(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104794000 A
(43)申请公布日 2015.07.22

(21)申请号 201410030595.5

(22)申请日 2014.01.22

(71)申请人 深圳市沃信科技有限公司
地址 518000 广东省深圳市南山区海文花园海扉阁 B 座 202

(72)发明人 金清德

(74)专利代理机构 深圳盛德大业知识产权代理事务所（普通合伙）44333
代理人 贾振勇

(51)Int.Cl.
G06F 9/50(2006.01)

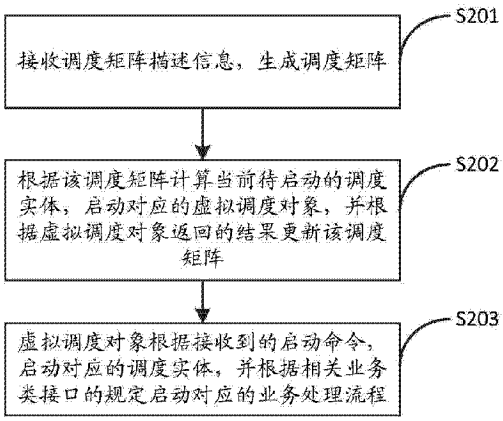
权利要求书2页 说明书11页 附图4页

(54)发明名称

一种作业调度方法及系统

(57)摘要

本发明适用于计算机领域，提供了一种作业调度方法及系统，所述方法包括下述步骤：接收调度矩阵描述信息，生成调度矩阵；根据所述调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象，并根据虚拟调度对象返回的结果更新所述调度矩阵；虚拟调度对象根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程。本发明通过矩阵方式进行作业调度，在调度过程中不仅关注资源的使用，而且考虑到了资源使用的结构系，使得很多复杂结构的调度可以很快完成。



CN 104794000 A

1. 一种作业调度方法，其特征在于，所述方法包括下述步骤：

接收调度矩阵描述信息，生成调度矩阵；

根据所述调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象，并根据虚拟调度对象返回的结果更新所述调度矩阵；

虚拟调度对象根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程；

所述调度矩阵描述信息至少包含调度矩阵的结构信息，以及虚拟调度对象的类别；

所述调度矩阵至少包括调度矩阵的标识，以及调度矩阵的方向标识；

所述调度矩阵包括一个或者多个下级调度矩阵，和/或一个或者多个虚拟调度对象的类别标识；

所述虚拟调度对象实现与对应的调度实体的接口，所述接口至少包括控制类接口；

所述调度矩阵的方向标识用于标识当前调度矩阵并行调度或者串行调度。

2. 如权利要求 1 所述的作业调度方法，其特征在于，所述调度矩阵描述信息为静态配置文件，或者动态指令。

3. 如权利要求 1 所述的作业调度方法，其特征在于，所述调度矩阵的方向分别用横向和纵向表示；

如果一个调度矩阵的方向是横向，表明所述调度矩阵的下级调度矩阵或者虚拟调度对象均串行执行；

如果一个调度矩阵的方向是纵向，表明所述调度矩阵的下级调度矩阵或者虚拟调度对象均并行执行。

4. 如权利要求1、2或3所述的作业调度方法，其特征在于，所述根据所述调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象的步骤具体为：

定位到一级调度矩阵；

获取所定位的调度矩阵的方向标识；

按照调度矩阵的方向访问所定位的调度矩阵的各下级调度矩阵；

判断下级矩阵是否为最简矩阵，是则按照调度矩阵的方向启动各个虚拟调度对象，否则定位到本级调度矩阵，继续根据本级调度矩阵的方向标识访问本级调度矩阵的各下级调度矩阵；

所述最简矩阵为下级调度矩阵中只有虚拟调度对象的调度矩阵。

5. 一种作业调度系统，其特征在于，所述系统包括：

调度矩阵生成单元，用于接收调度矩阵描述信息，生成调度矩阵；

虚拟调度对象，用于根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程；

调度执行单元，用于根据所述调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象，并根据虚拟调度对象返回的结果更新所述调度矩阵；

所述调度矩阵描述信息至少包含调度矩阵的结构信息，以及虚拟调度对象的类别；

所述调度矩阵至少包括调度矩阵的标识，以及调度矩阵的方向标识；

所述调度矩阵包括一个或者多个下级调度矩阵，和/或一个或者多个虚拟调度对象的类别标识；

所述虚拟调度对象实现与对应的调度实体的接口，所述接口至少包括控制类接口；

所述调度矩阵的方向标识用于标识当前调度矩阵并行调度或者串行调度。

6. 如权利要求 5 所述的作业调度系统，其特征在于，所述调度矩阵描述信息为静态配置文件，或者动态指令。

7. 如权利要求 5 所述的作业调度系统，其特征在于，所述调度矩阵的方向分别用横向和纵向表示；

如果一个调度矩阵的方向是横向，表明所述调度矩阵的下级调度矩阵或者虚拟调度对象均串行执行；

如果一个调度矩阵的方向是纵向，表明所述调度矩阵的下级调度矩阵或者虚拟调度对象均并行执行。

8. 如权利要求 5、6 或 7 所述的作业调度系统，其特征在于，所述调度执行单元包括：

一级调度矩阵定位模块，用于定位到一级调度矩阵；

方向标识获取模块，用于获取所定位的调度矩阵的方向标识；

下级调度矩阵访问模块，用于按照调度矩阵的方向访问所定位的调度矩阵的各下级调度矩阵；

最简矩阵判断模块，用于判断当前访问的下级矩阵是否为最简矩阵；

虚拟调度对象启动模块，用于在所述最简矩阵判断模块判断当前访问的下级矩阵是最简矩阵时，按照调度矩阵的方向启动各个虚拟调度对象；以及

调度矩阵定位模块，用于在所述最简矩阵判断模块判断当前访问的下级矩阵不是最简矩阵时，定位到本级调度矩阵；

所述最简矩阵为下级调度矩阵中只有虚拟调度对象的调度矩阵。

一种作业调度方法及系统

技术领域

[0001] 本发明属于计算机领域，尤其涉及一种作业调度方法及系统。

背景技术

[0002] 调度算法是指根据系统的资源分配策略所规定的资源分配算法。对于不同的的系统和系统目标，通常采用不同的调度算法。

[0003] 在大量作业的批处理系统中，为了照顾为数众多的短作业，常采用短作业优先的调度算法，平时处理事务时也是这样，分轻重缓急，根据事务权重的评估来进行优先排序。

[0004] 在大量终端用户的分时系统中，为了保证系统均具有合理的响应时间，应当采用轮转法进行调度，即如果无论当前的任务需要执行多久及多少资源，均每次都分配相同的执行时间段及相同的资源给它，以保证随时都有任务在进行中，充分提高效率。

[0005] 目前常见的调度方法有：

[0006] (1)先来先服务(FCFS)

[0007] 先来先服务(FCFS)是最简单的调度算法，按先后顺序对任务进行调度，即按照作业提交或进程变为就绪状态的先后次序，分派CPU。当前作业或进程占用CPU，直到执行完或阻塞才出让CPU(非抢占方式)，在作业或进程唤醒后(如I/O完成)并不立即恢复执行，通常等到当前作业或进程出让CPU。FCFS比较有利于长作业，而不利于短作业，有利于CPU繁忙的作业，而不利于I/O繁忙的作业。

[0008] (2)轮转法(Round Robin)

[0009] 该方法通常使得各个任务在就绪队列中的等待时间与享受服务的时间成正比例。

[0010] (3)多级反馈队列算法(Round Robin with Multiple Feedback)

[0011] 多级反馈队列算法是时间片轮转算法和优先级算法的综合和发展。该算法的基本运作模式如下：

[0012] 1. 设置多个就绪队列，分别赋予不同的优先级，如逐级降低，队列1的优先级最高。

[0013] 2. 每个队列执行时间片的长度也不同，规定优先级越低则时间片越长，如逐级加倍。

[0014] 3. 新进程进入内存后，先投入队列1的末尾，按FCFS算法调度；

[0015] 4. 若按队列1一个时间片未能执行完，则降低投入到队列2的末尾，同样按FCFS算法调度；

[0016] 5. 如此下去，降低到最后的队列，则按“时间片轮转”算法调度直到完成。

[0017] 6. 仅当较高优先级的队列为空，才调度较低优先级的队列中的进程执行。

[0018] 7. 如果进程执行时有新进程进入较高优先级的队列，则抢先执行新进程，并把被抢先的进程投入原队列的末尾。

[0019] 如上所述，当前的调度算法主要是从资源占用的角度入手，主要的考量是调度实体占用资源的时间，即调度的策略是根据调度实体的属性来决定，多个属性中最为重要的

属性就是资源占用的时间，根据这个资源占用的时间可以采用 FCFS、Round Robin、Round Robin with Multiple Feedback 等不同的处理方式。

[0020] 然而，随着应用的发展，越来越多的调度不仅仅只是根据调度属性中时间占用来决定分配资源的顺序，而且还需要根据调度任务之间的逻辑关系来决定资源分配的顺序。例如，ORACLE EBS ERP 系统有 200 多个子系统，所有子系统的表总量超过 1 万多，假设需要将 EBS ERP 系统中的数据表内容从一个服务器迁移到另外一台服务器上，则影响调度顺序的问题有很多，包括：

[0021] 1）各个子系统之间的先后顺序是什么？

[0022] 2）多个子系统可以分组运行吗？

[0023] 3）多组子系统可以并行运行吗？

[0024] 4）一个子系统的多个表运行是有依赖关系的，它们的先后顺序是什么？

[0025] 5）多个表之间可以并行运行吗？

[0026] 6）多个表之间可以分组吗？

[0027] 从上述的问题可以看出，在这种情况下影响调度的问题不是时间片，不是优先级，而是各个调度实体之间的关系，调度实体之间的逻辑关系决定了调度顺序和优先级。

发明内容

[0028] 本发明实施例提供一种作业调度算法，根据调度实体之间的逻辑关系确定调度的顺序和优先级，。

[0029] 本发明实施例是这样实现的，一种作业调度方法，所述方法包括下述步骤：

[0030] 接收调度矩阵描述信息，生成调度矩阵；

[0031] 根据所述调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象，并根据虚拟调度对象返回的结果更新所述调度矩阵；

[0032] 虚拟调度对象根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程；

[0033] 所述调度矩阵描述信息至少包含调度矩阵的结构信息，以及虚拟调度对象的类别；

[0034] 所述调度矩阵至少包括调度矩阵的标识，以及调度矩阵的方向标识；

[0035] 所述调度矩阵包括一个或者多个下级调度矩阵，和 / 或一个或者多个虚拟调度对象的类别标识；

[0036] 所述虚拟调度对象实现与对应的调度实体的接口，所述接口至少包括控制类接口；

[0037] 所述调度矩阵的方向标识用于标识当前调度矩阵并行调度或者串行调度。

[0038] 本发明实施例还提供一种作业调度系统，所述系统包括：

[0039] 调度矩阵生成单元，用于接收调度矩阵描述信息，生成调度矩阵；

[0040] 虚拟调度对象，用于根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程；

[0041] 调度执行单元，用于根据所述调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象，并根据虚拟调度对象返回的结果更新所述调度矩阵；

[0042] 所述调度矩阵描述信息至少包含调度矩阵的结构信息，以及虚拟调度对象的类别；

[0043] 所述调度矩阵至少包括调度矩阵的标识，以及调度矩阵的方向标识；

[0044] 所述调度矩阵包括一个或者多个下级调度矩阵，和/或一个或者多个虚拟调度对象的类别标识；

[0045] 所述虚拟调度对象实现与对应的调度实体的接口，所述接口至少包括控制类接口；

[0046] 所述调度矩阵的方向标识用于标识当前调度矩阵并行调度或者串行调度。

[0047] 本发明实施例通过矩阵方式进行作业调度，在调度过程中不仅关注资源的使用，而且考虑到了资源使用的结构系，使得很多复杂结构的调度可以很快完成。

附图说明

[0048] 图 1 是本发明实施例中一个调度矩阵的内存结构示例图；

[0049] 图 2 是本发明实施例提供的作业调度方法的实现流程图；

[0050] 图 3 是本发明实施提供的根据调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象的实现流程图；

[0051] 图 4 本发明实施例提供的作业调度系统的结构图；

[0052] 图 5 本发明实施例提供的作业调度系统中调度执行单元的结构图。

具体实施方式

[0053] 为了使本发明的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本发明进行进一步详细说明。应当理解，此处所描述的具体实施例仅仅用以解释本发明，并不用于限定本发明。

[0054] 在本发明实施例中，采用调度矩阵方式表达调度实体之间的逻辑关系，调度时根据调度矩阵信息确定调度实体的顺序和优先级，使得很多复杂结构的调度可以很快完成。

[0055] 在本发明实施例中，一个调度矩阵可以包括：

[0056] (1)一个或者多个下级调度矩阵；

[0057] (2)一个或者多个虚拟调度对象(Virtual Schedule Object，VSO)的类别标识；

[0058] (3)一个或者多个虚拟调度对象的类别标识，和一个或者多个下级调度矩阵。

[0059] 在本发明实施例中，一个调度矩阵可以是最简矩阵。最简矩阵是调度矩阵的特殊类型，其下级调度矩阵中只能有 VSO，不再包含其他下级调度矩阵。

[0060] 每个调度矩阵至少包括：

[0061] 1)MatrixID- 调度矩阵的标识，用来唯一标识一个调度矩阵；

[0062] 2)Direction- 调度矩阵的方向标识，用于指定当前调度矩阵并行调度或者串行调度。

[0063] 在本发明实施例中，调度矩阵的方向分别用横向和纵向表示。如果一个调度矩阵的方向是横向，则表明该调度矩阵的下级调度矩阵或者虚拟调度对象均串行执行。如果一个调度矩阵的方向是纵向，则表明该调度矩阵的下级调度矩阵或者虚拟调度对象均并行执行。

[0064] 在本发明实施例中，调度矩阵所调度的调度实体可以是进程、线程、函数、远程API、消息等，这些调度实体与虚拟调度对象通信。

[0065] 虚拟调度对象实现它与某类调度实体的接口，至少包括控制类接口，例如：

[0066] 如果调度实体是一个进程，则虚拟调度对象至少实现启动进程和停止进程的接口；

[0067] 如果调度实体是一个函数，则虚拟调度对象至少实现函数的调用接口；

[0068] 如果调度实体是一个消息，则虚拟调度对象至少实现消息的接收或发送接口。

[0069] 虚拟调度对象还可以实现它与某类调度实体的业务类接口，具体根据业务的需求来定义，例如：

[0070] 需要调度实体完成一次数据库中某张表的数据导出，则对应的调度实体的业务类接口必须实现发起数据库导出请求和接收导出结果应答两个业务接口。

[0071] 图1示出了本发明实施例中一个调度矩阵的内存结构示例，描述如下：

[0072] 1）一级调度矩阵方向为横向，MatrixID为1，包含3个二级调度矩阵，分别为调度矩阵2.1、调度矩阵2.2和调度矩阵2.3。

[0073] 2）二级调度矩阵2.1方向为竖向，包含2个三级调度矩阵2.1.1和2.1.2。

[0074] 3）三级调度矩阵2.1.1为竖向，包含2个虚拟调度对象的类别标识。

[0075] 4）三级调度矩阵2.1.2为横向，包含2个虚拟调度对象的类别标识。

[0076] 5）二级调度矩阵2.2方向为竖向，包含3个虚拟调度对象的类别标识。

[0077] 6)二级调度矩阵2.2方向为横向，包含3个三级调度矩阵2.3.1和2.3.2和2.3.3，每个三级调度矩阵中都包含3个虚拟调度对象的类别标识。

[0078] 图2示出了本发明实施例提供的作业调度方法的实现流程，详述如下：

[0079] 在步骤S201中，接收调度矩阵描述信息，生成调度矩阵；

[0080] 在本发明实施例中，调度矩阵描述信息可以是静态配置文件，也可以是用户发出的动态指令，包含：

[0081] （1）调度矩阵的结构信息，例如调度矩阵的层次、方向、调度矩阵的标识、最简调度矩阵中有多少个虚拟调度对象等；

[0082] （2）虚拟调度对象的类别，例如进程、线程、函数、远程API、消息和特定对象等。

[0083] 以下以可扩展标记语言（Extensible Markup Language，XML）为例描述一个调度矩阵：

[0084]

```
<Matrix id='Main_1'  Dir='V'>
  <Matrix id='Sub_1.1 Dir='H'>
      <VSO  Type='Sucker' />
  </Matrix>
  <Matrix id='Sub_1.2 Dir='H'>
      <VSO  Type='Parser' />
  </Matrix>
  <Matrix id='Sub_1.3 Dir='H'>
      <VSO  Type='Loader' />
  </Matrix>
</Matrix>
```

[0085] 上述示例描述一个一级调度矩阵，方向为纵向，包含 3 下级矩阵，方向为横向，

[0086] 每个子矩阵包含一个虚拟调度对象。这些虚拟调度对象有不同的类别，分别为 Suker（抽取类）、Parser（分析类）、Loader（加载类）。

[0087] 在本发明实施例中，调度矩阵生成后，可以根据接收到的外部指令，或者调度实体的启动或者运行的反馈信息不断动态调整。

[0088] 在步骤 S202 中，根据该调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象，并根据虚拟调度对象返回的结果更新该调度矩阵；

[0089] 在步骤 S203 中，虚拟调度对象根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程。

[0090] 作为本发明的一个实施例，当调度实体正常完成调度任务之后，调度实体通知虚拟调度对象任务完成，虚拟调度对象向调度实体发起退出命令，调度实体收到退出命令后将退出。

[0091] 当一个最简矩阵中的所有调度实体都退出后，将启动一下个矩阵的调度。

[0092] 如果一个调度实体不能够正常的完成任务，虚拟调度对象可以通过超时判断的方式来确定是否需要向该调度实体强制发起退出命令，并认为这个调度实体已经退出。

[0093] 图 3 示出了本发明实施提供的根据该调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象的流程，详述如下：

[0094] 在步骤 S301 中，定位到一级调度矩阵；

[0095] 在步骤 S302 中，获取该调度矩阵的方向标识；

[0096] 在步骤 S303 中，按照调度矩阵的方向访问各下级调度矩阵。如果调度矩阵的方向标识是横向，则依次访问各个下级调度矩阵，如果调度矩阵的方向标识是纵向，则同时访问各下级调度矩阵；

[0097] 在步骤 S304 中，判断下级矩阵是否为最简矩阵，是则执行步骤 S306，否则执行步骤 S305；

[0098] 在步骤 S305 中，定位到本级调度矩阵，返回执行步骤 S302 ；

[0099] 在步骤 S306 中，按照调度矩阵的方向启动各个虚拟调度对象。如果调度矩阵的方向标识是横向，则依次启动各个虚拟调度对象，如果是纵向则同时启动各个虚拟调度对象。

[0100] 以下通过以具体的应用场景为例对本发明实施例的具体实现进行描述 ：

[0101] 应用场景 1 ：数据库日志复制的调度

[0102] 数据库的日志中记录了针对数据库的所有历史记录，可以认为数据库的日志是数据所有操作的录像。复制这些日志操作，可以到另外的数据库恢复和还原该数据库。

[0103] 在本发明实施例的示例中，将该数据库日志复制操作分为日志抽取、日志分析和日志装载三个组件之间的协同。

[0104] 这个三个组件就是本发明实施例的调度对象，本发明实施例可以设计一个层叠矩阵来调度这三个调度对象 ：

[0105]

```
<Matrix id='Main_1'  Dir='V'>
  <Matrix id='Sub_1.1 Dir='H'>
      <VSO  Type='Sucker' />
```

[0106]

```
  </Matrix>
  <Matrix id='Sub_1.2 Dir='H'>
      <VSO  Type='Parser' />
  </Matrix>
  <Matrix id='Sub_1.3 Dir='H'>
      <VSO  Type='Loader' />
  </Matrix>
</Matrix>
```

[0107] 当根据上述调度矩阵的描述生产调度矩阵后，将在顶级矩阵下并发生产三个下级矩阵，每个下级矩阵对应有不同的虚拟调度对象，各个虚拟调度对象负责不间断的和同名的调度实体通信，这样就可以完成整个任务的调度工作。

[0108] 应用场景 2 ：数据库表数据迁移的调度

[0109] 典型的企业管理软件（ERP）通常有几十到几百个子系统。在数据库中通常有几千到几万张表，这些子系统之间有相互的依赖关系，例如 GL （总账系统）通常是被其他的系统依赖。

[0110] 如果需要将一个 ERP 系统的数据库表中的数据迁移到另外的一个数据库中，就既需要考虑并发的性能问题，又需要考虑各个表数据之间的依赖关系问题。

[0111] 以下将给出一个简单的模型，通过本发明实施例完成类似的调度工作 ：

[0112] 1）示例中迁移 5 个子系统，分别为 SYS1，SYS2，SYS3，SYS4，SYS5 ；

[0113] 2）示例中对每个子系统迁移两张表，迁移方式为串行 ；

[0114] 3）示例中子系统迁移方式为：SYS1 和 SYS2 可以同时迁移，SYS3，SYS4；SYS5 可以同时迁移，但是必须在 SYS1 和 SYS2 迁移完成后才能进行，也就是说 SYS1+SYS2 和 SYS3+SYS4+SYS5 之间是串行的关系；

[0115] 4）每张表的迁移通过几个调度实体来完成，分别为 MigOut（表数据移出对象）和 MigIn（表数据移入对象），对应该两个调度实体，有两类虚拟调度对象，分别为 MIGOUT 和 MIGIN。

[0116] 以下给出对应的调度矩阵的描述内容：

[0117]

```
<Matrix id='Main_1'>
  <Matrix  id='Sub_1.1',  Dir='H'>
    <Matrix id='Sub_1.1.1',  Name='SYS1', Dir='V'>
      <Matrix id='Sub_1.1.1.1',  Name='SYS1.T1', Dir='V'>
        <VSO TYPE='MIGOUT'/>
        <VSO TYPE='MIGIN'/>
      </Matrix>
      <Matrix id='Sub_1.1.1.2',  Name='SYS1.T2', Dir='V'>
        <VSO TYPE='MIGOUT'/>
        <VSO TYPE='MIGIN'/>
      </Matrix>
    </Matrix>
    <Matrix id='Sub_1.1.2',  Name='SYS2', Dir='V'>
      <Matrix id='Sub_1.1.2.1',  Name='SYS2.T1', Dir='V'>
        <VSO TYPE='MIGOUT'/>
        <VSO TYPE='MIGIN'/>
      </Matrix>
      <Matrix id='Sub_1.1.2.2',  Name='SYS2.T2', Dir='V'>
        <VSO TYPE='MIGOUT'/>
        <VSO TYPE='MIGIN'/>
      </Matrix>
    </Matrix>
  </Matrix>
  <Matrix  id='Sub_1.2 Dir='H'>
    <Matrix id='Sub_1.2.1',  Name='SYS3', Dir='V'>
      <Matrix id='Sub_1.2.1.1',  Name='SYS3.T1', Dir='V'>
        <VSO TYPE='MIGOUT'/>
        <VSO TYPE='MIGIN'/>
```

[0118]

```
    </Matrix>
    <Matrix id='Sub_1.2.1.2',  Name='SYS3.T2', Dir='V'>
      <VSO TYPE='MIGOUT'/>
      <VSO TYPE='MIGIN'/>
    </Matrix>
  </Matrix>
  <Matrix id='Sub_1.2.2',  Name='SYS4', Dir='V'>
    <Matrix id='Sub_1.2.2.1',  Name='SYS4.T1', Dir='V'>
      <VSO TYPE='MIGOUT'/>
      <VSO TYPE='MIGIN'/>
    </Matrix>
    <Matrix id='Sub_1.2.2.2',  Name='SYS4.T2', Dir='V'>
      <VSO TYPE='MIGOUT'/>
      <VSO TYPE='MIGIN'/>
    </Matrix>
  </Matrix>
  <Matrix id='Sub_1.2.3',  Name='SYS5', Dir='V'>
    <Matrix id='Sub_1.2.3.1',  Name='SYS5.T1', Dir='V'>
      <VSO TYPE='MIGOUT'/>
      <VSO TYPE='MIGIN'/>
    </Matrix>
    <Matrix id='Sub_1.2.3.2',  Name='SYS5.T2', Dir='V'>
      <VSO TYPE='MIGOUT'/>
      <VSO TYPE='MIGIN'/>
    </Matrix>
  </Matrix>
 </Matrix>
</Matrix>
```

[0119] 根据上述的调度矩阵描述生产调度矩阵后，就可以按照调度矩阵的描述内容进行调度执行。

[0120] 图 4 示出了本发明实施例提供的作业调度系统的结构，为了便于说明，仅示出了与本发明实施例相关的部分。

[0121] 调度矩阵生成单元 41 接收调度矩阵描述信息，生成调度矩阵。

[0122] 虚拟调度对象 42 根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程。

[0123] 调度执行单元 43 根据该调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象 42，并根据虚拟调度对象 42 返回的结果更新该调度矩阵。

[0124] 在本发明实施例中，调度矩阵描述信息至少包含调度矩阵的结构信息，以及虚拟调度对象的类别。

[0125] 在本发明实施例中，调度矩阵至少包括调度矩阵的标识，以及调度矩阵的方向标识。调度矩阵的方向标识用于标识当前调度矩阵并行调度或者串行调度。

[0126] 在本发明实施例中，调度矩阵包括一个或者多个下级调度矩阵，和 / 或一个或者多个虚拟调度对象的类别标识；

[0127] 在本发明实施例中，虚拟调度对象实现与对应的调度实体的接口，该接口至少包括控制类接口。

[0128] 作为本发明的一个优选实施例，调度矩阵描述信息可以为静态配置文件，或者动态指令。

[0129] 作为本发明的一个优选实施例，调度矩阵的方向可以分别用横向和纵向表示。

[0130] 如果一个调度矩阵的方向是横向，表明该调度矩阵的下级调度矩阵或者虚拟调度对象均串行执行。

[0131] 如果一个调度矩阵的方向是纵向，表明该调度矩阵的下级调度矩阵或者虚拟调度对象均并行执行。

[0132] 作为本发明的一个实施例，当调度实体正常完成调度任务之后，调度实体通知虚拟调度对象 42 任务完成，虚拟调度对象 42 向调度实体发起退出命令，调度实体收到退出命令后将退出。

[0133] 当一个最简矩阵中的所有调度实体都退出后，调度执行单元 43 将启动一下个矩阵的调度。

[0134] 如果一个调度实体不能够正常的完成任务，虚拟调度对象 42 可以通过超时判断的方式来确定是否需要向该调度实体强制发起退出命令，并认为这个调度实体已经退出。

[0135] 图 5 示出了本发明实施例中调度执行单元的结构，为了便于说明，仅示出了与本发明实施例相关的部分。

[0136] 调度启动时，一级调度矩阵定位模块 431 定位到一级调度矩阵。

[0137] 方向标识获取模块 432 获取所定位的调度矩阵的方向标识。

[0138] 下级调度矩阵访问模块 433 按照调度矩阵的方向访问所定位的调度矩阵的各下级调度矩阵。

[0139] 最简矩阵判断模块 434 判断当前访问的下级矩阵是否为最简矩阵。

[0140] 在最简矩阵判断模块 434 判断当前访问的下级矩阵是最简矩阵时，虚拟调度对象启动模块 435 按照调度矩阵的方向启动各个虚拟调度对象。

[0141] 在最简矩阵判断模块 434 判断当前访问的下级矩阵不是最简矩阵时，调度矩阵定

位模块 436 定位到本级调度矩阵，继续进行调度。

[0142] 在本发明实施例中，最简矩阵为下级调度矩阵中只有虚拟调度对象的调度矩阵。

[0143] 本发明实施例通过矩阵方式描述调度实体之间的结构关系，同时具有树和队列的特性，具有极高的普适性，可以满足绝大多数调度场景的需求，将调度实体通过虚拟调度对象在矩阵中映射，可以通过编写不同的虚拟调度对象来适应不同的调度实体，使得调度有很好的移植性和扩展性，在调度时将并行和串行转化描述为调度矩阵的纵向和横向属性，使得原来很复杂的资源分配方式变得极其简单，设计调度矩阵和执行调度矩阵的算法难度大幅度下降。

[0144] 另外，本发明实施例对虚拟调度对象的调度接口只有很少的限制，调度实体在支持 START 和 STOP 两个接口后，就可以纳入到调度体系中，极大地降低了对现有系统功能模块进行修改和升级的要求。

[0145] 综上所述，本发明实施例通过矩阵方式进行作业调度，在调度过程中不仅关注资源的使用，而且考虑到了资源使用的结构系，使得很多复杂结构的调度可以很快完成。

[0146] 以上所述仅为本发明的较佳实施例而已，并不用以限制本发明，凡在本发明的精神和原则之内所作的任何修改、等同替换和改进等，均应包含在本发明的保护范围之内。

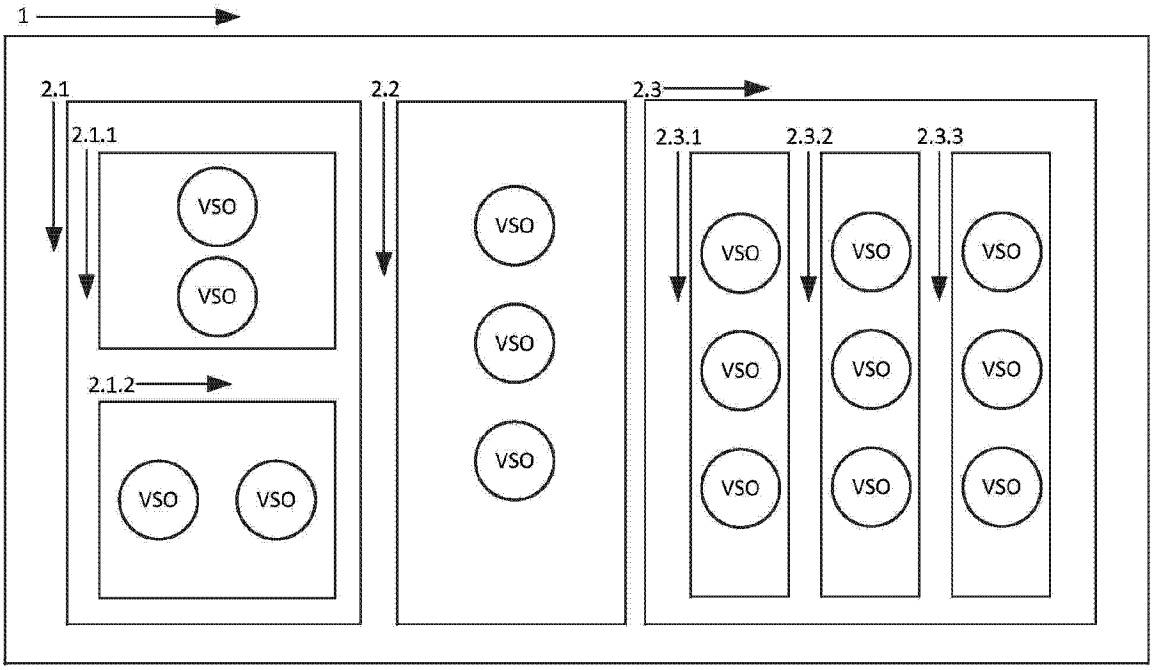

1
2.1
2.1.1
VSO
VSO
2.1.2
VSO
VSO
2.2
VSO
VSO
VSO
2.3
2.3.1
2.3.2
2.3.3
VSO
VSO
VSO
VSO
VSO
VSO
VSO
VSO
VSO

图 1

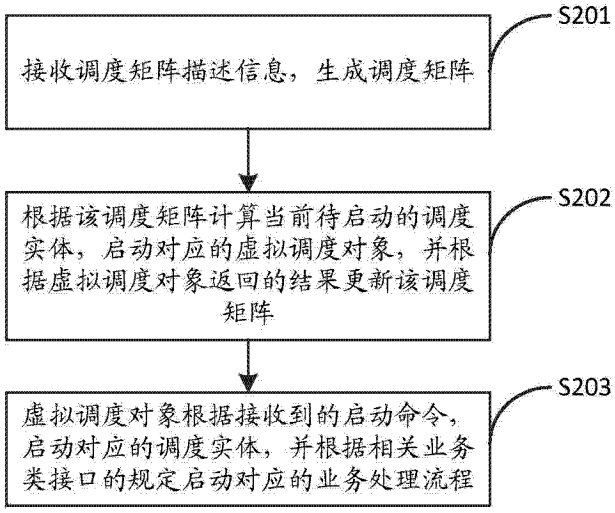

接收调度矩阵描述信息，生成调度矩阵
S201
根据该调度矩阵计算当前待启动的调度实体，启动对应的虚拟调度对象，并根据虚拟调度对象返回的结果更新该调度矩阵
S202
虚拟调度对象根据接收到的启动命令，启动对应的调度实体，并根据相关业务类接口的规定启动对应的业务处理流程
S203

图 2

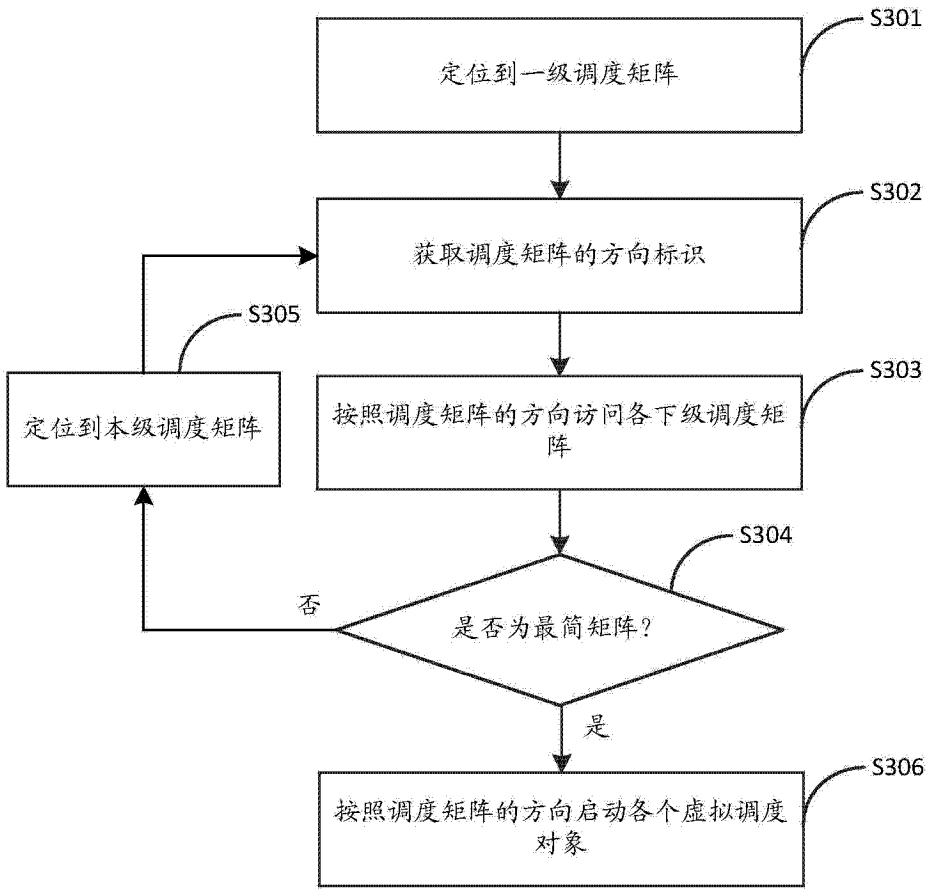

S301
定位到一级调度矩阵
S302
获取调度矩阵的方向标识
S303
按照调度矩阵的方向访问各下级调度矩阵
S304
是否为最简矩阵?
否
是
S305
定位到本级调度矩阵
S306
按照调度矩阵的方向启动各个虚拟调度对象

图 3

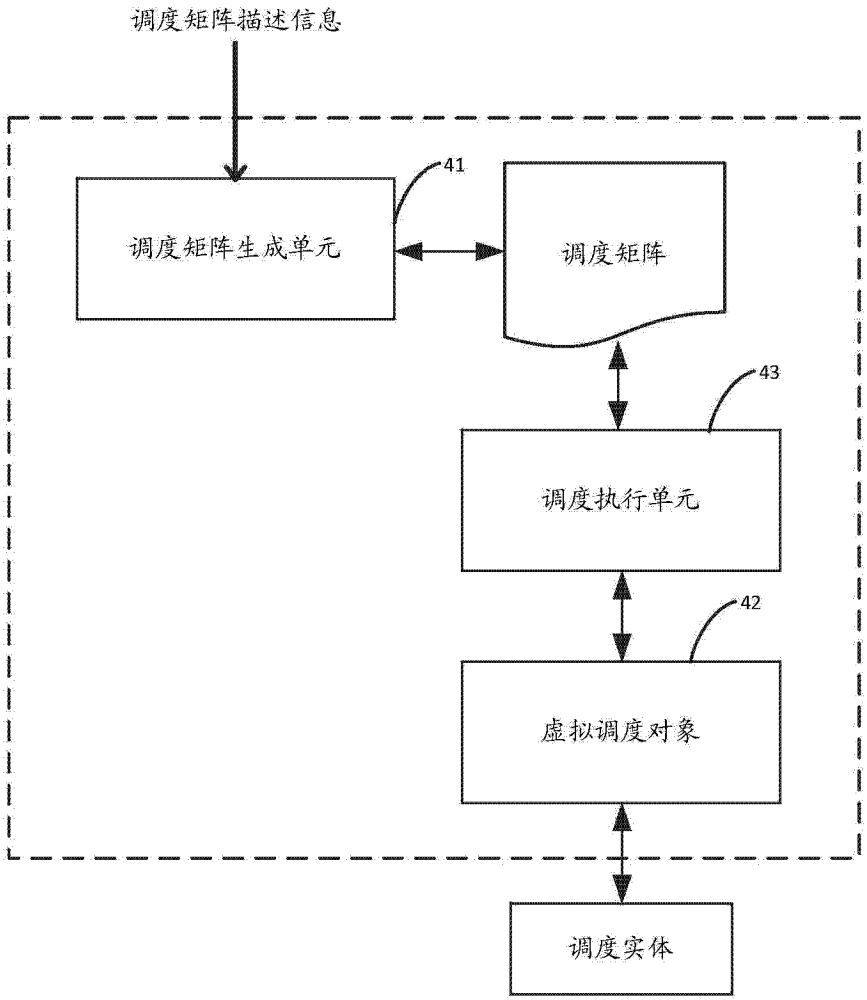


图 4

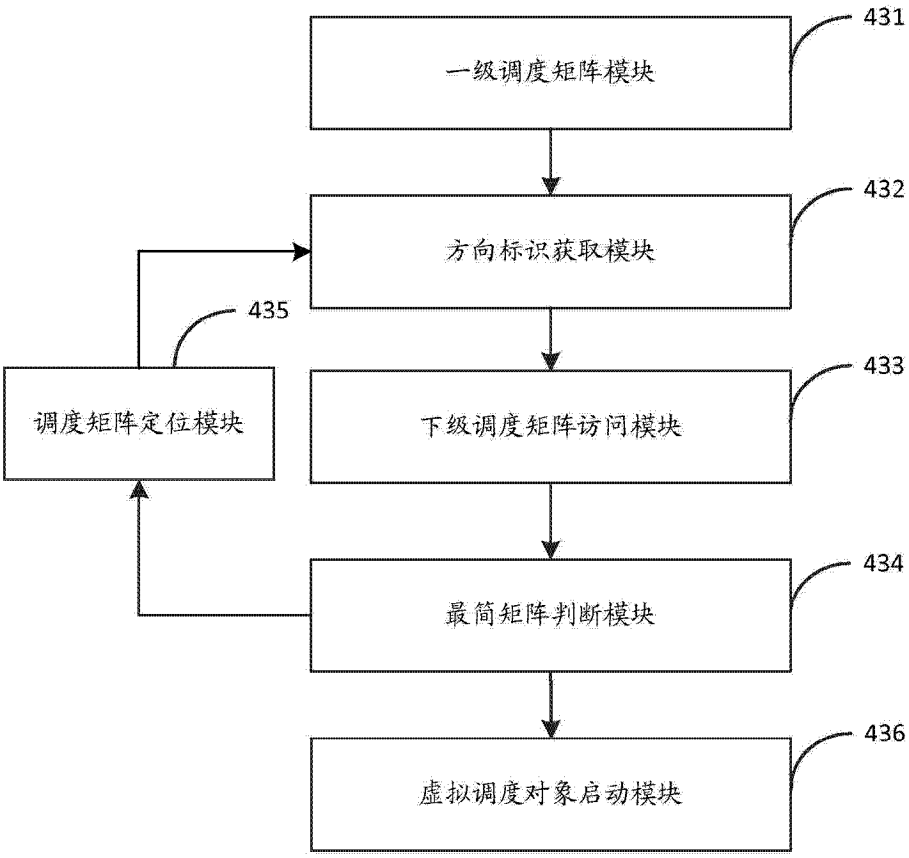

一级调度矩阵模块
431
方向标识获取模块
432
调度矩阵定位模块
435
下级调度矩阵访问模块
433
最简矩阵判断模块
434
虚拟调度对象启动模块
436

图 5