多资源公平调度器在 Hadoop 中的实现

马肖燕 洪 爵

（ 中国科学院深圳先进技术研究院 深圳 518055 ）

摘 要 目前Hadoop的作业调度算法都是将系统中的多类资源抽象成单一资源，分配给作业的资源均是节点资源中固定大小的一部分，称为插槽。这类基于插槽的算法没有考虑到系统多资源的差异性，忽略了不同类型作业对资源的不同需求，因此导致系统在吞吐量和平均作业完成时间上性能低下。本文研究了多资源环境下公平调度算法在Hadoop中的实现，设计了一种多资源公平调度器MFS(Multi-resource Fair Scheduler)。MFS采用了DRF(Dominant Resource Fairness)调度思想，使用需求向量来描述作业对各类资源的需求，并按照需求向量中各资源的大小给作业分配资源。MFS能更加充分有效地使用系统的各类资源，并能满足不同类型作业对资源的不同需求。实验表明相比于基于插槽的Fair Scheduler与Capacity Scheduler，MFS提高了系统的吞吐量，降低了平均作业完成时间。

关键词 Hadoop；作业调度算法；插槽；多资源；公平

A Muti-resource Fair Scheudler of Hadoop

MA Xiao-yan HONG Jue

(*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen,* 518055, *China*)

Abstract Hadoop job schedulers typically use a single resource abstraction and resources are allocated at the level of fixed-size partition of the nodes, called slots. These job schedulers ignore the different demands of jobs and fair allocation of multiple types of resources, leading to poor performance in throughput and average job completion time. This paper studies and implements a Muti-resource Fair Scheduler (MFS) in Hadoop. MFS adopts the idea of Dominant Resource Fairness (DRF). It uses a demand vector to describe demands for resources of a job and allocates resources to the job according to the demand vector. MFS uses resources more efficiently and satisfies multiple jobs with heterogeneous demands for resources. Experiment results show that MFS has higher throughput and shorter average job completion time compared to Hadoop slot-based Fair Scheduler and Capacity Scheduler.

Keywords Hadoop; job scheduling algorithm; slot; muti-resource; fair

1 引 言

Hadoop是一种开源的分布式系统基础架构，常用于大规模数据处理。现在Hadoop有着广泛的应用领域，尤其是在Web领域。国内外很多公司都使用Hadoop作为系统平台。淘宝利用Hadoop系统存储并处理电子商务交易的相关数据，搜索巨头百度使用Hadoop进行数据挖掘和日志分析，Adobe从社会服务到结构化数据存储使用的平台都是Hadoop，FaceBook借助Hadoop支持其机器学习和数据分析。

作业调度是Hadoop的核心技术之一。它的主要功能是按照特定的算法来选择调度作业并对计算资源进行控制，因此作业调度算法直接关系到Hadoop整个系统的性能和资源的利用情况。但是目前Hadoop的作业调度算法都是将系统中的多类资源抽象成单一资源，分配给作业的资源均是节点资源中固定大小的一部分，称为插槽。这类基于插槽的作业调度算法主要存在三个问题。首先，这类算法并没有考虑到节点的实际运行能力，只是按照运行的任务个数来判断节点

作者简介：马肖燕，研究助理，研究方向为高性能计算，E-mail：xy.ma@siat.ac.cn；洪爵，助理研究员，研究方向为云计算、高性能计算。

能否再接受任务，可能会造成集群的某些节点负载过重。其次，它将多类资源抽象成单一资源进行公平分配，并未考虑系统中多类资源的公平分配。再者，它忽略了不同类型作业对资源的不同需求。作业占用的资源均是节点中固定大小的一部分资源，不管作业对资源需求的不同。这造成了作业已分配资源与需求资源之间的不匹配，这种不匹配并不能通过改变节点插槽的个数来解决。

本文设计并实现了一种多资源公平调度器MFS。DRF是针对多资源的公平调度算法，能够有效解决目前Hadoop作业调度算法存在的问题，因此MFS采用了DRF[1]调度思想。MFS针对Hadoop的运行环境，对DRF算法进行了改变和补充，实现了用户和作业两个层次的公平调度。由于MFS使用需求向量来描述作业对各类资源的需求，并按照需求向量中各资源的大小给作业分配资源，因此MFS实现时需要修改和增加Hadoop的一些属性。MFS的实现主要包括四个内容：增加作业属性、更改TaskTracker申请任务的条件、设置java虚拟机参数和继承抽象类TaskScheduler。MFS考虑到了系统的多资源分配和多类作业的不同需求，能更加充分有效地使用各类资源。实验表明MFS提高了系统的吞吐量，降低了平均作业完成时间。

论文第二部分介绍Hadoop作业调度算法的相关工作，第三部分介绍MFS的设计与实现，第四部分介绍实验过程和结果，第五部分给出结论并指出接下来的工作。

2　相关工作

Apache的Hadoop版本自带了三种作业调度算法：FIFO，Fair Scheduler，Capacity Scheduler。FIFO调度即先来先调度，是Hadoop默认的调度算法。FIFO调度算法比较简单，适合单用户提交的大型批处理作业，在多作业多用户的情况下效率低下，因为FIFO每次只允许一个作业运行，运行的作业占用整个集群的所有资源，即使有空闲资源也不会分配给其他作业。

Fair Scheduler[2]是FaceBook提出的作业调度算法，目的是能够满足多用户多类型作业的并行运行。该算法的设计思想是尽可能保证所有的作业都能够获得等量的资源份额。算法以池的形式管理作业，默认情况下每个用户都有一个独立的作业池，用户能获得相等份额的资源而不管用户提交了多少作业。所有的作业按照赤字进行排序，赤字最大的作业优先被调度。赤字是作业应得的计算资源量与实际获得的计算资源量之间的差值，差值越大说明该作业受到的不公平待遇越多。Fair Scheduler优先调度赤字大的作业，目的是尽可能保证作业之间的公平。

Capacity Scheduler[3]是由雅虎提出的作业调度算法。该算法支持多队列，每个队列都可以通过配置获得一定数量的资源。当系统中有了空余资源，该算法首先选择一个具有最多空闲资源的队列。队列的空闲度是队列中空闲资源与该队列分得的计算资源之间的比值，比值越大说明队列空闲资源越多。选择出一个队列后，在选出的队列中按照FIFO调度一个作业。

Delay scheduler[4]通过优化数据的本地化进而提高Hadoop的吞吐量。在Hadoop公平调度算法中，都存在着调度公平性与数据本地化的冲突。延迟调度的设计目标是尽可能小的减少公平性，并提高作业的数据本地化，最终提高系统的吞吐量。延迟调度的思想是：如果需要调度的作业没有数据本地化的任务，就让该作业延迟一定时间，选择下一个可以调度本地化任务的作业。该算法是针对小作业提出的，延迟调度能大量提高小作业的吞吐量。

资源感知调度[5]针对Hadoop基于插槽的算法没有考虑节点上各类型资源的可利用情况与实际负载水平这个问题，提出了两种具体的解决算法：动态空闲插槽算法与空闲插槽优先级算法。动态插槽算法允许根据节点实际的资源状态动态更改插槽的值。空闲插槽优先级算法将节点按照实际资源可利用率进行排序，资源可利用率高的节点先被分配任务。

另外还有其他的作业调度算法，是针对特殊应用提出的。LATE[6]算法是针对集群的节点是异构的环境提出的；Dynamic Proportional Scheduler[7]允许用户动态调整作业的优先级，但是并不保证作业在指定的时间内完成；Polo[8]提出的算法是针对软实时的作业提出的；Deadline Constraint Scheduler[9]专门应用于硬实时的作业。

3　MFS作业调度器

本节介绍MFS作业调度器的设计与实现。由于MFS采用了DRF算法的思想，因此3.1节先简单介绍DRF算法。3.2节介绍MFS作业调度器的设计。3.3节介绍MFS作业调度器在Hadoop中的实现。

3.1 DRF算法

DRF是针对多资源分配的公平算法。目前该算法

在Mesos[10]平台中进行了实现。DRF算法的公平满足四个特性。第一是共享性(Sharing Incentive)，即DRF确保各用户平均占用资源；第二是真实性(Strategy-Proofness)，即每个用户都不能通过欺骗来获得更多资源；第三是帕累托效率性(Pareto-Efficient)，即每个用户都不能在未减少其他用户占有的资源的情况下增加自己的资源份额；第四是非抢占性(Envy-Free)，即每个用户都不能抢占其他用户的资源。DRF充分考虑到了不同用户对多类资源的不同需求，使用需求向量来描述作业对各类资源需求的大小。用户根据运行作业的资源需求，设置一个需求向量，如<1 CPU，1 GB>表明该用户每个任务运行时被分配资源的大小是1个CPU，1 GB的内存。需求向量的大小和种类根据用户的需求及系统资源的情况来设置。

在用户多类型的资源需求中，必有一类资源是最主要的，例如CPU密集型的作业最主要的资源是CPU，内存密集型的作业最主要资源是内存。不同的用户最主要资源也是不相同的。DRF算法为每个用户计算各类资源的占用份额，其中最大的资源份额称为dominant share，最大份额对应的资源即是用户最主要的资源。Dominant share最小的用户优先被调度和分配资源。

3.2 MFS设计

DRF是针对多资源的公平调度算法，能够有效解决目前Hadoop作业调度算法存在的问题，因此MFS采用了DRF调度思想。但是DRF调度算法并不能直接应用在Hadoop平台中，具体原因包括下面两个方面。首先，在DRF算法中，用户每次只提交一个作业，用户之间的公平即是作业之间的公平，算法只需实现用户层次的调度。在Hadoop中，各用户提交的作业数目是不同的，MFS要满足用户和作业两个层次的公平，实现用户与作业两个层次的调度。其次，在Hadoop中，作业被分成多个任务来运行。DRF算法中用户作业的任务类型都是一样的，在选择任务时，可以随机从用户作业的任务中选择其中一个。Hadoop的任务类型有两种：map和reduce，map与reduce任务存在着先后的依赖关系。MFS在选择任务时，需要选择任务类型，还要考虑任务之间的依赖关系。

在Hadoop中，不同用户提交的作业数目是不同的。MFS为每个用户设置一个组(group)，每个用户的作业都提交到各自的组中。MFS使用需求向量来描述作业对各类资源的需求，并按照需求向量中资源的大小给作业分配资源。MFS为每个用户和作业计算各类资源的占用份额，其中最大的资源份额即是dominant share，dominant share小的用户和作业优先被调度。用户可以给用户组和作业设置权重，MFS通过权重来调节dominant share的值。在MFS中，作业和用户的资源份额计算过程分别如图1和图2所示。

假设集群中资源种类有 m 种，集群中各类资源总的大小分别是 r1,r2,…rm；
计算作业权重 jobweight；
获取作业正在运行的任务个数 N，包括 map 与 reduce 任务；
获取作业需求向量中各类资源的大小，即为

图1 作业dominant share计算过程

计算用户作业组的权重 groupweight；
计算用户各类资源的占用份额 u1,u2,…,um，ui=用户组中各作业的 i 类资源份额之和；
用户 dominant share=max{u1,u2,…,um}/groupweight

图2 用户dominant share计算过程

MFS的调度分为两个层次。一是在用户层次，dominant share值最小的用户先被调度。二是在作业层次，选择出用户后，该用户组内dominant share最小的作业优先被调度。在Hadoop中，作业被分成多个任务来运行，MFS要选择出作业的一个任务。由于map与reduce任务存在着先后的依赖关系，MFS优先选择map任务，即若作业中有map任务尚未调度，则选择map任务，否则选择reduce任务。MFS调度流程如图3所示。

```
JobTracker 接收到来自 TaskTracker 的任务请求；
将各用户按照 dominant share 进行递增排序;
for u in users do
    将该用户组中的作业按照 dominant share 进行递增排序;
    for j in jobs do
        if (节点的剩余资源>作业 j 的需求) then
            优先选择作业 j 的 map 任务；
        else
            return null;
        end if
    end for
```

图3 MFS调度流程

3.3 MFS实现

我们在Hadoop中实现MFS作业调度器。MFS的实现主要包括四个内容：增加作业属性、更改

TaskTracker申请任务的条件、设置java虚拟机参数和继承抽象类TaskScheduler。

MFS使用需求向量来描述作业对各类资源的需求，Hadoop中并没有这些作业属性，因此需要增加有关作业需求向量的各种属性，以及用于描述用户组的属性。JobConf是Hadoop作业属性的设置文件，是在Hadoop框架中用户描述MapReduce作业的主要接口。Hadoop会严格按照用户设置的JobConf的作业属性去运行作业。因此我们将新的属性增加到Hadoop的JobConf文件中。

在Hadoop目前的作业调度算法中，TaskTracker的节点若有空闲插槽，则TaskTracker就可以向JobTracker申请任务。MFS不是按照运行的任务个数来判断节点能否再接受任务，而是考虑到了各节点可利用资源的大小。在MFS中，当节点中资源利用率低于某个值时，TaskTracker向JobTracker申请任务。

在Hadoop中，任务是在java虚拟机中运行的，因此要设置java虚拟机，使java虚拟机按照需求向量的大小占用资源。为了满足任务的需求，java虚拟机内存可以设置为作业需求向量中内存的值，这一过程通过修改虚拟机创建时的参数配置完成。其他类型资源，如CPU、I/O、网络资源，我们使用cgroup[11,12]来控制节点中各java虚拟机占用的资源比例。在节点中，我们为每个任务创建一个控制组，控制组中包含CPU、I/O、网络等子系统，我们为各子系统设置权重。当节点中运行一个任务时，我们获取该任务java虚拟机的进程pid，并将该进程的pid加入到新的控制组中。

Hadoop提供了一个抽象类TaskScheduler，具体实现的作业调度器都是该类的继承。MFS作业调度器需要继承抽象类TaskScheduler，实现该类的四个方法：start、terminate、getJobs和assignTasks。另外，MFS作业调度器还要继承抽象类JobInProgressListener来完成作业的加入队列与从队列中删除等操作，继承类HttpServLet用于显示JobTracker网页上的信息。

4　实验分析与结果

实验使用的Hadoop版本为Hadoop-0.20.2，集群的节点数为21个，包括一个主节点，20个从节点，每个节点的CPU个数为16，内存大小为16 GB。Hadoop的作业调度算法大都是针对特殊应用需求提出的，如针对实时性作业、节点异构情况等，Fair Scheduler与Capacity Scheduler是针对多用户多类型作业提出的普遍适用的调度算法，所以我们将MFS与Hadoop版本中的Fair Scheduler与Capacity Scheduler进行比较，衡量算法的指标是平均作业完成时间和吞吐量。

实验通过Hadoop示例中的四类作业对各算法进行衡量，这四类作业分别是Sort、WordCount、Terasort和Grep，其中Sort、Terasort常作为Hadoop的基准测试程序，WordCount、Grep是使用非常广泛的程序。为了更加精确地分析不同作业对算法的影响，我们将作业按照处理数据的大小级别分为小作业与大作业。大、小作业处理数据的大小在设置时区分出级别，我们选取小作业处理数据的大小范围是60~500 MB，大作业处理数据的大小范围是4~8 GB。实验分为三组，这三组实验的设置如表1所示。第一组实验主要对小作业进行测试，第二组实验是对大作业进行测试，第三组是测试混合类作业，包括大作业和小作业。

Fair Scheduler与Capacity Scheduler是基于插槽的，在实验过程中，我们根据机器的配置情况与作业对资源的需求，设置插槽数为13到17。在MFS中，需求向量用于描述作业对各类资源的需求，由于CPU和内存是最主要的两项资源，因此在设置作业需求向量时，我们目前暂时仅考虑CPU和内存资源。我们根据作业的性质及对资源的需求情况，设置需求向量中CPU的个数为1或2，内存大小范围设置为500 MB到2 GB之间。CPU的个数与内存大小均在特定范围内随机产生数值。

表1 作业参数设置

作业类型	作业个数	Maps/Job	Reduces/Job	DataSize/Job
小作业	80	1~10	1	60MB~500MB
大作业	40	70~160	7~16	4GB~8GB
混合类型作业	小作业：40	小作业：1~10	小作业：1	小作业：60MB~500MB
	大作业：20	大作业：70~160	大作业：7~16	大作业：4GB~8GB

4.1　小作业测试结果与分析

在小作业的运行中，我们同时提交了80个作业，

包括四类作业，每类作业提交20个。我们通过平均作业完成时间与吞吐量对三类调度算法进行比较，吞吐量是在十分钟内作业的完成个数。吞吐量的计算过程是先计算单位时间内完成的作业个数，然后转换为十分钟内作业的完成个数。平均作业完成时间的结果如图4所示，吞吐量计算结果如图5所示，图中不同插槽数测试的结果用不同颜色的柱形来表示。通过图4与图5可以看出，Fair Scheduler与Capacity Scheduler在不同插槽数的情况下，运行的效果差别不是太大。Capacity Scheduler算法应用于小作业时，性能较差。MFS的平均作业完成时间最短，吞吐量最高，性能最好。

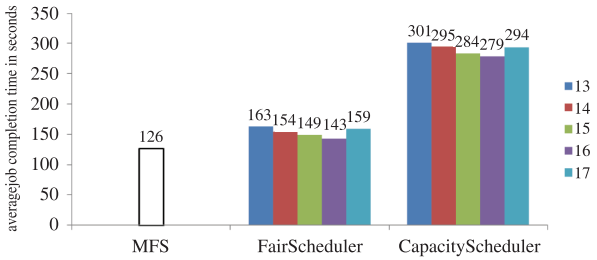

图4 小作业平均完成时间

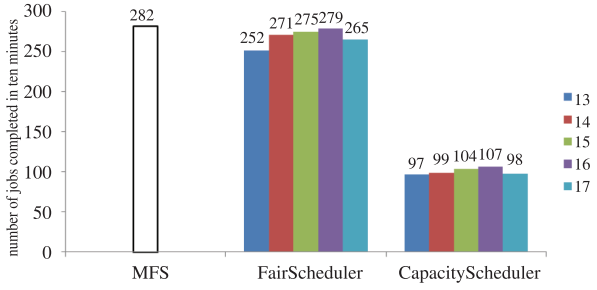

图5 小作业在十分钟内完成个数

4.2 大作业测试结果与分析

在大作业的运行中，我们同时提交了40个作业，包括四类作业，每类作业提交10个。平均作业完成时间的结果如图6所示，吞吐量计算结果如图7所示。通过图6与图7可以看出，Capacity Scheduler算法应用于大作业时性能比较好，Fair Scheduler的性能反而较差，MFS的平均作业完成时间最短，吞吐量最高。MFS的性能在大作业运行的情况下也是最好的。

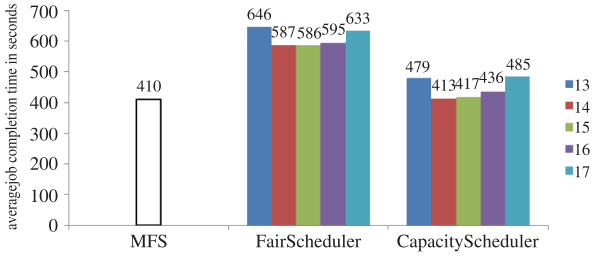

图6 大作业平均完成时间

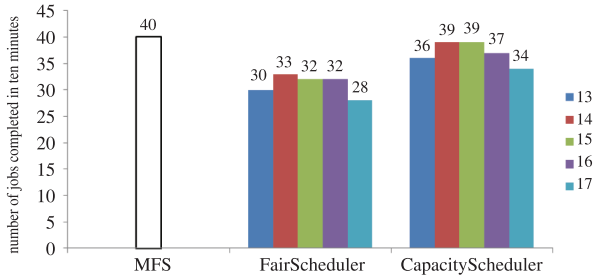

图7 大作业在十分钟内完成个数

4.3 混合类作业测试结果与分析

在混合类作业的运行中，我们同时提交了60个作业，包括用户A提交的40个小作业与用户B提交的20个大作业，大小作业都包含四类作业，每类作业的提交个数均相同。平均作业完成时间的结果如图8所示，吞吐量计算结果如图9所示。通过图8与图9可以看出，在混合类作业的运行中，MFS比Capacity Scheduler与Fair Scheduler的性能都要好。

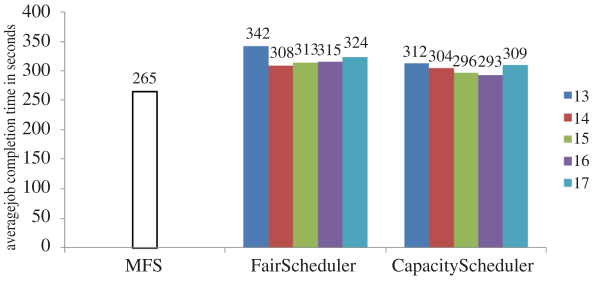

图8 混合类作业平均完成时间

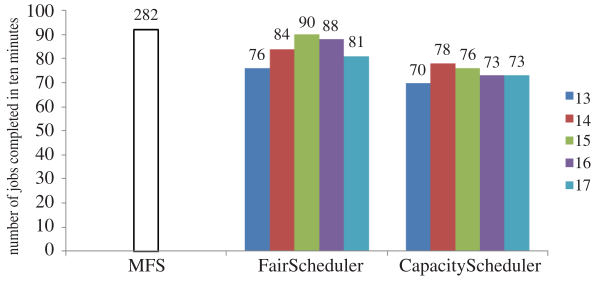

图9 混合类作业在十分钟内完成个数

通过这三组实验，我们可以看出MFS适合各类型的作业，在各类型作业运行中，MFS均比Capacity Scheduler与Fair Scheduler的性能好。这是因为MFS使用需求向量来描述作业对各类资源需求的大小，能够更加精确有效地使用系统的各类资源。其次，MFS充分考虑到了各类型作业对不同类型资源的不同需求，用户可以根据实际需求设置作业的需求向量中各类资源的大小。另外，MFS算法还考虑到了各节点可利用资源的差异，不是简单地根据节点中运行的任务个数来判断该节点是否还能够运行任务。

5 总　结

本文针对目前Hadoop作业调度算法存在的不足，设计实现了一种多资源公平调度器MFS。MFS采用DRF多资源公平调度思想，使用需求向量来描述作业对各类资源需求的大小，能更加充分有效地使用系统的各类资源，并能满足不同类型作业对资源的不同需求。实验表明，MFS适合各类型的作业，相比于Fair Scheduler与Capacity Scheduler，MFS提高了系统的吞吐量，降低了平均作业完成时间。MFS中需求向量的设置直接关系着系统性能的好坏，目前并没有具体的方法对其进行精确设置，我们采用的只是在某个范围内随机产生数值。下一步的研究内容就是需求向量的设置。

参 考 文 献

[1] Ghodsi A, Zaharia M, Hindman B, et al. Dominant resource fairness: fair allocation of multiple resource types [C] // USENIX Symposium on Networked Systems Design and Implementation Conference. Boston, USA, 2011.

[2] Fair Scheduler [EB/OL]. http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html.

[3] Capacity Scheduler [EB/OL]. http://hadoop.apache.org/common/docs/r0.20.2/capacity_scheduler.html.

[4] Zaharia M, Borthakur D, Sen Sarma J, et al. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling [C] // EuroSys Conference. Paris, France, 2010.

[5] Yong M, Garegrat N, Mohan S. Towards a resource aware scheduler in hadoop [C] // The IEEE International Conference on Web Services. Los Angeles, USA, 2009.

[6] Zaharia M, Konwinski A. Improving mapreduce performance in heterogeneous environments [C] // USENIX Symposium on Operating Systems Design and Iplementation. San Diego, USA, 2008.

[7] Thomas S, Kevin L. Dynamic proportional share scheduling in hadoop [C] // Job Scheduling Strategies for Parallel Processing. Atlanta,USA, 2010.

[8] Polo J, Carrera D, Becerra Y, et al. Performance-driven task co-scheduling for mapreduce environments [C] // IEEE/IFIP Network Operations and Management Symposium. Osaka, Japan, 2010.

[9] Kc K, Anyanwu K. Scheduling hadoop jobs to meet deadlines [C] // International Conference on Cloud Computing, Indianapolis ,USA,2010.

[10] Hindman B, Konwinski A, Zaharia M, et al. Mesos: a platform for fine-grained resource sharing in the data center s [C] // USENIX Symposium on Networked Systems Design and Iplementation. Boston, USA, 2011.

[11] Cgroup [EB/OL]. http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt

[12] Qin A, Tu D D, Shu C C, et al. XConveryer: guarantee hadoop throughput via lightweight OS-level virtualization [C] // International Conference on Grid and Cooperative Computing. Lanzhou, China, 2009.

◆ 动态与信息 ◆

《集成技术》征订启事

《集成技术》系由中国科学院深圳先进技术研究院及科学出版社主办的综合性学术刊物，2012 年 5 月创刊，主要报道计算机系统、机器人、先进制造、自动化、电动汽车、医疗设备、生物医药、互联网、通信、云计算、高性能计算、智能科学与技术、人机交互系统、装备技术、图形及图像处理系统、CAD/CAE/CAM 等领域科技的前沿和进展。期刊遵循理论与实践统一和百花齐放、百家争鸣的方针，集科学性、学术性、实用性与知识性为一体，以科研人员、高等院校师生以及工程技术人员为主要读者对象，努力向着成为高水平研究成果和思想的交流平台的目标前进。

《集成技术》热忱欢迎国内外专家学者投稿，也欢迎感兴趣的读者订阅。每期 48 元，全年 6 期含邮费 288 元。

地址：深圳市南山区西丽大学城学苑大道 1068 号（邮编：518055）

电话：0755-86392070

E-mail：jcjs@siat.ac.cn

网站：http://jcjs.siat.ac.cn/ch/index.aspx