第 46 卷 第 1 期 计 算 机 工 程 2020 年 1 月
Vol. 46 No. 1 Computer Engineering January 2020

·体系结构与软件技术· 文章编号：1000-3428(2020)01-0187-09 文献标志码：A 中图分类号：TP319

大规模异构计算集群的双层作业调度系统

孙震宇[1,2]，石京燕[1]，孙功星[1]，杜 然[1]，姜晓巍[1]，邹佳恒[1]，谭宏楠[1,2]

（1. 中国科学院高能物理研究所，北京 100049；2. 中国科学院大学 物理科学学院，北京 100049）

摘 要： 高能物理计算平台中的 HTCondor 和 SLURM 计算集群为多个高能物理实验提供数据处理服务，然而 HTCondor 并行作业调度效率较低、SLURM 难以应对大量串行作业，且计算平台整体资源管理及调度策略过于简单。为满足高能物理计算集群高负荷运行的需求，在传统作业调度器上增加作业管理层，设计双层作业调度系统，通过高效调度串并行作业并兼顾实验组间资源的使用公平性，实现用户对作业的细粒度管理。测试结果表明，双层作业调度系统支持大批量高能物理作业的快速提交，并充分利用计算平台的总体资源，具有较好的作业调度性能。

关键词： 计算集群管理；作业调度器；高通量计算；高性能计算；高能物理计算

开放科学（资源服务）标志码（OSID）：

中文引用格式： 孙震宇，石京燕，孙功星，等. 大规模异构计算集群的双层作业调度系统[J]. 计算机工程，2020，46(1)：187-195.

英文引用格式： SUN Zhenyu，SHI Jingyan，SUN Gongxing，et al. Dual layer job scheduling system for large scale heterogeneous computing clusters[J]. Computer Engineering，2020，46(1)：187-195.

Dual Layer Job Scheduling System for Large Scale Heterogeneous Computing Clusters

SUN Zhenyu[1,2]，SHI Jingyan[1]，SUN Gongxing[1]，DU Ran[1]，JIANG Xiaowei[1]，ZOU Jiaheng[1]，TAN Hongnan[1,2]

（1. Institute of High Energy Physics，Chinese Academy of Sciences，Beijing 100049，China；
2. School of Physical Sciences，University of Chinese Academy of Sciences，Beijing 100049，China）

【Abstract】 HTCondor and SLURM computing clusters in high-energy physics computing platforms provide data processing services for many high-energy physics experiments. However，HTCondor is not efficient in parallel job scheduling，and SLURM could not manage massive serial jobs. Also，the overall resource management and scheduling strategies of computing platforms are too simple. To meet the demands of high-energy physics computing clusters running with heavy duties，this paper designs a dual layer job scheduling system，which adds a job management layer on the existing job scheduler. The system is designed to efficiently schedule serial and parallel jobs，ensure fair use of resources between experiment groups，and enable users to implement fine-grained management of jobs. Test results show that the dual layer job scheduling system supports rapid submission of massive high-energy physics jobs，makes full use of resources of a computing platform，and has a high performance in job scheduling.

【Key words】 computing cluster management；job scheduler；High Throughput Computing(HTC)；High Performance Computing(HPC)；high energy physics computing

DOI： 10.19678/j.issn.1000-3428.0053582

0 概述

高能物理是研究微观物质世界的基本粒子组成及其相互作用的学科。现代高能物理实验产生海量实验数据，巨大的实验数据量可以提高物理结果的统计精度，有助于寻找稀有反应过程。高能物理数据处理主要通过在计算集群上运行大量作业，每个作业分析处理一个或一批数据文件。目前高能物理计算软件框架仍以串行计算为主，通过同时运行大量作业，分别处理不同数据文件达到并行处理的效

基金项目： 国家自然科学基金(11475210)；国家自然科学基金青年基金(11805225)。

作者简介： 孙震宇(1990—)，男，博士研究生，主研方向为计算集群作业调度；石京燕，副研究员、博士；孙功星，研究员、博士生导师；杜 然，助理研究员、博士；姜晓巍，工程师、硕士；邹佳恒，副研究员、博士；谭宏楠，硕士。

收稿日期： 2019-01-07 **修回日期：** 2019-04-09 **E-mail：** sunzy@ihep.ac.cn

计算机工程 www.ecice06.com

果。高能物理计算集群管理及作业调度软件以集中式作业调度系统为主,其中常用的开源作业调度软件包括 PBS[1]、SLURM[2]和 HTCondor[3]等。

中国科学院高能物理研究所计算中心运行并管理着一套本地高通量计算(High Throughput Computing,HTC)集群,以及一套高性能计算(High Performance Computing,HPC)集群。HTC 集群具有 13 000 个 CPU 核心,通过 HTCondor 实现资源管理以及作业调度,主要用于处理北京谱仪[4]、大亚湾中微子实验[5]、羊八井宇宙线实验[6]、江门中微子实验[7]、高海拔宇宙线观测站[8]等大型高能物理实验的串行计算作业。HPC 集群具有 4 000 个 CPU 核心、80 块 Tesla V100 GPU 卡,采用 SLURM 实现资源管理以及作业调度,主要用于处理 BES 分波分析、JUNO、格点量子色动力学、环形正负电子对撞机、高能同步辐射光源等实验的并行计算作业。

由于各个并行作业的并行度大小不一,目前计算中心的 HPC 集群的整体资源利用率仅为 50% 左右,而 HTC 集群利用率长期处于饱和状态且有大量作业排队等待,因此在不影响并行作业正常调度的前提下,迁移 HTC 作业到 HPC 集群上运行是计算中心需要解决的一个重要问题。

本文设计并开发双层作业调度系统以及配套的用户作业管理工具包。双层调度系统可以快速接收海量的串行或并行作业,以插件形式支持多种调度策略,实现对资源和作业的细粒度管理,并且通过多个作业队列和特定作业调度算法保证各实验组对计算资源的公平使用。

1 相关研究

高能所计算中心自 2001 年开始一直使用开源软件 TORQUE[9]管理集群资源,并借助 Maui[10]实现作业调度。高能所 TORQUE 计算集群中的计算节点分属于不同高能物理实验,隶属于各实验的计算节点仅运行本实验的计算作业,各实验有自己的专用作业队列。为保证实验内各用户之间的公平性,调度器还对单用户可以同时运行的最大作业数进行限制。然而在实际运行过程中,由于各实验对计算需求的峰谷时段不同,计算集群的整体资源利用率仅为 50%~60% 。此外,TORQUE/Maui 还存在对 GPU 计算的支持不足、作业数量过大时调度性能下降等问题。

高能所计算中心自 2015 年开始逐步将计算资源从 TORQUE/Maui 集群向 HTCondor 集群迁移。HTCondor[3]基于分类广告(ClassAd)及匹配操作的调度模式[11]使其可以高效地调度大量高通量计算作业。文献[12]展示了其调度管理 16 000 个作业同时运行的能力。因此,HTCondor 被广泛应用于高能物理计算[13]、地理信息系统[14]、遥感数据处理[15-17]等领域。高能所计算中心将所有实验计算资源组成共享资源池,供各个实验错峰共享使用,这使得计算集群的整体资源利用率提高到 90% 以上。然而,HTCondor 调度算法单一,且没有作业队列的概念,难以提供类似 PBS 的资源细粒度分配和高级作业管理功能。此外,HTCondor 架构不具备中心管理功能,难以做到对计算节点的统一管理。另外,HTCondor 对 MPI 并行作业的调度效率较低,在串并行作业混合排队时,该问题表现得更加突出。

随着科学技术的发展,高能物理数据处理正在由单进程处理向并行计算模式转换。为此,计算中心建立了 HPC 计算集群,采用 SLURM 实现资源管理以及作业调度。SLURM 具有大量的调度算法,对于并行作业的资源分配有较高的效率,因此被应用于天河 2[18]超级计算机以及深度学习[19]、计算机辅助工程[20]等典型的高性能计算场景中。但经测试,SLURM 在处理大量 HTC 短作业时调度效率下降严重,无法应对万级 CPU 核计算集群上的万级串行作业资源分配。因此,计算中心无法借助单一的开源作业调度软件来管理整个计算集群,形成了 HTC 及 HPC 作业各有一个计算集群的现状。

文献[21]着手解决将 HTC 作业迁移到 HPC 集群上运行的问题,但其提出的解决方案需要用户设置作业属性以允许作业迁移,对用户不完全透明。文献[22]中的 CMS 实验选用 GlideinWMS[22]作为其计算资源供应层,将分布在世界各地的多级站点组成一个计算资源池。CMS 预计其 Run 2 期间需要约 20 万 CPU 核的计算规模[23],并通过调度器并行化、单核作业转换为多核作业等手段实现了对 20 万 CPU 核的有效管理[24]。然而上述工作并未涉及到基于计算资源拥有份额的公平性问题以及对计算节点的统一管理问题。文献[25]实现了类似 GlideinWMS 的作业排队、Agent 分发以及作业拉取机制。文献[26]使站点通过 Mesos[27]提供计算资源给 HTCondor 集群,但两者同样侧重于对异地计算资源的利用。

2 双层作业调度系统架构设计

高能所计算平台对计算作业调度的需求为:

1)提供用户单一作业管理功能,即用户通过单一作业工具提交、查询、管理自己的串并行作业。

2)保证大并行度作业的资源有效分配,同时实现对大规模 HTC 作业的高效调度。

3)提供多种作业队列,以满足不同实验用户的多种计算需求。

4)保证实验组间资源使用公平性,即各个实验组使用的资源数量与其拥有的资源数量比例相当。

5)充分利用空闲计算资源,提高整体资源利用率,降低作业排队时间。

计算机工程 www.ecice.com

本文设计的双层作业调度系统采用分治的思想,将本地的大规模异构计算集群拆分为一个或多个同构或异构子集群,每个子集群分别由 HTCondor 或 SLURM 等作业调度软件管理。同时,本文在子集群的基础上增加了一个作业管理层,用来实现作业队列设置以及调度算法的扩展,同时可兼顾用户和队列的优先级,以及 HPC 作业对计算资源的特殊需求。双层作业调度系统充分利用 HTCondor 自有的高效调度性能以及 SLURM 对于并行作业管理的优势,弥补了 HTCondor 调度算法简单、功能不全以及 SLURM 面对大量排队作业性能下降的问题。

双层作业调度系统整体架构如图 1 所示,由作业管理层和作业执行层组成。上层为作业管理层,包括作业管理器、资源状态收集模块、作业调度器、作业提交模块、作业监视模块等,负责向用户提供统一作业管理接口,建立作业调度队列以及实现作业一级调度。作业管理器接收用户提交的计算作业,并维护作业队列。作业调度器根据系统管理员指定的调度算法,将用户作业在合适的时间提交到作业执行层。作业调度算法保证作业一旦被提交到作业执行层,将被立即执行。作业执行层上长期保持没有或只有少量作业排队的状态。

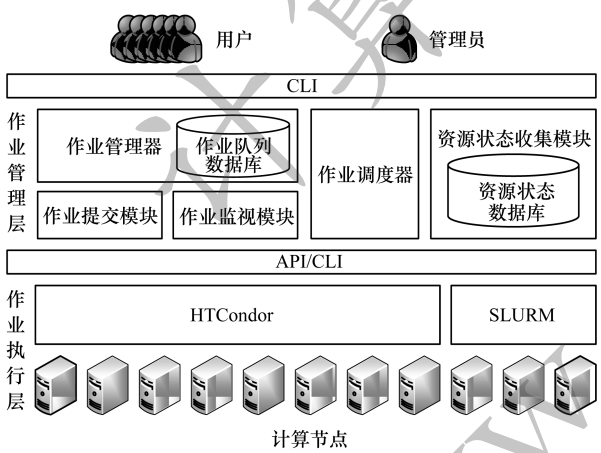

图 1　双层作业调度系统整体架构

Fig. 1　Overall architecture of the dual layer job scheduling system

下层为作业执行层,由一个或多个 HTCondor、SLURM 等作业调度软件构成,分别管理计算集群内的一部分计算资源,相当于在逻辑上将整个计算集群分割为一个或多个子集群。作业执行层主要负责作业在计算节点上的启动、状态监视、文件输入/输出等工作。

用户提交作业到作业管理器,作业信息被存储到作业管理器的数据库中。资源状态收集模块定期采集作业执行层的各个子集群的资源信息,并存储于资源状态数据库。作业调度器根据设定的作业调度算法,为排队作业计算优先级,再按优先级为作业指定当前可用计算资源。作业管理器根据作业调度器的指令调用作业提交模块,向作业执行层提交作业。作业监视模块监视每个作业的运行状态,并反馈给作业管理器。

作业管理器按统一标准格式接收各种不同类型的计算作业,再根据作业需求分别提交到作业执行层的不同子集群。这样的设计使得双层作业调度系统可以在作业执行层灵活增加不同的作业子集群,并对用户透明,用户作业脚本及提交作业命令保持不变。资源状态收集、作业提交、作业监视等模块通过调用各个子计算集群自有的 API 或命令行接口实现,但它们与作业调度器的 API 接口格式定义固定不变。这样的设计方案保证了作业管理层与作业执行层之间的松耦合性。

3　作业管理层设计及实现

3.1　各模块对外提供的接口

高能物理计算领域的通常作法是用户通过命令行接口实现作业提交查询等操作,因此本文开发了基于命令行的作业管理工具包。例如,用户 somebody 向作业管理器提交作业的命令如下:

```
job_submit_client/path/test.sh
```

该命令的第一个参数指定了作业脚本的位置,即/path/test.sh。在作业成功提交后,命令行接口向用户返回的信息如下:

```
Successfully submitted 1 job(s):44071
```

其中,44071 是服务端为该作业分配的 ID 编号。

作业管理层的各模块对外提供形似 RESTful API 的 TCP 接口,上述命令行接口是对 TCP 接口的封装。客户端通过 TCP 接口向服务器发出的查询请求和收到的回复都是 JSON 格式的字典类型变量,请求包含 method、path 等键值对,而回复包含 status、content 等键值对。上述 somebody 用户通过命令行向作业管理器提交作业时,客户端实际发出的内容如下:

```
{"method":"POST",
"path":"/jobs",
"user":"somebody",
"user_group":"somebody",
"body":[{"input_path":"/path/test.sh",
"exp_group":"somebody",
"req_cpu":1,
"req_preempt":0}]}
```

其中,body 中的 input_path 指定了作业脚本的位置,即/path/test.sh,而服务端回复给客户端的内容如下:

```
{'status':202,
'comment':'Accepted',
'content':[44071]}
```

计算机工程 www.ecice06.com

其中,44071 是服务端为该作业分配的 ID 编号。

3.2 基于 Eventlet 的基本框架

为达到快速高效的作业调度目标,本文将图 1 中的队列数据库和资源状态数据库均存放在内存中。计算作业和计算资源的信息均为高度结构化信息,如果使用 Redis 等 NoSQL 内存数据库在信息匹配时性能较差,则选用 SQLite 内存数据库存储作业信息和资源状态。SQLite 是一种嵌入式 SQL 数据库,可以像传统数据库一样将数据完全存储在磁盘上,还可以将整个数据库建立在内存中。基于内存的 SQLite 数据库能够提供优于传统磁盘数据库的访问性能,同时兼顾 SQL 数据库结构化数据表的特点,更易于开发及管理。

选用 SQLite 内存数据库要求作业管理层中的每个组件都需在同一进程中具备以下功能:

1)维护一个 SQLite 嵌入式内存数据库。

2)对外提供服务接口,响应系统管理员、用户、作业调度器及作业监视模块的请求,并对数据库进行相应的插入、删除、修改、查询等操作。例如,查询数据库中某个用户的排队信息,或者更新某个作业的运行状态等。

3)作为作业执行层或作业管理层内其他组件的客户端,定期采集作业执行层中资源及作业状态信息并实时更新相应数据库,以保证数据库内数据的实时性。例如查询 SLURM 集群的空闲资源信息,或者查询 HTCondor 集群的历史作业信息等。

本文采用协程(coroutine)模型实现上述功能,将请求响应部分和客户端操作部分分别编写为同一进程中的不同协程,使两者运行时互不阻塞,同时 SQLite 数据库以全局对象的形式供上述协程访问。

本文为作业管理层的各模块设计如图 2 所示的基本框架。Python 具有海量的第三方扩展包(Package),且易于编码,系统选择 Python 作为编程语言。Eventlet 是 Python 下实现协程模型的框架,被应用于 Openstack 等大型开源项目的开发,其性能与 Gevent、Twisted 等类似框架基本相同,因此选用 Eventlet 软件包实现各协程之间的并发。SQLAlchemy 是 Python 下访问 SQL 数据库的框架,用来访问包含 MySQL、SQLite 在内的多种 SQL 数据库,同时实现了对象关系映射(Object Relational Mapping,ORM),将对 SQL 数据行的操作转化为对 Python 对象的操作,避免手工书写 SQL 语句带来的兼容性问题及安全隐患,因此选用 SQLAlchemy 包作为 SQLite 数据库的访问接口。

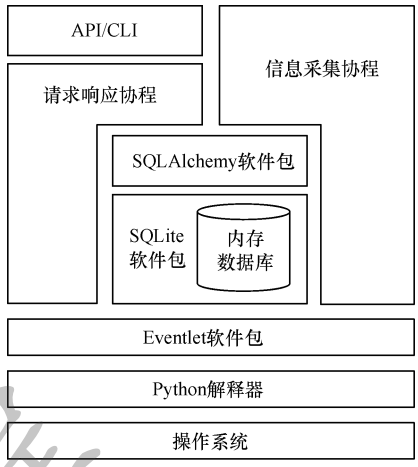

图 2 作业管理层的模块设计

Fig. 2 Design of modules in the job management layer

3.3 作业管理器

作业管理器为用户及作业调度器模块提供作业信息保存及作业队列管理功能。用户向作业管理器提交大量作业后,作业信息首先被存放于作业管理器中,只有作业调度器才能决定作业被提交到作业执行层中的指定子集群。一旦作业执行层中没有可用于某一作业运行的计算资源,则该作业会在作业管理器中排队等待。该设计可以绕过作业执行层各子集群自有的作业调度算法对作业的管理,将作业管理上移至作业管理层完成。作业管理层松耦合的架构还可以综合不同作业子集群的情况,提供诸如多队列、队列间公平性等更丰富的作业管理功能及作业调度策略。作业管理器的主要组成部分包括:

1)SQLite 内存数据库:通过 SQLAlchemy 提供访问,用来存储作业信息。作业数据库中存储的作业信息及相应数据表格式(部分)如表 1 所示,其中作业 ID 为该数据表的主键且为自增类型。

表 1 作业信息及相应的数据表格式

Table 1 Job information and its data table format

作业信息	数据库字段名	字段类型(长度)
作业 ID	job_hashname	int(11)
作业名称	job_name	varchar(45)
输入路径	input_path	varchar(4 096)
输出路径	output_path	varchar(4 096)
提交作业的用户名称	user	varchar(45)
作业所需 CPU 数	req_cpu	int(11)
作业状态	status	varchar(45)
作业所在子集群	run_host	varchar(45)
作业所在节点	run_node	varchar(45)

2)SocketServer 服务端:对用户提供形似 RESTful API 的 TCP 接口,响应用户的作业提交、

计算机工程 www.ecice06.com

查询、删除请求,作业调度器的作业批量查询、调度请求,以及作业监视模块的作业运行状况更新请求。

3)作业 ID 生成器:作业管理器包含一个全局单例对象,用来为新作业分配递增的 ID 信息,并将当前作业 ID 最大值同步至指定文件中。若作业管理器进程被意外中止并恢复,可从该文件中读出最近作业 ID 最大值,确保作业管理器内不存在重复作业 ID,以保证作业运行状态和记账进程的正常运行。

3.4　资源状态收集模块

资源状态收集模块负责收集作业执行层的可用计算资源信息,并暂存在内存数据库中。这提高了作业调度效率,尤其当作业执行层中存在异地计算子集群时,该模块会预先将异地计算资源的信息同步到本地,避免了异地站点间的高通信延迟。资源状态收集模块的主要组成部分包括:

1)SQLite 内存数据库:通过 SQLAlchemy 提供访问,用来存储计算资源的各项信息。资源状态数据库存储的计算资源信息及相应的数据表格式(部分)如表 2 所示,其中计算节点名称为该数据表的主键。

表 2　计算资源信息及相应的数据表格式
Table 2　Computing resource information and its data table format

计算资源信息	数据库字段名	字段类型(长度)
计算节点名称	node_name	varchar(45)
所在子集群	host_name	varchar(45)
计算资源拥有者	owner	varchar(45)
总 CPU 数	tot_cpu	int(11)
空闲 CPU 数	free_cpu	int(11)
作业提交者	allowed_user	varchar(4 096)

2)SocketServer 服务端:对用户提供形如 RESTful API 的 TCP 接口,可响应批量查询请求。

3)数据采集部分:根据配置文件,通过 API 定期查询各子集群的可用计算资源。

3.5　作业调度器

作业调度器是作业管理层的核心组件,负责根据作业调度策略,为用户作业设定执行的先后顺序及其应被提交到的子集群。作业调度器是一个独立模块,可以灵活添加多种调度算法。作业调度器在一个调度周期内完成的工作包括:

1)向作业管理器查询当前排队作业,向资源状态收集模块查询当前可用计算资源。

2)计算每个作业的优先级,并将作业按优先级排序。目前的作业调度器将作业的排队时间作为作业优先级,即 job_prio = cur_time - time_submit,时间越长,优先级越高,等价于先来先服务(First Come First Service,FCFS)算法。

3)按优先级顺序遍历作业。为每个作业寻找适合该作业的计算资源,并将该资源分配给该作业。如果某个计算集群或计算节点的全部资源已经被分配完,则将该集群或该节点从资源列表中删除,以加快其他资源的分配速度。作业调度器用一个列表记录成功分配到计算资源的作业以及每个作业相应的计算资源。

4)将上述计算资源分配列表发送至作业管理器,作业管理器调用作业提交模块,向指定计算集群提交指定作业。

作业调度器支持对机会计算作业的调度。机会计算作业由用户在提交作业时指定,可运行在任何一个子集群的任何可用资源上,以尽量减少排队时间;但如果这类作业运行期间,有更高优先级的作业需要计算资源,子集群的调度器有权中断机会计算作业的运行,将作业送回作业管理层中继续排队。为此,作业调度器在为作业分配计算资源时会执行如图 3 所示的算法,该算法具体步骤如下:

1)判断作业类型。若用户设置了机会计算作业标志,则认为作业为机会计算作业;若作业需要使用通用计算资源(GRES)或至少两个 CPU 核心,则认为作业为并行计算作业;否则认为作业为串行计算作业。

2)对于串行计算作业,尝试在 HTC 子集群的计算资源中为该作业分配资源。目前的作业调度器可将 HTCondor 及 TORQUE 计算集群作为 HTC 子集群。

3)若分配失败,则无论作业类型如何,尝试在 HPC 子集群中为该作业分配计算资源。目前的作业调度器可以将 SLURM 计算集群作为 HPC 子集群。

4)若分配失败且作业类型为并行计算作业,则先假设 HPC 子集群中的机会计算作业全部被清空,其占用的计算资源全部被释放,然后再次尝试为该作业分配计算资源。

5)若有作业通过步骤 4 分配到计算资源,则在步骤 4 假设的计算集群中,为在步骤 3、步骤 4 中已分配到资源的所有并行计算作业重新分配资源,并撤销步骤 3、步骤 4 对其他类型作业的资源分配。

计算机工程 www.ecice06.com

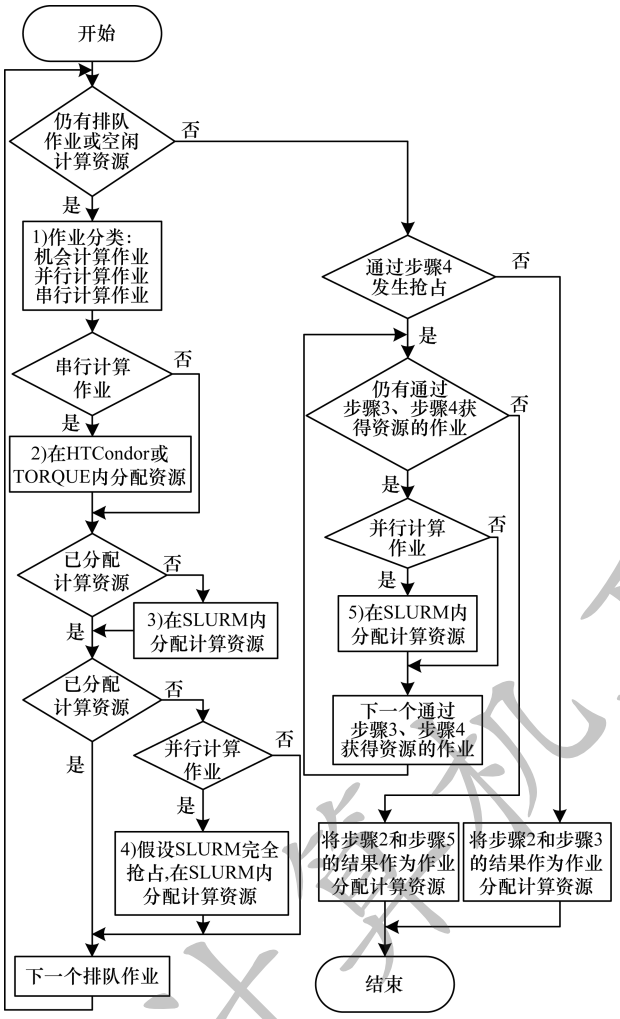

图 3 支持机会计算作业的资源分配算法

Fig. 3 Resource allocation algorithm supporting opportunity computing jobs

综上所述,当前的作业调度器具备以下 3 类调度策略:

1)作业优先级策略:作业调度器根据作业属性计算各个作业的优先级并进行排序。

2)每个作业的资源选择策略:为高优先级的作业选取合适的计算资源,并将作业提交到该资源所在的计算集群。

3)全局作业调度策略:当计算集群暂时没有与某一作业完全匹配的计算资源,且作业无法被立即执行时,作业调度器在当前作业、当前计算资源以及其他正在排队的作业之间实现协调。

3.6 作业提交模块

作业提交模块负责将作业提交到各子集群。作业调度器为作业选定计算集群后,由作业管理器以子进程的形式启动作业提交模块。作业提交模块进程以作业属主的身份向相应的子集群发送作业提交请求。系统中的作业提交模块可以将作业提交到下列类型的计算集群:

1)HTCondor 集群:通过 HTCondor 的 Python API 提交作业。作业提交模块将作业的脚本、输入/输出路径、资源需求等内容打包成一个 Python 字典,调用 HTCondor 的 classad 包生成相应的作业 ClassAd,调用 htcondor 包将 ClassAd 发送给 HTCondor SchedD 进程。

2)SLURM 集群:通过 SLURM 的 sbatch 命令行接口提交作业。作业提交模块将作业的脚本、输入/输出路径、资源需求等内容作为该命令的参数发送给 SLURM 集群。

3)TORQUE 集群:通过 TORQUE 的 qsub 命令行接口提交作业。作业提交模块将作业的脚本、输入/输出路径、资源需求等内容作为该命令的参数发送给 TORQUE 集群。

作业提交模块不仅是作业管理器与作业执行层解耦的必要条件,而且可避免作业管理器在作业提交期间的属主切换问题及不必要的等待,增强作业管理器的稳定性及作业调度器的运行效率。

3.7 作业监视模块

作业监视模块负责监视作业在作业执行层上的运行及完成情况。对于每个子计算集群,作业监视模块会为其创建协程,完成下列数据采集工作:

1)作业运行状态监视:通过 API 或命令行定期查询子集群上的作业运行情况,从中筛选出由作业管理层提交的作业,并通知作业管理器更新相应作业的状态。

2)作业完成情况监视:以定期增量查询的方式查询子集群的作业日志文件或作业数据库,包括但不限于 HTCondor 的 history 文件以及 SLURM 的 job_completions. txt 文件。从读取的数据中筛选出由作业管理层提交的作业,并通知作业管理器更新相应作业的状态,存档或删除相应的作业记录。

4 系统运行性能测试

4.1 作业管理器的作业接收性能测试

高能物理数据处理需要读写大量数据文件,通用方法是通过提交大量作业分别处理不同的数据文件达到快速处理数据的目的。因此,作业调度系统接收用户大批量作业的速率直接关系到用户的交互体验,这要求作业管理层具备快速接收用户大批量作业的能力。本文对系统大批量接收作业的性能进行测试,并与用户直接向 HTCondor 集群接收同样数量作业的性能进行比较。

计算机工程 www.ecaa06.com

测试环境包括运行作业接收服务端程序和作业提交客户端程序的两台服务器。这样既可模拟用户从专用登录节点提交作业的过程,又便于观察两类程序各自的 CPU 及内存使用情况。两台服务器均为双路 E5-2620 v3,16 GB 内存,Scientific Linux 6.5 操作系统。在该测试环境上部署 Python 2.6.6,并以此运行本文所述的作业管理器组件,作为该测试的实验组。同时在该测试环境上部署 HTCondor 8.4.11,使用 HTCondor 的 SchedD 组件和原生作业提交命令作为该测试的对照组。

在作业管理器组件和 HTCondor 上,分别测试了两种提交作业的方式:1)一次提交一个作业,逐个提交 500 个、1 000 个、1 500 个、2 000 个不同的作业,用时结果如图 4 所示。2)批量提交 500 个、1 000 个、1 500 个、2 000 个作业,用时结果如图 5 所示。

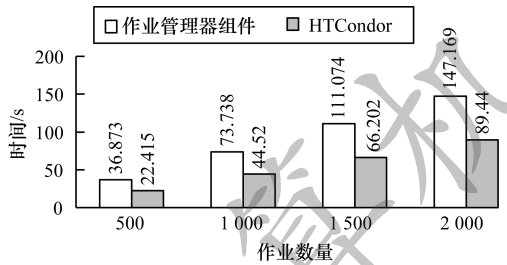

图 4　逐个提交大量作业用时对比结果

Fig. 4　Comparison of time consumed by the individual submission of massive jobs

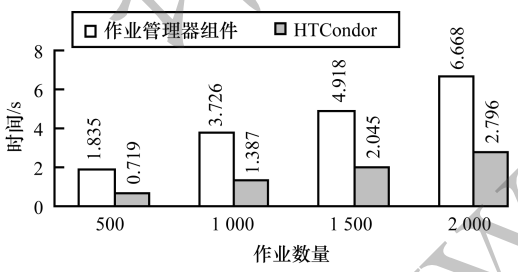

图 5　批量提交作业用时对比结果

Fig. 5　Comparison of time consumed by the batch submission of jobs

由图 4 可以看出,在逐个提交作业的性能测试中,作业管理器组件与 HTCondor 之间存在 1.64 倍 ~ 1.68 倍的性能差距。本文的作业管理器组件是 Python 程序,作业提交客户端也同样是 Python 脚本,而 HTCondor 的 SchedD 端与客户端都是 C 程序,两者的执行效率之间有一定的差距。本文在测试时直接通过 bash 执行 for 循环调用了实验组和对照组的客户端各 1 000 次,每次只提交一个作业,这就放大了上文所述的执行效率差距。每个作业提交成功后,在交互终端上对用户有一个成功提示信息,对于用户来说,提交 1 000 个作业是等待 44 s 还是 73 s 的差别并不明显。

由图 4 还可以发现,无论是本文中的作业管理器组件还是 HTCondor,逐个提交作业时的效率都不高,即使是性能较好的 HTCondor,其接收用户作业的速率也只有 22.4 个/s。因此,在第 2 个批量提交的性能测试中,通过两者的客户端分别批量提交 500 个、1 000 个、1 500 个、2 000 个作业。由图 5 可见,对于作业管理器,批量提交作业同通过循环提交等量的作业相比,约有 19.8 倍 ~ 22.6 倍的速度提升。在批量提交作业的情况下,作业管理器与 HTCondor 之间仍然存在 2.38 倍 ~ 2.69 倍的性能差距。然而,就绝对时间而言,两者批量提交 1 000 个作业的时间只相差 2.34 s,对于用户交互体验的影响并不明显。

4.2　作业管理器的作业转发性能测试

作业启动速率即作业从排队状态转入运行状态的速率,决定了一个 HTC 作业调度系统可以支持的计算集群的最大规模。在有足量作业排队的情况下,如果作业启动速率过慢而作业结束速率过快,则会导致计算资源被闲置。对于本文所述的双层作业调度系统,作业管理层的作业调度器承担了大部分作业调度算法的执行工作,最可能成为整个系统的性能瓶颈。同时,将大量作业逐个提交到作业执行层的各个子集群上也需要一定的时间。为探究作业管理层的作业转发性能及其对计算集群整体资源利用率的影响,本文对作业管理层接收并运行大量作业的完整流程进行测试,并与单独的 HTCondor 和 SLURM 调度系统进行比较。

测试环境为同一台服务器上的 3 台相同规格的虚拟机,其中一台为作业提交及调度节点,另外两台为计算节点。测试环境的宿主机使用 E5-2665 CPU,每台虚拟机拥有 1 个 CPU 核心和 4 GB 内存,宿主机和虚拟机均使用 Scientific Linux 6.9 操作系统。本文在该测试环境上分别部署 Python 2.6.6、HTCondor 8.5.6、SLURM 15.08.1 以及本文所述的作业管理器组件,并配置 HTCondor 使每个计算节点提供 24 个作业槽,配置 SLURM 使每个节点提供 24 个虚拟 CPU 核,因此每个实验组和对照组都可以同时运行 48 个单核作业。如表 3 所示,两个实验组将作业管理器组件分别与 HTCondor 及 SLURM 搭配使用,而两个对照组不使用作业管理器组件,仅分

计算机工程 www.ecice06.com

别使用 HTCondor 及 SLURM 进行作业调度。HTCondor 及 SLURM 上使用的调度算法为两者的默认算法。对于两个实验组,在作业管理层的作业调度器中,以作业排队时间作为优先级,实现了简单的先进先出(First In First Out,FIFO)调度算法。

表 3 测试环境内实验组与对照组的软件配置

Table 3 Software configuration of the experimental group and the control group in the test environment

比较项目	实验组		对照组	
	1	2	1	2
作业管理层	有	有	无	无
作业执行层	HTCondor	SLURM	HTCondor	SLURM
计算资源总量	48 个作业槽	48 个虚拟核	48 个作业槽	48 个虚拟核

本文分别向上述 4 组软件环境提交 500 个、1 000个、1 500 个、2 000 个计算作业,统计从第一个作业开始运行到最后一个作业完成所消耗的时间。提交的每个计算作业均为 sleep 1 000 s,占用 HTCondor 的 1 个作业槽或 SLURM 的 1 个虚拟 CPU 核,因此测试资源的实际数量不会对测试结果造成影响。该测试环境内的每个实验组和对照组都可以同时运行 48 个 sleep 作业,在不考虑作业调度、转发、I/O 等开销的情况下,理论上运行完 x 个作业需要的最短时间为$\lceil x \div 48 \rceil \times 1\,000$ s。当 $x = 1\,000$ 时,作业运行用时约 21 000 s,实际测试结果如图 6 所示。

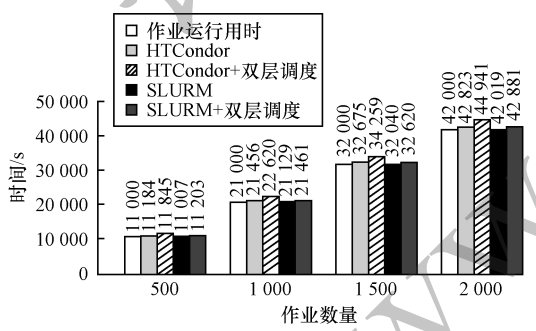

图 6 大批量提交作业用时对比结果

Fig. 6 Comparison of time consumed by the massive batch submission of jobs

由图 6 可以看出,双层作业调度模式与单一的 HTCondor 或 SLURM 等作业调度系统相比,作业执行用时分别增加了 4.8%~5.9% 和 1.5%~2.0% ,这是多队列、自定义调度算法等功能的必要开销。因此,笔者认为作业管理层的作业调度、作业转发过程都较高效,基本能够满足 HTC 计算集群满负荷运行的需求。

5 结束语

目前高能物理计算任务以单核串行作业为主,通过在计算集群上同时运行大量互不相关的计算作业来提高数据处理速度。随着计算集群规模的扩大,传统的 PBS、HTCondor 等集中式作业调度系统逐渐出现性能或功能方面的不足。本文提出双层作业调度系统的概念及基本框架,使系统管理员可以在作业管理层灵活配置多种调度算法,同时整个系统具有更好的伸缩性。在此基础上,本文实现了作业管理层的原型系统,并将其与 HTCondor 及 SLURM 进行整合。针对高能所计算平台的实际需求,本文对作业管理器的批量作业接收性能以及作业转发性能进行测试,结果表明作业管理器的性能对双层作业调度系统的效率基本没有影响,可满足高能物理计算集群高负荷运行的需求。

双层作业调度系统的核心是通过作业调度器实现的调度算法,但本文只实现了最基本的先进先出算法,后续工作将针对高能物理数据处理的特性提供一些更为复杂的调度算法,以保证多个用户组及作业队列之间的调度公平性。另外,为兼顾 HTC 和 HPC 作业在同一计算集群内的运行效果,计算资源在作业执行层的多个子集群之间的动态调配也是需要进一步研究的内容。

参考文献

[1] HENDERSON R L. Job scheduling under the portable batch system[C]//Proceedings of Workshop on Job Scheduling Strategies for Parallel Processing. Berlin, Germany: Springer, 1995: 279-294.

[2] YOO A B, JETTE M A, GRONDONA M. SLURM: simple Linux utility for resource management[C]// Proceedings of Workshop on Job Scheduling Strategies for Parallel Processing. Berlin, Germany: Springer, 2003: 44-60.

[3] LITZKOW M, LIVNY M, MUTKA M. Condor——a hunter of idle workstations[C]//Proceedings of the 8th International Conference of Distributed Computing Systems. Washington D. C., USA: IEEE Press, 1988: 104-111.

[4] XU Rongsheng, LANG Pengfei, CHEN Yaqing, et al. BES offline data processing[J]. Chinese Physics C, 1991, 15(7): 577-583. (in Chinese)
许榕生,郎鹏飞,陈雅青,等. 北京谱仪数据的离线处理[J]. 高能物理与核物理,1991,15(7):577-583.

[5] WANG Yifang. A neutrino experiment using the Daya bay reactor[J]. Physics, 2007, 36(3): 207-214. (in Chinese)
王贻芳. 大亚湾反应堆中微子实验[J]. 物理,2007,36(3):207-214.

计算机工程 www.ecice06.com

[6] NIE Simin,ZHANG Jilong,TAN Youheng,et al. Real time transmission and analysis of the Yangbajing cosmic rays observation data[J]. Nuclear Electronics and Detection Technology,2007,27(1):14-17. (in Chinese)
聂思敏,张吉龙,谭有恒,等. 羊八井宇宙线观测数据实时传输及处理系统[J]. 核电子学与探测技术,2007,27(1):14-17.

[7] AN Fengpeng, AN Guangpeng, AN Qi, et al. Neutrino physics with JUNO[J]. Journal of Physics G: Nuclear and Particle Physics,2016,43:222-401.

[8] CAO Zhen. A future project at tibet: the Large High Altitude Air Shower Observatory (LHAASO)[J]. Chinese Physics C,2010,34(2):249-252.

[9] TORQUE resource manager[EB/OL]. [2018-12-27]. http://www. adaptivecomputing. com/products/open-source/torque/.

[10] MAUI——adaptive computing[EB/OL]. [2018-12-27]. http://www. adaptivecomputing. com/products/open-source/maui/.

[11] LIVNY M, BASNEY J, RAMAN R, et al. Mechanisms for high throughput computing[J]. SPEEDUP Journal, 1997,11(1):36-40.

[12] LAHIFF A, DEWHURST A, KELLY J, et al. HTCondor at the RAL tier-1[EB/OL]. [2018-12-27]. https://indico. cern. ch/event/272785/contributions/1612799/.

[13] SUN Zhenyu, SHI Jingyan, JIANG Xiaowei, et al. Evaluation of resource management methods for large high energy physics computer cluster[J]. Computer Science,2017,44(10):85-90. (in Chinese)
孙震宇,石京燕,姜晓巍,等. 大型高能物理计算集群资源管理方法的评测[J]. 计算机科学,2017,44(10):85-90.

[14] TIAN Wei, LIU Yunliang, SUN Yu, et al. Energy models of urban building stock based on geographic information system[J]. Building Energy Efficiency, 2017,45(7):40-46. (in Chinese)
田玮,刘云亮,孙禹,等. 基于地理信息系统的城市建筑能耗模型建立方法[J]. 建筑节能,2017,45(7):40-46.

[15] WANG Weihong, XU Wentao, XIA Liegang, et al. Study for parallel segmentation optimization algorithm of mean shift in large-scale remote sensing image[J]. Journal of Chinese Computer Systems, 2015, 36(5): 1085-1090. (in Chinese)
王卫红,徐文涛,夏列钢,等. 大规模遥感影像 Mean Shift 并行分割优化算法研究[J]. 小型微型计算机系统,2015,36(5):1085-1090.

[16] JIANG Shuzheng. The design and implementation of remote sensing data processing grid platform[D]. Qingdao: Shandong University of Science and Technology,2011. (in Chinese)
姜书政. 遥感数据处理网格平台的设计与初步实现[D]. 青岛:山东科技大学,2011.

[17] ZHANG Li. Study on architecture of cooperative processing for remote sensing images and its application on photogrammetry[D]. Zhengzhou: PLA Information Engineering University,2015. (in Chinese)
张丽. 遥感影像协同处理体系及摄影测量应用研究[D]. 郑州:解放军信息工程大学,2015.

[18] LIAO Xiangke, XIAO Liquan, YANG Canqun, et al. MilkyWay-2 supercomputer: system and application[J]. Frontiers of Computer Science,2014,8(3):345-356.

[19] LU Zhonghua, HU Tengteng, WANG Yangang, et al. The design and implement of HPC based on Slurm for deep learning[J]. e-Science Technology and Appliation, 2018,9(2):40-45. (in Chinese)
陆忠华,胡腾腾,王彦棡,等. 基于 Slurm 的深度学习高性能计算平台设计及其调度实现技术[J]. 科研信息化技术与应用,2018,9(2):40-45.

[20] LIU Yangbing. Research on job management middleware based on cluster[D]. Changsha: Hunan University, 2014. (in Chinese)
刘杨兵. 基于集群环境的作业管理中间件的研究与实现[D]. 长沙:湖南大学,2014.

[21] HOLLOWELL C, BARNETT J, CARAMARCU C, et al. Mixing HTC and HPC workloads with HTCondor and slurm[J]. Journal of Physics: Conference Series, 2017,898(8):1-8.

[22] SFILIGOI I. glideinWMS——a generic pilot-based workload management system[J]. Journal of Physics: Conference Series,2008,119(6):1-6.

[23] BALCAS J, BELFORTE S, BOCKELMAN B, et al. Pushing HTCondor and glideinWMS to 200K + jobs in a global pool for CMS before run 2[J]. Journal of Physics: Conference Series,2015,664(6):1-6.

[24] BALCAS J, BOCKELMAN B, HUFNAGEL D, et al. Stability and scalability of the CMS global pool: pushing HTCondor and glideinWMS to new limits[J]. Journal of Physics: Conference Series,2017,898(5):1-5.

[25] KIM J S, RHO S W, KIM S Y, et al. HTCaaS: leveraging distributed supercomputing infrastructures for large-scale scientific computing[EB/OL]. [2018-12-27]. http://citeseerx. ist. psu. edu/viewdoc/summary? doi = 10. 1. 1. 646. 9632.

[26] AIFTIMIEI D C, ANTONACCI M, BAGNASCO S, et al. Geographically distributed batch system as a service: the INDIGO-DataCloud approach exploiting HT Condor[J]. Journal of Physics: Conference Series,2017, 898(5):1-8.

[27] Apache Mesos[EB/OL]. [2018-12-27]. http://mesos. apache. org/.

编辑　陆燕菲

计算机工程 www.ecice.com