文章编号:1006-3080(2008)01-0096-07

并行分布环境下的黑板模型

冯少荣[1,2], 肖文俊[1]
(1. 华南理工大学计算机科学与工程学院,广州 510640;
2. 厦门大学信息科学与技术学院,福建 厦门 361005)

摘要:黑板模型支持并行性,它是分布式和并行编程可用的强有力的模型之一。在一个需要并行性和分布式编程的系统中,黑板模型有助于组织和概念化并发性及通信。本文着重分析了黑板模型的结构、构造方法、控制策略。基于 CORBA(Common Object Request Broker Architecture)对象和全局对象研究了黑板和知识库的实现。最后,通过一个具体实例的实现方法和过程,说明了黑板模型解决分布式和并行编程问题的可行性。

关键词:黑板模型; 并行分布; 知识库; 问题解决者
中图分类号: TP311 **文献标识码**: A

Blackboard Model on Parallel Distributed Environments

FENG Shao-rong[1,2], *XIAO Wen-jun*[1]
(1. *School of Computer Science and Engineering, South China University of Technology, Guangzhou* 510640, *China*; 2. *College of Information Science and Technology, Xiamen University, Xiamen* 361005, *Fujian, China*)

Abstract: The blackboard model supports parallelism. It is one of the most powerful model for distributed and parallel programming. In parallel and distributed programming system, the blackboard model is helpful to organizing and conceptualization concurrency and communicating. This paper researches the structure of the blackboard model, construction method, control strategies. Based on CORBA (Common Object Request Broker Architecture) objects and global objects, the implementing of blackboard and knowledge source are studied. At last, by using implementation method and process of a typical example, this paper illustrates the feasibility that blackboard models are used to solve distributed and parallel programming.

Key words: blackboard model; distributed parallel; knowledge source; problem solver

在并行编程[1~5]中,基本的目标之一就是把程序要做的工作划分成一个任务集合,而任务集合中的任务要尽可能地并行执行。这个目标并没有提出具体标准。要找出正确的工作分解结构 WBS (Work Breakdown Structure)来支持并行性并能产生正确有效的结果,这无疑是个挑战。使用模型化和结构化的方法可以获取可接受的 WBS。实际上,只要尽可能自然地对问题和解中每一个对象或过程进行建模,就会找到必要的并行性。模型同时也识别出问题和解中存在并行性的位置。黑板体系结构可以帮助实现这个建模过程。黑板模型也有助于在一个需要并行性和分布式编程的系统中实现并发操

收稿日期:2007-01-04
基金项目:福建省自然科学基金项目(A0310008); 福建省高新技术研究开放计划重点项目(2003H043)
作者简介:冯少荣(1964-),副教授,在职博士生,主要研究方向:并行分布数据库、数据仓库、数据挖掘。E-mail:shaorong@xmu.edu.cn

作和通信。

1　黑板模型

黑板模型[6~21]适用于解决协同工作中遇到的问题。黑板用于处理两个或多个基于软件的问题解决者之间的记录、协调和通信事务。因此,黑板模型中有两种主要的组件类型:黑板和问题解决者。

1.1　黑板

黑板是所有问题解决者都访问的集中对象。问题解决者可以读取并改变黑板上的内容,黑板上的内容是时刻变化的。黑板上的初始化内容包括将要解决的问题。黑板上可能包括的其他信息表示了问题的初始状态、约束关系、目标和目的。当问题解决者求解问题时,所有中间结果、假设和结论都将记录在黑板上。任何一个问题解决者公布于黑板上的中间结果都可能对其他正在阅读黑板的问题解决者有所帮助。实验性的解也提交到黑板上,如果这些解被认为不够充分,就会被删除。同时,其他的解被提交。问题解决者之间不能直接通信,而是使用黑板来交换部分结果和彼此的发现。黑板可以被配置为仲裁者,在解到达时通知问题解决者,或者告知它应该开始或结束工作。黑板是主动对象,而不仅仅是存储位置。在某些情况下,由黑板调用哪些问题解决者以及接受或拒绝哪些内容。同时黑板也可以组织问题解决者的逐次结果或中间结果。黑板还可以翻译或解释一个问题解决者集合的工作,以便将工作结果提供给另外一个问题解决者集合使用。

1.2　问题解决者

问题解决者是一种软件,这种软件在某些领域或问题域中具有特定的知识或处理能力。在黑板模型中,这些问题解决者叫做知识库 KS(Knowledge Source)。使用黑板解决问题时需要多个知识库,而且每个知识库都拥有不同的关注焦点或专业领域。如果问题可以划分成多个任务,并且每个任务都可以独立或半独立地解决,那么黑板是合适的方法。在基本的黑板配置中,每个问题解决者处理问题的一个不同部分。每个问题解决者都只会看到问题中它所熟悉的部分。如果问题的任意部分的解依赖问题其他部分的解或部分解,可以借助黑板来协调问题解决者之间的通信,并整合部分方案。黑板的不同问题解决者之间并不要求是同构的,每个问题解决者都可以通过不同的技术来实现。此外,不同的问题解决者可以使用完全不同的问题解决范例。

黑板模型并不指定黑板使用任何特定的结构或布局,也不会建议如何构造知识库。实际上,黑板的结构是问题相关的。知识库的实现也根据要解决的问题来定。黑板框架是一个概念模型,它描述了黑板和知识库的关系,而没有描述黑板和知识库的结构。黑板模型没有要求知识库的数量和目标。黑板可以是一个单独的全局对象,也可以是一个在多个计算机上有组件的分布式对象。黑板系统可以由多个黑板组成,每个黑板负责处理初始问题的一部分。这使得在解决问题过程中,黑板成了一个极其复杂的模型。黑板模型支持并行编程和分布式编程。因为知识库可以同时执行,每个知识库处理问题中属于它的一部分。知识库可以通过单独的线程或者在相同或不同计算机上运行的独立进程来实现。

黑板可以分段以允许多个知识库的并发访问。黑板支持并发读互斥写、互斥读互斥写、多指令流多数据流[1~5]。

当知识库用分离的线程实现时,可以把黑板用一个全局对象或对象的集合来实现。因为线程共享同样的地址空间,所以每个线程级的知识库都可以访问用全局对象或对象族来实现的黑板。如果知识库是通过运行在相同或不同计算机上的分离的进程来实现的,那么,黑板就可以用一个 CORBA[22]对象或 CORBA 对象集来实现。由于 CORBA 对象支持并行和分布的计算模型,因此,在这里黑板被实现成一种分布式共享存储器,使用 CORBA 来支持在不同地址空间中执行的任务对黑板的访问。这些任务可以是并行虚拟机 PVM(Parallel Virtual Machine)任务,也可以是传统的 fork-exec 函数调用产生的任务,还可以是新的 posix_spawn( ) 函数调用产生的任务。

2　黑板模型的构造

2.1　构造黑板的方法

构造黑板的方法不止一种。但是,大多数黑板都具有相同的固定特征和属性。黑板的初始化内容通常包含待解决问题的解空间的某种划分。解空间包含问题的所有部分解和完全解,有时候组织成分层结构。知识库同时分别处理分层结构中的各个层次。解空间也可以组织成图表,图表中每个结点表示解的某个部分,而每条边表示了两个部分解的关系。解空间还可以表示一个或多个矩阵,矩阵的每个元素包含了一个解或一个部分解。在黑板体系结构中,解空间的表示法是个很重要的组件。解空间如何划分常常由问题的性质来决定。除了解空间组

件外,黑板通常还有一个或多个规则(启发式)组件,用于确定部署哪个知识库和接受或拒绝哪些解。在解空间分层结构中,规则组件也可以用于把部分解从一个层次转换到另一个层次。规则组件还可以提高知识库方法的优先级,如果有些知识库可能无解,黑板就可以撤销这个知识库集,以便有利于另一个集合的工作。黑板还可以用规则组件来建议知识库在已有的部分假设基础上采用一个更适合的假设。除了解空间和规则组件外,黑板常常还包括初始值、约束值和辅助目标。在某些情况下,黑板还包括一个或多个事件队列,用于捕获来自问题空间和知识库的输入。图1显示了基本的黑板体系结构的逻辑配置图。图中如果KS是一个进程,通信可以通过网络或进程间通信IPC(Interprocess communication)。如果KS是一个线程,通信可以通过传递参数实现。黑板包含许多段,每个段都有多种实现方法。这表明黑板并不仅仅是几个全局的存储器或传统的数据库。图1中显示了大多数黑板常有的核心组件。但黑板结构并不只限于这些组件。黑板其他的一些有用组件包括问题的上下文模型和域模型,这些组件可以在解空间中对问题解决者进行导航。C++[23]提供的对面向对象设计和编程的支持能很好地满足黑板模型的灵活性要求。大多数黑板体系结构可以使用C++中的类来建模。在黑板模型的实现过程中,利用了C++容器类和标准算法的优势。除了内置类以外,还要为黑板中用到的互斥锁和其他同步变量构造接口类。因为多个知识库可以同时访问黑板,所以黑板是临界区,对它的访问需要同步。因此,除了黑板包含的其他组件外,还需要对黑板使用同步对象。

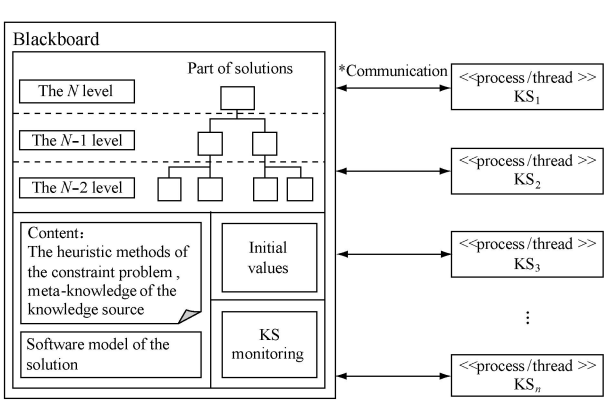

图1 基本黑板体系结构的逻辑配置图

Fig. 1 Logic configuration graph of basic blackboard architecture

2.2 知识库构造方法

知识库可以表示成对象、过程、规则集和逻辑断言,在某些情况下也可表示成完整的程序。知识库包含条件部分和动作部分。若黑板包括的某些信息满足了知识库的条件部分,那么知识库的动作部分就会被激活。如果知识库是一个对象,那么知识库的动作部分可以通过对象的方法来实现。知识库的条件部分可以当成对象的数据成员。一旦对象处于一个特定的状态,对象的动作部分就会被激活。为了简化,可以把知识库与线程或进程匹配起来,这样,每个线程中只有一个知识库,或者每个进程中只有一个知识库。当黑板使用PVM任务时,一个知识库就相当于一个PVM任务。图2显示了一个知识库结构的逻辑配置图。

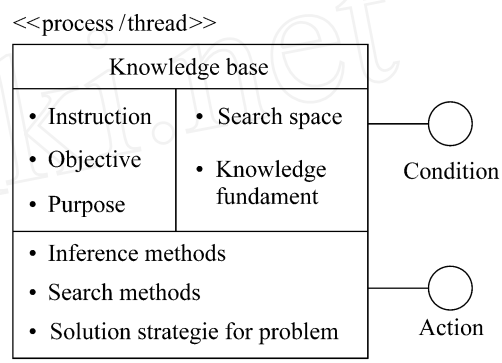

图2 知识库的逻辑配置

Fig. 2 Logic configuration of knowledge source

每个知识库的条件部分由黑板更新。一些知识库的动作部分更新黑板。图2中进程空间和知识库之间或线程空间和知识库之间存在意义对应的关系。知识库的一个重要属性就是它的结构。每个知识库都是一个专家,并且独立于其他问题解决者。这显示了并行程序的一个必需属性。理想情况下,并行程序中的任务能够并发地操作,而不需要太多与其他任务的交互。这正是黑板模型中的情况知识库独立行动,而任何主要的交互都通过黑板进行。所以从知识库的角度来看,它是单独行动的,从黑板取得额外的信息,并把它的发现记录在黑板上。它并不知道其他知识库的行为,也不知道它们的策略和结构。在黑板模型中,问题被划分给许多自治或半自治的问题解决者,这是黑板模型优于其他模型的地方。在最灵活的配置中,知识库是智能Agent。Agent将是完全自满足的,并与黑板尽可能少的交互。智能Agent表明了最大尺度并行的最大机会。

2.3 黑板控制策略

在黑板的实现中,激活知识库的并发控制可以分为几层。在最底控制层,知识库的同步方案必须保证黑板的完整性。因为黑板是一个共享并可修改

的资源,所以黑板是一个临界区。在并行环境中,必须协调知识库的读访问和写访问。这些协调和同步机制可以使用文件锁、信号量、互斥锁等。这一控制层并不直接包含在知识库正在求的解中。这是控制的实用程序层,应独立于黑板要解决的问题。为了实现黑板的并发性,这一层可以选择互斥读互斥写、并发读互斥写、互斥读并发写、并发读并发写 4 种并行访问黑板物理实现方法中的一种或几种,这些方法将用于知识库的算法或者试探法中。黑板的分段方式决定了适合使用哪种并发机制。最灵活的并发读并发写可以基于黑板的结构来完成。

控制层的下一层为集中/注意层,它包含了在求解过程中选择哪些知识库,并确定集中在问题的哪个特定领域,根据该领域选择相应的知识库。这一层计算问题的初始条件,然后控制使用哪些知识库和从哪里开始。黑板知道哪些知识库可用,并且通常知识库会接收信息和参数。这些信息和参数告诉知识库如何开始或在解空间中从哪里开始搜索。对于并行实现,这一层确定了并行性的基本模型(问题解决者的分布)。对于黑板来说,这是多指令流多数据流模型,因为每个知识库/问题解决者有它专门的领域。若根据问题的性质决定使用单指令流多数据流模型,那么控制层可能产生 N 个同样的知识库,但是给每个知识库传递不同的参数。

在集中/注意层的下一层包含了确定如何处理写到黑板上的解或部分解。这一层将确定知识库是否停止工作或者生成的解是被接受、不被接收或部分被接受等。这一层对黑板和所有的部分解和实验解完全可见。除了黑板的配置和知识库的结构,黑板模型还需要设置一个控制组件,但是并没有指定如何构造这些控制组件。控制组件可以作为黑板的一部分,也可以由知识库来实现,还可以由黑板结构外部的模块来实现,或者是使用这几种方法的组合来实现。控制组件有助于管理知识库的集体搜索策略,并监督实验解和部分解以保证知识库不会采取不实际的策略。控制组件还能为问题选择最好或最合适的知识库。图 3 显示了一个黑板结构中可能的控制配置和它们的分层。

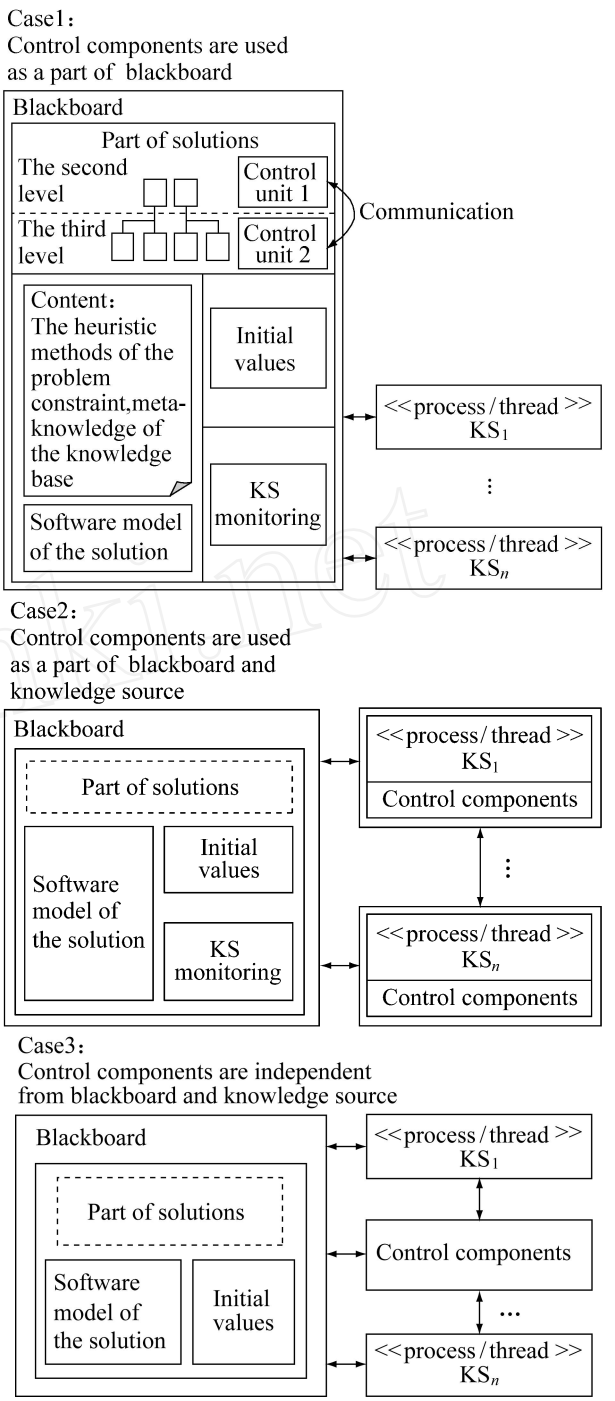

图 3　黑板体系结构的控制

Fig. 3　Control of blackboard architecture

3　黑板和知识库的实现

3.1　基于 CORBA 对象实现的黑板和知识库

在 Intranet 或 Internet 环境中实现知识库时,或者出于模块性和封装等目的在分离的进程里实现知识库时,选择基于 CORBA 的黑板是合适的。

CORBA 对象是一个平台无关的分布式对象。运行于同一机器上的进程可以访问 CORBA 对象,而连接到同一网络上的不同机器上运行的进程也可以访问 CORBA 对象。在 PVM 环境中,程序被划分成一些进程,这些进程可以运行在同一计算机上,也可以运行在不同计算机上。因此,在 PVM 环境中,可以选择使用 CORBA 对象,接下来要考虑的就是共享存储器方法。由于 CORBA 对象可以对任何非分布式对象表示的事务进行建模,所以可以共享

访问 CORBA 对象的 PVM 任务,可以访问容器对象、框架对象、模式对象、域对象以及任何实体对象。若要 PVM 任务可以访问黑板对象,可以使用消息传递模型增补对复杂对象的共享访问功能。除了 PVM 任务可以访问分布式 CORBA 对象外,由 posix_spawn( )函数或 fork-exec( )函数产生的传统任务也可以访问 CORBA 对象。由 posix_spawn( )函数或 fork-exec( )函数创建的任务位于同一机器分离的地址空间上,而 CORBA 对象可以位于任何机器上。CORBA 黑板实现支持的知识库实现方法有:在 PVM 任务实现、在传统 UNIX/Linux 任务中实现或在分离的线程中用 POSIX(Protable Operation System Inferface)线程库实现。构造函数中产生的任务类型决定了黑板与 POSIX thread、传统的 UNIX/Linux 进程还是 PVM 的任务合作。

当任务运行在分布的计算机上时,在 PVM 任务中实现知识库或问题解决者是非常有用的。每个知识库都可以发挥一台特定计算机所拥有的任何专用资源的优势。PVM 任务也可以用于有多个处理机的单个计算机。虽然黑板可以通过与端口的连接来实现,但也可以只使用传统的 UNIX/Linux 进程来包含知识库。当在标准的 UNIX/Linux 进程中创建知识库并且计算机有多个处理机时,那么知识库就可以并行地运行在可用的处理器上。明显地,如果知识库的数量大于处理器,那么多任务是必要的。

3.2 基于全局对象实现的黑板和知识库

当知识库可以在同一进程和相同计算机上实现时,多线程提供了优秀的解决方案。因为使用多线程效率高、开销小、易于使用和启动,多个线程之间的通信也很方便,因为线程共享相同的地址空间并且可以使用全局变量。实际上,这样实现的黑板会被实例化为一个全局对象,可以提供给进程中的所有线程访问。当知识库实现为单个程序中的线程时,不需要进程间通信、套接字通信或任何其他种类的网络通信。同时,CORBA 协议的附加层也不再需要,并且对象可以只设计成规则的 C++类。如果程序运行在有多个处理器的机器上,那么线程就可以并发地运行在可用的多个处理器上。黑板的线程配置在对称多处理系统和大规模并行处理器系统中非常具有吸引力,通常情况下,线程会达到最好的性能。因为线程的开销不像传统的 UNIX/Linux 那么多,可移植操作系统接口线程库(或 Pthread 库)虚拟地提供了知识库创建和管理所需的各种事务。因为黑板在多线程环境中实现,所以需用 Pthread 互斥锁和条件变量来同步对黑板的访问。互斥锁和条件变量可以封装在接口类中,Pthread_cond_signal( )和 pthread_cond_broadcast( )可以用来协调和同步知识库执行的工作。因为黑板创建了线程,所以黑板易于访问每个知识库的线程 id。黑板在必要的情况下可以使用 pthread_cancel( )删除一个线程,使用 pthread_join( )来同步各种知识库。

4 黑板示例

用黑板实现一个基于软件的课程咨询系统,为大学生解决典型的课程安排问题,学生使用基于黑板技术的实时课程建议软件来解决问题。黑板可以及时地对学生的学术记录和目前进行或停止的课程进行实时的访问,能够访问学生的学位计划、大学对该学位的计划要求、学生可用的安排和学生的目标等。这些条目中的每一个都可以用 C++类和 CORBA 类来建模,并且它们组成了黑板的组件。为了简化黑板例子,只看下列 4 个知识库(通用需求顾问、主修课顾问、选修课顾问、副修课顾问)。图 4 反映了 4 个知识库并发读写黑板分段的情形,图 5 的 UML 活动图显示了黑板和知识库的同步。

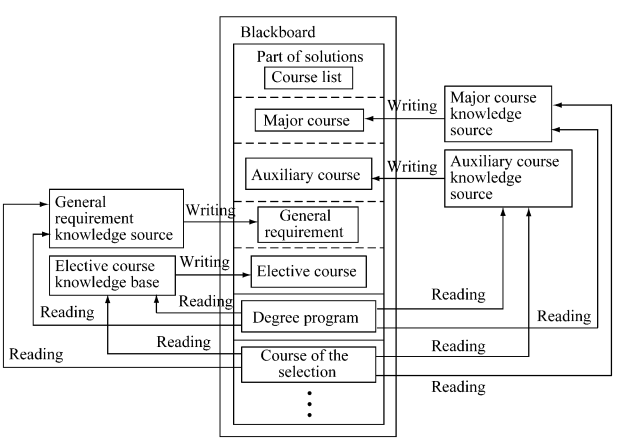

图 4 知识库并发读写黑板的情形

Fig. 4 Situation of concurrent read and write blackboard

案例实现过程算法描述:

黑板类的两个必要的 CORBA 声明

```
typedef sequence < long > courses;
interface black_board {
    // …
    void suggestionsForGeneral(in courses General);
    void suggestionsForMajor(in courses Major);
```

```
    void suggestionsForElectives(in courses Electives);
    void suggestionsForMinor(in courses Minor);
    courses currentDegreePlan();
    courses suggestedSchedule();
    // …
};
```

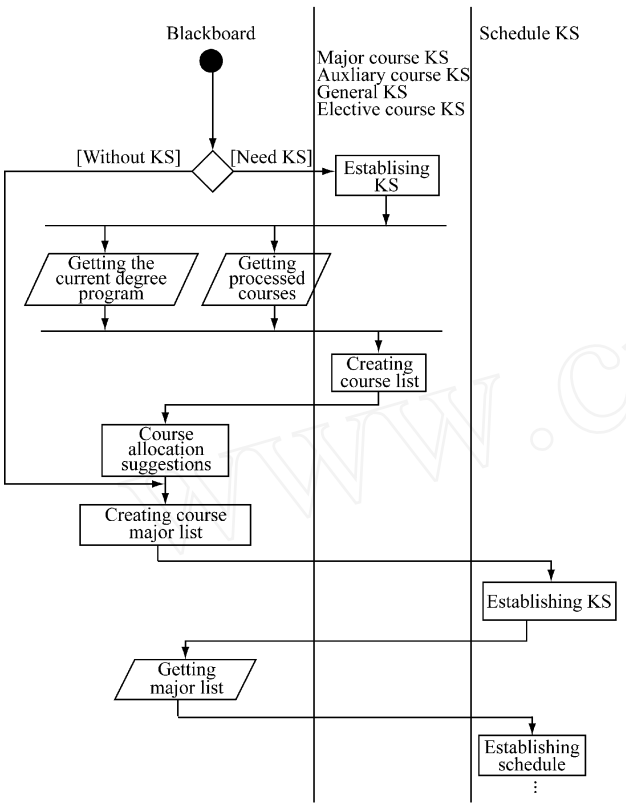

图 5　黑板和知识库的同步

Fig. 5　Synchronization of blackboard and knowledge source

black_board **接口类的实现**:

```
class black_board : virtual public POA_black_board {
protected:
    // …
    set<long> SuuggestionForGeneral;
    set<long> SuggestionForMajor;
    set<long> SuggestionForElectives;
    set<long> SuggestionForMinor;
    courses Schedule;
    courses DegreePlan;
public:
    blackboard(void);
    ~blackboard(void);
    void suggestionsForGeneral(const courses &X);
    void suggestionsForMajor(const courses &X);
    void suggestionsForElectives(const courses &X);
    void suggestionsForMinor(const courses &X);
    courses * currentDegreePlan(void);
    courses * suggestedSchedule(void);
    // …
};
```

连接到基于 CORBA **黑板的知识库实现中的部分代码如下**:

```
int main(int argc, char *argv[])
{
    CORBA::ORB_var Orb = CORBA::ORB_init(argc, argv, "mico-local-orb");
    CORBA::Object_var Obj = Orb->bind("IDL:black_board:1.0", "inet:porthos:12458");
    courses Courses;
    // …
    // …
    black_board_var Blackboard = black_board::_narrow(Obj);
    int Pid;
    // …
    // …
    cout << "created the knowledge source" << endl;
    Courses.length(2);
    Courses[0] = 255551;
    Courses[1] = 253212;
    string FileName;
    strstream Buffer;
    Pid = pvm_mytid();
    Buffer << "Result." << Pid << ends;
    Buffer >> FileName;
    ofstream Fout(FileName.data());
    BlackBoard->suggestionsForMajor(Courses);
    Fout.close();
    Pvm_exit();
    return (0);
}
```

即使是完美的课程表,学生也经常会遇到麻烦。比如,在登记课程的过程中,经常会碰到课堂的座位竞争问题,而且在某些时间点,重要的课程会停止,这就是先来先服务处理方法带来的问题。在登记课程的时候,有成千上万的学生要选修数量有限的课程,学生希望能选到直接应用到自己学位的课程。

理想情况下,这些课程能在适当的时候进行,并且想修的学生都能参加。而且,学生都希望课程量限制在某种范围内,这样可以留点时间参加其他课外活动。这里存在的问题是,当学生准备去上一门已经安排好的课时,却由于某些原因不一定能上,所以经常用代替课或补课来取代。这些额外课加重了学生的花费,也延长了学习的持续时间。从学生的利益角度考虑,这些花费和耽误的时间都是负面的结果。当然,如果这些附加课是与学生的学术之外的兴趣、习惯或目标有关,那么还是可以勉强接受的,而且,每个学位之下都有一些选修课和可选课。但学生都是希望选到合适的课,使得自己在预算之内可以提前或按时毕业,并尽可能具有最大灵活性。基于黑板技术的实时课程建议系统就很好地解决了这一问题,满足了学生个性化的选课要求。而且这种技术方法可以在许多类似问题的环境中加以运用。具有明显的应用前景。

5 结 论

黑板模型支持并行性,黑板是问题解决模型。根据知识领域的不同对问题进行划分,每个划分指派给一个知识库或问题解决者。知识库和问题解决者是典型的自包含,并且很少需要与其他知识库的交互。必要的通信通过黑板进行,因此,知识库和问题解决者服务于模块化程序中的处理过程。这些模块可以分别对待,并且它们可以并行执行而没有复杂的同步需求。黑板可以使用 CORBA 对象实现。在使用 CORBA 对象时,知识库可以分布于 Intranet 和 Internet 上。在一个 PVM 型的环境中,黑板为任务充当一种共享分布史存储器。黑板模型易于支持多指令流多数据流和单指令流多数据流模型。黑板的概念激发了设计者顺着知识分界线分解程序需要解决的问题。这使得程序有一个知识专家的 WBS。黑板包含了问题域和解空间的软件模型,这个模型帮助设计者和开发者发现要实现的程序的必要并行性。除了分布式编程的经典的客户/服务器模型,黑板模型也是分布式和并行编程可用的强有力的模型之一。黑板模型中的知识库和问题解决者经常当成 Agent 实现。

参考文献:

[1] 陈国良. 并行计算——结构·算法·编程[M]. 北京:高等教育出版社, 1999.

[2] 陈国良. 并行算法的设计与分析(修订版)[M]. 北京:高等教育出版社, 2002.

[3] 陈国良. 并行算法实践[M]. 北京:高等教育出版社, 2004.

[4] Barry Wilkinson, Michael Allen. 并行程序设计[M]. 陆鑫达译. 北京:机械工业出版社, 2002.

[5] Jack dongarra, Ian Foster. 并行计算综论[M]. 莫则尧, 陈军译. 北京:电子工业出版社, 2005.

[6] NIIHP. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architecture [J]. The AI Magzine, 1986, **17**(2): 38-53.

[7] Gilmore J F, Roth S P, Tynor S D. A blackboard system for distributed problem solving [M]. San Diego, CA: Academic Press, 1989.

[8] McManus J W. Design and analysis techniques for concurrent blackboard systems [J]. IEEE Trans On System, Man and Cybernetics Part A: Systems and Humans, 1996, **26**(6): 669-680.

[9] 王树林, 梅多伦. 黑板结构研究[J]. 计算机研究与发展, 1990, **27**(10): 1-5.

[10] 高维君, 苏士权. 一个基于面向对象方法的并行黑板系统的设计[J]. 小型微型计算机系统, 1994, **15**(1): 8-14.

[11] 曹文君, 应效宇, 黄 强,等. 一种面向对象黑板系统及其在大容量知识处理中的应用[J]. 复旦大学学报(自然科学版), 1994, **33**(1):67-74.

[12] 黄 波, 倪重匡. 一种基于黑板模型的智能决策系统生成器的结构设计[J]. 计算机研究与发展, 1997, **34**(5): 382-386.

[13] 陈洪波, 刘 群. 支持混合知识表示的面向对象黑板模型[J]. 哈尔滨工程大学学报, 1998, **19**(1): 47-54.

[14] 李旗号. 面向对象的黑板集成模型在 ICU 医疗辅助诊断中的应用[J]. 合肥工业大学学报(自然科学版), 2000, **23**(4): 567-571.

[15] 于存贵, 李自勇, 马志文. 基于黑板模型的多属性决策模式[J]. 南京理工大学学报, 2000, **24**(4): 334-337.

[16] 辛明军, 李伟华. 基于黑板决策模型的工程软件集成开发平台研究[J]. 西北工业大学学报, 2001, **19**(2): 288-291.

[17] 王 斌, 张尧学, 陈松乔. 一种基于黑板模型的多 Agent 系统通信方法[J]. 小型微型计算机系统, 2002, **23**(11): 1355-1358.

[18] 周光明. 分布式黑板模型的设计[J]. 计算机工程与应用, 2004, (3): 110-112.

[19] 王 斌, 张尧学, 陈松乔. 分布式环境下代理协同的主动黑板结构设计模式[J]. 计算机工程, 2004, **30**(9): 147-149.

[20] (德) Frank Buschmann, Regine Meunier, Hans Rohnert, 等. 面向模式的软件体系结构(卷 1):模式系统[M]. 贲可荣, 郭福亮, 赵 皑, 等译. 北京:机械工业出版社, 2004.

[21] 刘君强. 黑板系统的关键技术与开发[J]. 计算机时代, 2003, (8):1-2.

[22] Stephan C. Werges, David Naylor. CORBA infrastructure for distributed learning and meta-learning [J]. Knowledge-based System, 2002, **15**(1-2):139-144.

[23] 郑阿奇. Visual C++实用教程(第2版)[M]. 北京:电子工业出版社, 2003.