 T 后，后继请求能提前执行的时间为 $lt=2|C3-C2| * DF+cT$。

5)$DataNode$ 关系如图 4(e)所示，3 个请求的 $DataNode$ 关系为 $C2 \leqslant C1 \leqslant C3$。删除 T 后，后继请求能提前执行的时

间为 $lt=2|C3-C1|*DF+cT$。

6)$DataNode$ 关系如图 4(f)所示，3 个请求的 $DataNode$ 关系为 $C2\leqslant C3\leqslant C1$。删除 T 后，后继请求能提前执行的时间为 $lt=cT$。

可以总结出如下规律：当 $C1\leqslant C3\leqslant C2$ 或 $C2\leqslant C3\leqslant C1$ 时，$lt=cT$。当 $C1\leqslant C2\leqslant C3$ 或 $C3\leqslant C2\leqslant C1$ 时，$lt=2|C3-C2|*DF+cT$。当 $C2\leqslant C1\leqslant C3$ 或 $C3\leqslant C1\leqslant C2$ 时，$lt=2|C3-C1|*DF+cT$。

删除 T 之后将引起区 $Psuccesorj(j=1,2,\cdots,n)$ 中请求提前执行。

请求超时判断：

若 $min\text{-}st(Psuccessorj)+lt<0$（或者 $min\text{-}st(r)-(\Delta(P)-lt)<0$），则说明有请求超时，也即删除 T 仍不能满足实时请求的时限要求，则需要继续删除请求，直到 $min\text{-}st(Psuccessorj)+lt\geqslant 0$（或者 $min\text{-}st(P)+lt<0$）。

T 删除之后，需修改 $\Delta(Psuccesorj)$、$min\text{-}st(Psuccesorj)$ 以及 $\Delta(P)$、$min\text{-}st(P)$ 和 P 中请求 r 的最早开始执行时间和剩余时间以及 $min\text{-}st(r)$。

$\Delta(Psuccessorj)$ 修改为 $\Delta(Psuccessorj)-lt$；

$min\text{-}st(Psuccessorj)$ 修改为 $min\text{-}st(Psuccessorj)+lt$。

$min\text{-}st(P)$、$\Delta(P)$ 的修改以及 P 中请求 r 最早开始执行时间和剩余时间以及 $min\text{-}st(r)$ 的修改步骤有如下 4 步：

1)$min\text{-}st(P)$ 取 $min\text{-}st(P)+lt$ 与 ST 的最小值，即 $min\text{-}st(P)=min(min\text{-}st(P)+lt,ST)$。

2)修改在 P 中 T 所有后继的最早开始执行时间和剩余时间。在 P 中 T 的一个后继 r 的最早开始执行时间和剩余时间以及 $min\text{-}st(r)$ 分别修改为：$EST(r)=EST(r)+(\Delta(P)-lt)$；$Sr=Sr-(\Delta(P)-lt)$；$min\text{-}st(r)=min\text{-}st(r)-(\Delta(P)-lt)$。

3)修改在 P 中 T 所有前驱的最早开始执行时间和剩余时间。在 P 中 T 的一个前驱 q 的最早开始执行时间和剩余时间分别修改为：$EST(q)=EST(q)+\Delta(P)$；剩余时间 $Sq=Sq-\Delta(P)$。

4)令 $\Delta(P)=0$。

2.3 超时判断

当将一个请求 T 插入区 P 中（或将一个请求 T 从区 P 中删除），在 P 之前的区中请求的服务时间和剩余时间是不变的，只需判断在 P 和 P 之后的区中请求是否超时。

判断方法如下：判断 $min\text{-}st(Psuccessorj)-et$ 是否小于零，判断 $min\text{-}st(r)-(\Delta(P)+et)$ 是否小于零以及 ST 是否小于零即可，而不需逐个判断 T 之后的每个请求剩余时间都小于零。从而减小了计算量。算法如下：

```
If min-st(r)-(Δ(P)+et)<0 或 ST<0 then return 1
Psuccessor=p
While Psuccessor()NULL
begin
    If min-st(Psuccessor)-et<0 then return 1
    Psuccessor= Psuccessor->next
End
Return 0
```

3 模拟模型

在模拟模型中，单位时间内的情报数据存取请求到达个数服从泊松分布，每个请求之间是相互独立的。情报数据所在的 $DataNode$ 号 C 是一个在 $[1,MC]$ 上均匀分布的随机变量，MC 是最大 $DataNode$ 号，实验中 $MC=3000$。每个请求的响应时间包括在调度队列中的等待时间和情报数据存取的服务时间。其中服务时间的计算公式采用 $access(n)=[DF*sqrt(n)+cT]$ms，其中 n 为序列移动的距离，DF 为序列服务因子，cT 为平均数据传输时间。这里设置 $DF=0.88$，$cT=9.66$。实验中产生的请求个数为 80000。

截止时间由公式定义，

$eadline=arrival\text{-}time+access(n)+slack\text{-}time$。

其中，$arrival\text{-}time$ 是情报数据存取请求的到达时间，$slack\text{-}time$ 是松弛时间；设松弛时间是一个在 $[min\text{-}slack,max\text{-}slack]$ 上均匀分布的随机变量。

这里假设 $min\text{-}slack=10$ms，$max\text{-}slack=45$ms。

令 $RT(r)$ 表示系统中非实时任务的平均响应时间，r 是实时请求占整个请求的百分比。

4 性能评测

《基于 Hadoop 的大数据情报分析调度方法》一文中提出了一种调度方法 JSR。下面对这种方法的性能进行模拟实验，并给出实验结果。

4.1 实时请求超时率实验

本实验的目的是研究 HDFS 系统中增加实时性后，实时请求的响应情况，即系统的实时性。模拟实验中，r 分别取值 0.95，0.75，0.50，0.25 和 0.05。

图 5 描绘了 r 取值为 0.25 时的请求超时率。在请求的到达率较低的情况下，所有算法的请求超时率没有很明显的差别。随着请求到达率的逐渐增大，FIFO 算法请求超时率比 EDF 算法的请求超时率略高，JSR 在各种情况下都有较好的性能。在请求的到达率超过 200 时，JSR 算法的请求超时率比 FIFO 算法和 EDF 算法分别减少 50% 和 30%。JSR 算法在实时性能上是较好的。

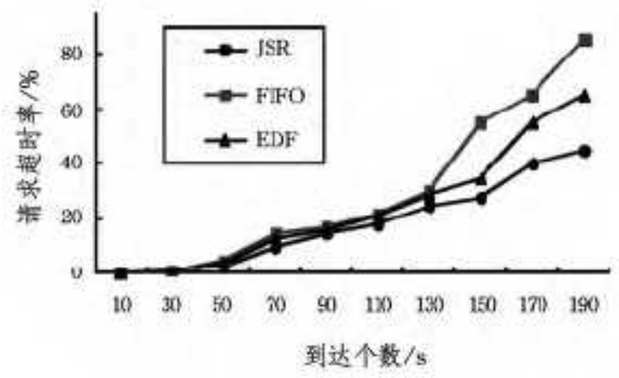

图 5 I/O 请求到达率和请求超时率的关系

4.2 非实时请求响应时间实验

本实验的目的是考察系统对非实时请求的响应情况。由于采用了多列队结构，响应时间可能会高于单队列结构的算法。因此，本实验中对比算法分别是多队列 FIFO 算法和多队列 EDF 算法。图 6 显示了在 $r=0.8$ 的情况下，JSR 算法、多队列 FIFO 算法和多队列 EDF 算法的平均响应时间 $RT(r)$。在请求的到达率较低的情况下，算法的响应时间值较低。随着请求到达率的增加，响应时间也增加，但大体保持一个平滑上升的态势。请求的到达率较高时，算法的性能仍然在一个可以接受的范围内。这说明采用 JSR 算法，在实时请求数较多时，系统依然有着良好的响应时间指标。

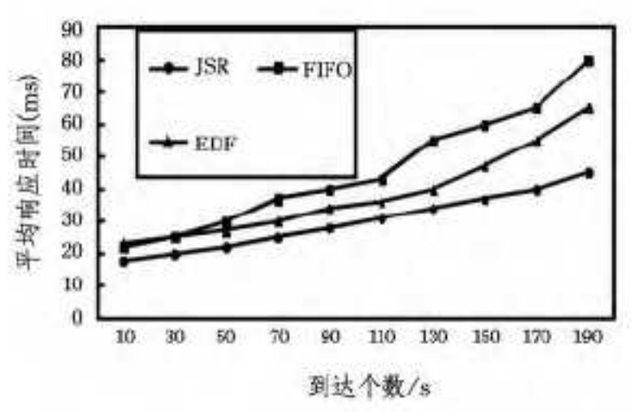

图 6　3 种算法的响应时间对比

结束语　在 Hadoop 框架中，研究为 HDFS 系统增加实时情报数据存取功能的方法，进而提出了一种新的情报数据存取调度方法 JSR。对 DataNode 访问请求进行分区管理，分析了各种情况下 DataNode 的分区访问特性，给出了一种实时请求的超时预测方法。模拟实验表明其平均访问时间和实时请求超时率与已有算法相比，性能较高。在请求到达率取值超过 190 时，JSR 算法的请求超时率比多队列 FIFO 算法和多队列 EDF 算法分别减少 50% 和 30%。JSR 算法响应时间指标比 FIFO 算法和 EDF 算法分别减少 27% 和 40%。

参 考 文 献

[1] 于薇．“大数据”背景下的信息处理技术分析与研究[OL]．http://www.dlf.net.cn

[2] 杨玉海，等．异构实时多处理器系统的动态调度算法研究[J]．小型微型计算机系统，2008，29(1)：130-134

[3] 熊家军，杨玉海，蒋苏蓉，等．机载预警雷达情报分析技术及应用[M]．北京：军事谊文出版社，2013

[4] 李建锋，彭舰．云计算环境下基于改进遗传算法的任务调度算法[J]．计算机应用，2011，31(1)：184

[5] 赵春燕．云环境下作业调度算法研究与实现[M]．北京：北京交通大学出版社，2009

[6] 苑迎春，李小平，王茜，等．基于逆向分层的网格工作流调度算法[J]．计算机学报，2008，31(2)：282

[7] 启红超，伊鹏，郭云飞．一种公平服务的动态轮询调度算法[J]．软件学报，2008，19(7)：1856

(上接第 393 页)

成合成数据流，主要参数有：混合模型 $\Lambda_Y=\{(\Lambda_{\alpha 1},\theta_1),\cdots,(\Lambda_{\alpha J},\theta_J)\}$，数据流长度 T，噪音概率 ρ，事件类型字母表 ε 的大小 M，被嵌入的模型大小 N，被嵌入的模型数量 J，EHMM 噪音参数 $\eta_{\alpha j}$，以及每个 $\Lambda_{\alpha j}$ 的混合系数 θ_j。

实验 1 研究前窗长度 W 对预测性能的影响。合成数据生成的参数：$J=10$，$N=5$，$\rho=0.9$，$T=100000$，$M=55$，EHMM 噪音参数 $\eta_{\alpha j}=0.5$，混合比例是随机的。W 在 2 至 16 之间变化。如图 2 所示窗口的大小使得性能恶化的趋势是逐渐减小的

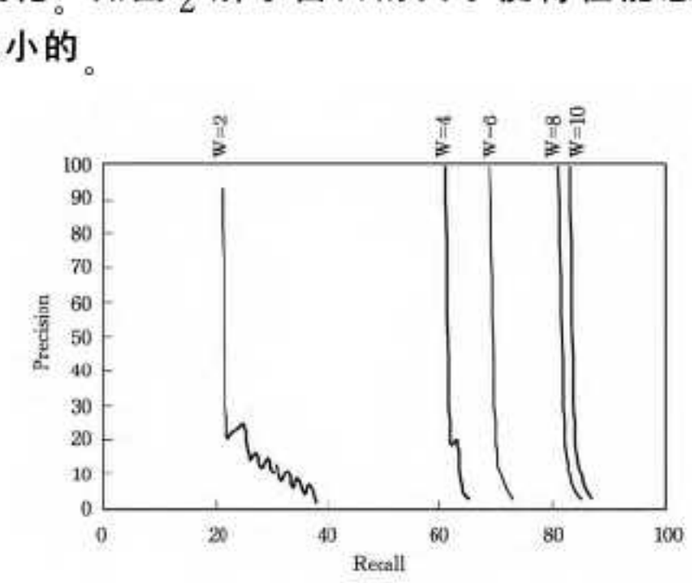

图 2　前窗的长度 W 对预测性能的影响

实验 2 研究混合模型中的模型数量对预测性能的影响。合成数据生成的参数：$N=5$，$\rho=0.9$，$T=100000$，$M=55$，噪音参数 $\eta_{\alpha j}=0.5$，混合系数 $=1/J$，$W=8$，J 在 10 至 1000 中变化。如图 3 所示，当模型数量较小时，预测性能较稳定；当模型数量增加时，它们的频率下降，导致前窗更像是随机序列，从而恶化了预测性能。

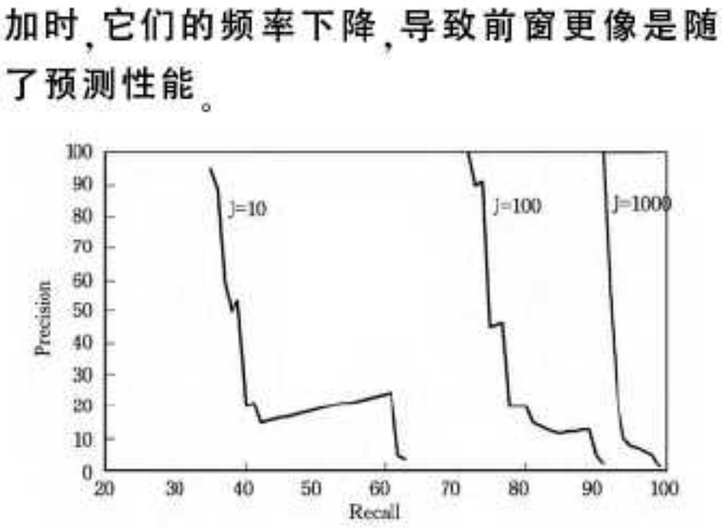

图 3　模型数量对预测性能的影响

结束语　本文提出的改进算法 NONEPI++ 用于挖掘事件序列上非重叠发生的频繁情节，算法通过时间戳计算来产生新的频繁情节，避免了重复扫描事件序列，提高了情节挖掘的效率。同时，给出了基于频繁情节实现数据流预测的步骤：第一步将挖掘的每个频繁情节表示为一个隐马尔可夫模型 EHMM，并基于历史数据和 EM 算法构造由多个 EHMM 组成的混合 EHMM；第二步基于生成的混合 EHMM 模型，预测给定窗口长度 W 最近一次非重叠发生的目标事件。实验表明，混合 EHMM 模型能有效地预测数据流。

参 考 文 献

[1] Manilla H, Toivonen H, Verkamo A. Discovering frequent episodes in sequences[C]// Proceedings of the First International Conference on Knowledge Discovery and Data Mining. 1995：210-215

[2] Mannila H, Toivonen H, Verkamo A I. Discovery of frequent episodes in event sequences[J]. Data Mining and Knowledge Discovery, 1997, 1(3)：259-289

[3] Mannila H, Toivonen H. Discovering Generalized Episodes Using Minimal Occurrences[C]// KDD. 1996：146-151

[4] Laxman S, Sastry P S, Unnikrishnan K P. Discovering frequent episodes and learning hidden markov models：A formal connection[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(11)：1505-1517

[5] Fletcher A K, Rangan S, Goyal V K. Estimation from lossy sensor data：Jump linear modeling and Kalman filtering[C]// Proceedings of the 3rd international symposium on Information processing in sensor networks. ACM, 2004：251-258

[6] Cho C W, Zheng Y, Wu Y H, et al. A tree-based approach for event prediction using episode rules over event streams[C]// Database and Expert Systems Applications. Berlin Heidelberg：Springer, 2008：225-240

[7] 朱辉生，汪卫，施伯乐．基于情节规则匹配的数据流预测[J]．软件学报，2012，23(5)：151-162

[8] 王志超，刘惠义．一种基于隐马尔可夫模型的人脸识别方法[J]．计算机应用与软件，2013，30(2)：304-307

[9] 肖献强，任春燕，王其东．基于隐马尔可夫模型的驾驶行为预测方法研究[J]．中国机械工程，2013，24(21)：2972-2976

[10] 闫新娟，谭敏生，严亚周，等．基于隐马尔科夫模型和神经网络的入侵检测研究[J]．计算机应用与软件，2012，29(2)：294-297

[11] 吕岸，胡振程，陈慧．基于高斯混合隐马尔科夫模型的高速公路超车行为辨识与分析[J]．汽车工程，2010(7)：630-634