一个支持跨域资源同步分配的虚拟作业模型

丁肇辉[1]，魏晓辉[1]，马达[1]，骆远[1]，Wilfred Li[2]，Peter Arzberger[2]
（1．吉林大学 计算机科学与技术学院，长春 130012
2．美国加州大学 国家生物医学计算资源，圣地亚哥，加州，美国，92093)

摘要：当前，越来越多的科学家开始使用网格技术解决不同科研领域的问题，然而在网格环境中运行并行应用程序仍然面临着重大挑战。尽管 MPICH-G2 能够利用多个域的资源运行 MPI 并行应用，但是，由于 MPICH-G2 不能同步地分配跨域资源，在并行作业的子作业等待同步启动时会导致资源浪费，在多个并行作业竞争资源时，甚至可能发生资源死锁情况。本文提出了一个能够在跨域环境中为网格并行应用程序完成同步资源分配的虚拟作业模型 VJM，VJM 能够避免由于多个并行作业竞争而产生的死锁，还可以利用小作业装填技术缓解因同步资源分配而产生的浪费。而且，VJM 并不依赖于资源预约机制，因此它可以通过标准的 GRAM 协议与绝大多数现有的局部调度器协作。最后，在元调度器 CSF4 中实现了该模型，并通过生物信息并行应用程序 mpiBLAST-G2，验证了模型的合理性。

关键字：网格计算，虚拟作业，跨域，CSF4，同步资源分配
中图分类号：TP391　　文献标识码：A

A Virtual Job Model to Support Cross-Domain Synchronized Resource Allocation

DING Zhaohui[1], WEI Xiaohui[1], MA Da[1], LUO Yuan[1], Wilfred LI[2], Peter ARZBERGER[2]
（1．College of Computer Science and Technology, Jilin University，Changchun 130012
2．University of California at San Diego / NBCR，San Diego, CA, USA, 92093）

Abstract：Although more and more scientists start to take advantages of grid technologies to facilitate their researches, running parallel jobs crossing domains in a grid environment is still a challenge. Even MPICH-G2 is able to run MPI applications on across domain resources, however, the resource allocations are not synchronized which will cause dead lock and other serious problems. In this paper, we introduced a virtual job model (VJM) which achieves synchronized cross-domain resource allocation for parallel grid applications. VJM is able to prevent the resource allocation deadlock caused by multiple parallel jobs competing resource, and alleviate the resource waste by backfilling small jobs. VJM can work with almost all kinds of local schedulers via standard Grid Resource Allocation and Management (GRAM) protocol as it does not depend on resource reservation. We have implemented VJM in meta-scheduler CSF4 and validate the rationality of VJM by mpiBLAST-g2, a parallel bioinformatics application.

Keywords: Grid Computing, Virtual Job, Cross-domain, CSF4, Synchronized Resource Allocation
收稿日期：2007-10-10.

作者简介：丁肇辉（1979～），男，汉族，博士，从事网格计算与元调度研究，Email：zhaohui.ding@email.jlu.edu.cn. 联系人：魏晓辉（1972～），男，汉族，博士，教授，从事分布式与网格系统研究，Email：weixh@jlu.edu.cn.

基金项目：国家自然科学基金（项目号 60703024），美国远距离医学与高级技术研究中心基金（项目号 W81XWH-07-2-0014），吉林省自然科学基金的（项目号：20060532）；吉林省杰出青年基金项目（项目号 20070122）。

1. 引言

网格技术的目的是在分布、异构、自治的网络环境中实现资源共享与协同工作。当前，网格技术已经在生物信息、高性能物理、地质等科研领域得到应用。但是，在运行并行应用程序时，网格使用者更多的仍然是依靠局部调度器管理，使用单个域的资源完成，因为协同跨域资源执行并行作业仍面临着诸多挑战。

在传统的单一集群环境中，编写并行应用程序通常要遵循某种消息传递机制，如PVM[1]、MPI[2]。美国Argonne国家实验室依照MPI-2 标准实现了MPICH，并在MPICH的基础上开发了适用于网格环境的版本MPICH-G2[3]。MPICH-G2 广泛地采用了Globus技术，包括使用网格资源分配与管理（GRAM）协议和动态更新的在线协同分配（DUROC）协议作为并行作业启动和相互通信的接口。但是，在网格环境中和单一集群中运行作业是有区别的，作业被派送到网格节点上时不是被立即执行，而是由本地调度器依照本地策略进行调度。由于MPICH-G2 并没有一个跨域资源同步分配的机制，在并行作业的子作业等待同步启动时会导致资源浪费，在多个并行作业竞争资源时，甚至可能发生资源死锁。此外，MPICH-G2 在资源可用性检查方面也不尽完善，单个资源的失效会使整个作业失败。

显然，如果没有一个高层的协调者，这些问题将很难解决。本文提出了一个能够管理网格并行作业和异构资源的虚拟作业模型（VJM），该模型处于元调度层，能够为网格并行作业同步分配跨域的异构资源、避免资源死锁；VJM并不依赖于资源预约机制，因此它可以通过标准的GRAM协议与绝大多数现有的局部调度器协作，如不支持资源预约的OpenPBS[4]、SGE[5]；VJM的小作业装填功能能够缓解资源浪费情况；此外，由于VJM并没有对MPICH-G2的函数库做修改，因此它可以很容易地部署在现有的网格环境中。

在VJM的实现方面，我们主要考虑的是尽量利用现有的软件和技术，避免重复开发。CSF4[6]是由本课题组之前开发的元调度器，目前已经作为执行管理组件加入到Globus Toolkit 4 中。由于CSF4 具有较好的可扩展性，因此我们将VJM实现在CSF4 中，并对CSF4 的作业管理和资源管理机制进行了扩展，使其能够适应并行作业的需要。

文章组织结构如下，第二节介绍以往的相关工作，并与本文提出的 VJM 做了比较；第三节详细地分析了目前运行网格并行应用所面临的问题，从体系结构、实现方法等方面阐述 VJM 如何解决这些问题；第四节通过实验数据说明了 VJM 的效率和合理性；最后是文章结论以及未来工作展望。

2. 相关工作

目前，已有多种技术在网格并行应用程序方面取得了一定成果。

MPICH-G2 将网格技术与MPI标准结合起来，使在网格环境中运行并行MPI作业成为可能。MPICH-GX[7]针对MPICH-G2 作业执行期的问题做了三个改进，1. 利用NAT服务和用户级代理支持了局域网内的虚拟IP；2. 提供的检查点恢复系统提高了作业启动后的容错能力；3. 优化了系统协同通信。但是，在资源分配方面，MPICH-G2 使用静态的资源配置信息和Round-robin的资源匹配方法来进行作业与资源的配对，这种做法可能产生包括资源分配死锁，资源浪费，资源无效等问题。本文提出利用虚拟作业为真实作业提前寻找和占据资源的方法，为并行作业同步地分配资源，能够解决上述问题。

资源提前预约技术是解决跨域资源同步分配的方法之一。文献[8]提出将异构资源同步分

配过程在于分为二步，即资源申请与资源分配，GARA[9]是在文献[8]的基础上采用预约实现的网格资源分配协议。Silver[10]是由美国ClusterResource公司开发的商用资源管理系统，该系统中的Moab是一个基于资源提前预约的网格调度器。Silver还针对并行作业提出了同步资源预约的概念。但是，采用“预约”技术来解决并行作业的资源同步问题有一定的局限性，第一，并非所有集群调度器都支持资源“预约”，例如在科研组织和开源社区很流行的OpenPBS和SGE；第二，不同的集群调度器对“预约”的定义并没有统一的标准，例如，较严格资源预约要求用户在申请预约时提供明确的预约起止时间，因此很难在资源管理层定义出通用的支持“预约”的协议。本文提出的VJM利用虚拟作业动态地申请资源，不需要提前指定使用资源的时间段，不依赖于资源预约，因此，更为灵活，适用范围也更为广泛。

从现有的元调度器或网格资源代理软件对并行作业的支持来看，多数软件并没有针对并行作业做特定的优化。有些仍然停留在分布式系统的层面，即仅支持利用单个域的资源完成并行作业，如Nimrod-G[11]，Condor-G[12]，或者不支持并行作业如Gridway[13]。本文将支持跨域资源同步分配的VJM实现在CSF4 元调度器中，这为在网格环境中运行并行作业提供了一个完整可靠的解决方案。

3. 虚拟作业模型

为了使并行作业能够在网格环境中运行，有三个关键问题需要解决。第一，数据的可用性，即无论应用程序在哪里执行，都必须能够访问到它需要的输入/输出文件；第二，跨域的通信，即隶属于同一并行程序的多个子进程能够在跨域的环境中相互通信；最后是资源分配，即能够协调那些跨域的、异构的资源，同步地为并行作业分配资源。数据网格技术，如Gfarm[14]，提供了一个全局共享的文件系统；MPICH-G2 是一个适用于网格环境的MPI实现，它使MPI作业能够跨越多个网格节点运行；但是，MPICH-G2 的资源分配机制不能满足网格并行作业的需要。本节分析了MPICH-G2 现存的问题，提出了虚拟作业模型（VJM），并从体系结构、实现方法等方面阐述VJM如何解决这些问题。

3.1 网格并行作业存在的问题

在使用 MPICH-G2 运行并行作业时，用户首先使用客户端工具 *mpirun* 处理用户请求，基于预先配置的资源描述文件或 MDS，*mpirun* 采用 round-robin 的资源匹配方法，产生一份该作业的 RSL 描述，然后，根据这份 RSL，*mpirun* 使用 *globusrun* 工具将作业提交到指定的集群上，该集群上调度器再依照本地策略对作业进行调度。按照这种做法，可能产生如下的问题。

1）由于没有资源可用性检查机制，一旦某个资源失效（如服务器没有启动），将导致整个作业失败。

2）当并行作业的各个子作业被派送到不同网格节点上的集群后，和分布式系统中的并行作业不同的是，作业并不被立即执行，而是由本地调度器依照本地调度策略调度。由于没有一个同步资源分配的机制，那些较早启动的子作业必须等待其他仍在本地调度器中排队的作业，在这种情况下，会导致资源浪费。

3）当同时提交多个并行作业时，在集群资源并不充足的情况下，不同的集群本地策略可能导致资源分配死锁。图 1 说明了一个最简单的资源分配死锁的例子。在本例中，管理员为 MPICH-G2 配置了 2 个可用集群 C1 和 C2，每个集群都只有 1 台可用机器。现有 2 个并行作业 a 和 b，它们都需要 2 台机器来完成，我们称 a 和 b 的子作业分别为为 a1、a2 和 b1、b2。按照 *mpirun* 的匹配方法，a1 和 b1 会被派送至 C1，而 a2

和 b2 被派送至 C2。尽管利用现有的可用资源能够完成这 2 个作业，但是有可能发生这种情况，a1 优先于 b1 抢占到资源并等待 a2 也抢占到资源来启动作业 a，但 a2 又在等待 b2 结束以获得资源，从而形成死锁。产生这种死锁的原因有 2 个，第一是两个作业的并发提交，而发生提交顺序为 a1，b1，b2，a2 的情况；第二是两个集群的调度策略不同，以优先级为例，a 在 C1 的优先级比 b 高，但在 C2 的优先级比 b 低。一旦发生死锁，不但两个作业都无法完成，资源还会被持续占据，直到本地调度器自动终止它们。

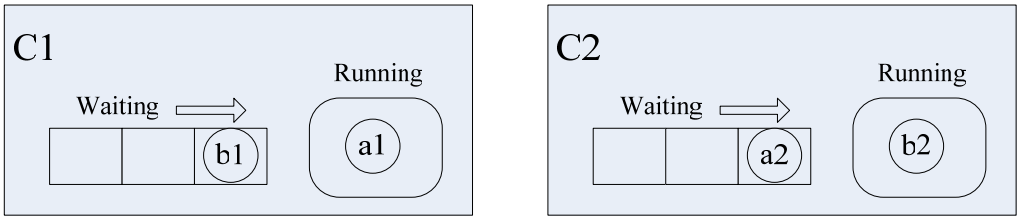

图 1 多个并行作业产生资源分配死锁的范例

产生上述问题的主要原因是，MPICH-G2 并没有一个资源管理、同步分配机制，而且作业被派送到网格节点上的集群时，并不是被立即执行。为此，我们设计了虚拟作业模型 VJM。

3.2 VJM 的设计

虚拟作业（VJ）是一个为 MPICH-G2 解决跨域资源同步分配的元调度模块。在 VJM 中，如图 2 所示，作业在提交期并不立刻确定即将使用的资源，而是根据作业的资源要求，派送一定数目的虚拟作业（VJ），VJ 的主要功能是，抢占资源并且将资源状况反馈给虚拟作业控制中心（VJC），VJC 根据虚拟作业返回的资源信息为并行作业同步分配资源。

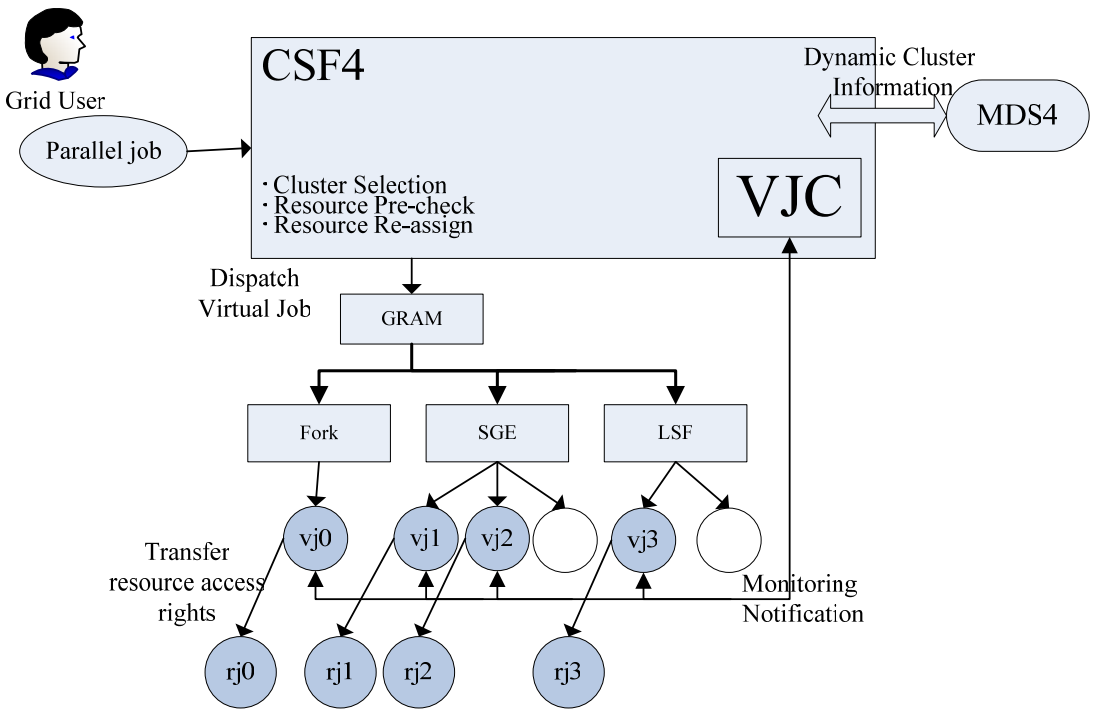

图 2 虚拟作业模型体系结构

在 VJC 的控制下，不仅能避免发生资源分配死锁，还能够实现小作业的装填来缓解资源浪费。与 CSF4 的整合是为了利用元调度器的资源检查和资源选择以更快更准确的定位到有效资源。考虑到网格环境的分布且异构的特点，VJM 并不依赖于那些并不被所有局部调度

器所支持的功能，如资源预约，同时，使用 VJM 也不需要网格节点改安装新的服务或改变环境的配置。下面我们将按照一个并行作业的生命周期来描述 VJM 的详细设计。

3.2.1 并行作业生命周期

首先，我们将并行作业提交工具 *mpirun* 替换为 CSF4 的作业提交工具 *csf-job-create*，并加入了针对并行作业的处理。我们并不像 *mpirun* 那样立即为并行作业决定所有即将使用的资源，而是利用 CSF4 的资源可用性预检测机制和集群信息获取机制，先决定一定数目的候选资源（通常大于所需资源数）。预检测机制是在提交作业之前，先发送一个认证请求，以检测目标集群是否已启动。基于预检测的结果，CSF4 就能够选出那些真正可用的集群，避免将不可用资源加入候选资源。而集群信息获取机制能够获得如 CPU 个数，集群队列长度等信息，CSF4 会根据这些信息向集群申请适当的资源数，并拒绝那些资源需求超出所有可用资源数的作业。尽管 CSF4 能够决定较优的候选资源，但是，元调度器并不是网格资源的真正拥有者，它无法直接地为作业分配资源，因此，我们引入了虚拟作业来完成资源协同分配。

我们设计了虚拟作业控制中心（VJC），首先，VJC 向所有候选资源分别发送一个 VJ。局部调度器会象调度正常作业一样调度 VJ，当 VJ 获得资源使用权并成功启动时，VJ 会通知 VJC，把刚获得资源的信息（包括主机地址和端口）向 VJC 注册，并等待 VJC 的下一步指示。如图 3 所示，VJC 根据这些资源注册信息，将资源编号，维护一个虚资源池，一旦虚资源池的资源能够满足并行作业的需要，VJC 就把真正的并行作业的各个子作业派送给已经获得资源使用权的 VJ，通知 VJ 启动并行作业。由于 VJ 保证了资源可用性，所以只需 VJ 将资源使用权转交给真实作业，就可以完成并行作业的同步启动。

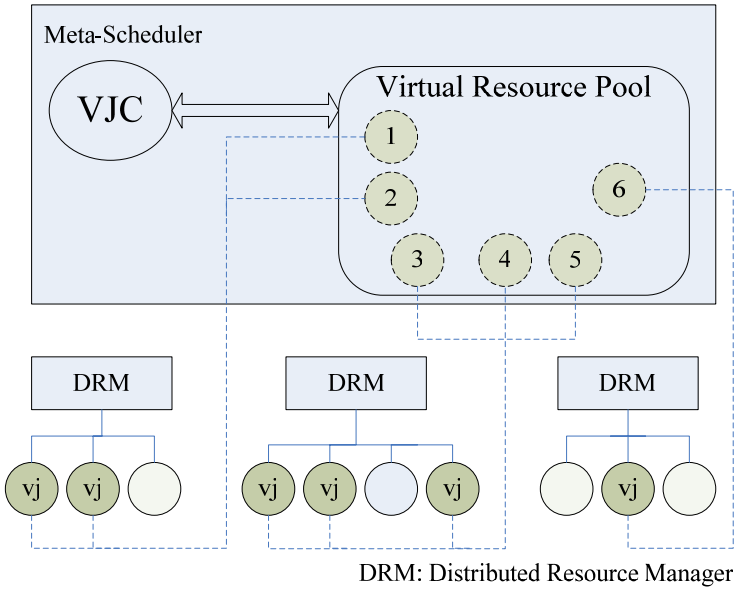

图 3 虚拟作业控制中心与虚资源池

可见，VJ 和普通作业的提交过程没有任何区别，不依赖于资源预约，所以我们可以采用标准的 GRAM 协议进行作业提交，也就能够适应绝大多数现有的局部调度器。VJM 也并没有对 MPICH-G2 的函数库做修改，因此它可以很容易地部署在现有的网格环境中。而且，由于 VJC 对资源和作业的统一管理，能够完全避免资源分配死锁情况的发生，还可以实现小作业装填以缓解资源浪费。

3.2.2 死锁预防与小作业装填

如 3.1 节中所分析的，并行作业的并发提交和各集群局部调度的随机性可能导致资源分配死锁。我们先给出死锁的产生条件与定义。

对于两个并行作业，Ja 和 Jb，它们均有子作业被派送到了集群 C 上，当且仅当 Ja 的子作业在集群 C 上占据了资源，而 Jb 的子作业无法获得资源，除非 Ja 的子作业释放资源，我们称之为“Ja > Jb on C”或“Jb < Ja on C”，那么死锁的产生条件即，

$$\exists (Ja,\ Jb,\ Cm,\ Cn) \rightarrow (Ja < Jb \text{ on } Cm,\ Ja > Jb \text{ on } Cn) \qquad (1)$$

存在一对集群 Cm 和 Cn，如果“Ja < Jb on Cm”且“Ja > Jb on Cn”，那么我们认为在集群 Cm 和 Cn 上，并行作业 Ja 和 Jb 发生了资源分配死锁。为了解决死锁情况，我们首选考虑采用死锁检测与打破的方法。死锁检测可以根据条件（1）实现，一旦发现死锁，可以根据某种算法打破死锁，如根据作业优先级的高低，中止造成死锁的作业中优先级较低的作业，并为该作业重新分配资源。实现一个死锁检测算法，在最坏的情况下，需要比较所有集群上的隶属于不同并行作业的子作业之间的依赖关系。显然，算法的时间复杂度相当大。

另一种解决死锁的方法是死锁预防。由死锁产生条件（1）可以看出，在资源分配期，并行作业的子作业之间存在的依赖关系是产生死锁的关键原因。我们的做法是在虚拟作业模型中，利用 VJ 避免这种依赖关系，从而消灭死锁产生条件。如前所述，VJ 的作用是为真实作业抢占资源，但是，在发送 VJ 时，并不为每个 VJ 对应一个特定的 RJ。一个 VJ 只是代表一个资源，当 VJ 未被启动时，它代表一个可用性未知的资源；当 VJ 处于运行状态时，它代表一个可用资源。VJC 将并行作业当作一个整体来调度，当可用资源池内的资源数量满足了并行作业的需要，VJC 再将并行作业的各个子作业一并提交出去。在资源分配期，由于各 VJ 之间并没有任何依赖关系，死锁的产生条件也就不存在了。

依照上述的死锁预防方法，能够有效地提前阻止了死锁形成条件。此外，这种做法还有一个好处，就是可以方便地实现小作业装填算法，以缓解在多个作业等待同步启动时产生的资源浪费问题。

在分布式系统中，小作业装填技术已经得到广泛地应用，它能够有效地提高资源利用率。在虚拟作业模型中，VJC 可用通过已获得资源使用权的 VJ，完成小作业装填。但是，作业装填算法需要调整以适应网格环境。在分布式系统中，局部调度器拥有整个集群的完全控制权限，因此，只要满足装填条件，任何作业都可以被装填到空闲资源上。而在虚拟作业模型中，作业装填是由虚拟作业完成的，而对于集群来说，虚拟作业只是一个普通的作业，拥有的普通用户权限。在进行小作业装填时，VJ 只能以自己的身份执行被装填作业，因而可能产生访问权限的问题。我们在对网格应用程序的调研发现，权限问题通常是由应用程序访问文件导致的。因此，只要能解决文件访问权限，就可用应付绝大多数应用。

在数据网格技术（如Gfarm）中，文件的访问控制采用认证中心/证书机制代替了传统的用户名/密码机制，只要用户拥有授权证书，就能够访问相应的文件。在之前的工作中，我们已经成功地集成了CSF4 与Gfarm[15]，在此基础上，我们只需要在装填作业时，将被装填作业所属用户的证书委托 (Delegate)给VJ，VJ负责证书在执行机上的设置与清除，被装填作业就可用成功地访问所需的文件，作业装填功能也就基本得以实现。

4. 实验

VJM 为同步资源分配调用的一些额外的进程，包括启动虚拟作业和 VJC，以及这两者的通讯，都导致了额外的开销。然而，资源同步分配、资源死锁避免和小作业装填技术，能够提高并行作业的吞吐量和资源的利用率。

我们通过如下实验检验虚拟作业模型的功能。当所有已提交作业的请求资源的总数超过了可用资源的总和，就有可能发生资源的死锁。我们搭建了两个集群(A 和 B)，每个集群都有 3 个节点，两个用户(user1 和 user2)。A 集群中，user1 的优先级要高于 user2 的优先级；B 集群中，user2 的优先级要高于 user1 的优先级。在实验中， user1 和 user2 同时提交一个需要 6 个计算节点的并行作业—作业 α 和作业 β。当用 MPICH-2 提交时，两个作业均不能被执行，并且资源死锁使这两个集群都处于不可用的状态，如图 4-(1)所示。当用 VJM 提交时，作业 β 得到了资源 C2 而未得到资源 C1 而因此被终止，并由 VJC 重新分配。最后，作业 α 得到了所需的 6 个资源而使其子作业得以执行，如图 4-(2)所示。为了检验小作业装填技术，不同的用户在提交并行作业的同时随机插入一些持续 10 秒的小作业。这些被填充的小作业都得到了执行。

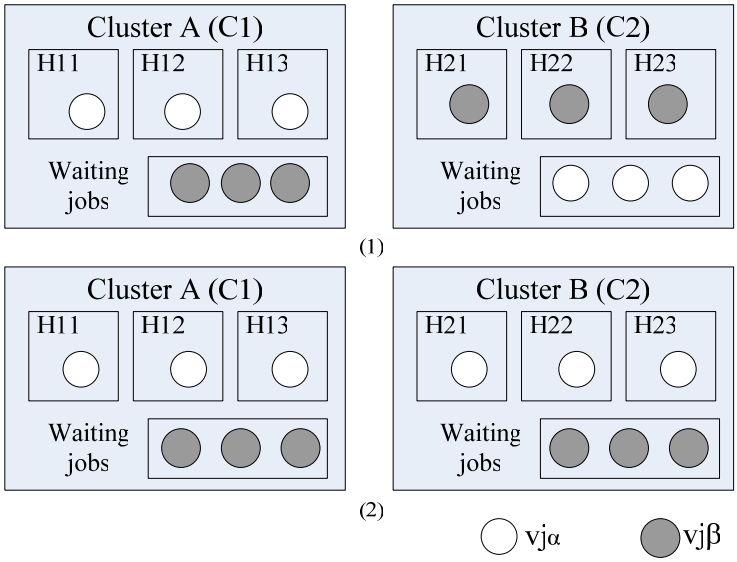

图 4. VJM 避免资源分配死锁

为了检验虚拟作业模型的性能，我们搭建了如下的实验环境，3 个集群分布于 2 个独立的域，这 3 个集群分别拥有 3、3、4 个 CPU。我们以 mpiBLAST-g2，一个并行的生物信息应用程序，作为测试程序进行了以下实验。我们用了 3 个测试用例，即网格负载状态为 Case 1：空闲--所有节点都空闲，Case 2：中等忙碌—请求节点数少于空闲节点数，Case 3：忙碌—请求的节点数大于空闲的节点数。Case 1 状态下，可用的节点总数为 10，Case 2 状态下，可用的节点总数为 8，Case 3 状态下，可用的节点总数为 6。

我们产生一组测试样本，执行程序均为 mpiBLAST-g2，它们所需的执行机数量为 4，6，8。我们将这组样本分别逐个地使用 MPICH-G2 和 VJM 进行提交。获得资源协同分配的平均时间如表 1 所示。可以看出，由于 MPICH-G2 根据静态的资源配置文件使用 Round-robin 的匹配方法，所以作业会被平均地分配到 3 个集群上。而 VJM 利用虚拟作业寻找和占据资源，再由 VJC 统一派送作业，因此在分配作业时能够考虑到集群计算能力、闲忙程度，以达到较好的负载平衡。当所有节点都空闲时(Case 1)，因为存在额外开销，VJM 提交作业时所用的资

源分配时间比 mpirun 长；当系统处于比较忙碌或很忙碌时(Case 2, Case 3)，VJM 提交作业所用的资源分配时间明显比 mpirun 短的多，这是因为 mpirun 将子作业平均分配到不同集群而不考虑节点的可用性。当系统处于忙碌时，两者的时间几乎相等，这是因为 VJC 必须等待一些优先级高的作业先完成。但是，由于我们的 VJC 能够装填一些小作业，资源的实际利用率要远高于 MPICH-G2。

Type	**Resource Co-allocation Time (Second)**								
Case (Resource Status)	**Case 1 (Idle)**			**Case 2 (Moderate)**			**Case 3 (busy)**		
Job Required CPU Number	**4**	**6**	**8**	**4**	**6**	**8**	**4**	**6**	**8**
MPICH-G2	2.4	2.4	2.5	2.5	28.3	35.1	2.5	32.7	37.7
VJM	4.9	6.2	7.6	5.0	6.9	7.3	5.3	6.8	39.4

表 1. VJM 与 MPICH-G2 资源同步分配时间比较

5. 总结与展望

本文分析了网格环境中运行并行作业存在的问题，介绍了相关工作研究现状，提出了一个支持跨域资源同步分配的虚拟作业模型 VJM。VJM 处于元调度层，它能够管理网格并行作业和异构资源，利用虚拟作业，VJM 能够为网格并行作业同步分配跨域的异构资源、避免资源死锁，利用小作业装填技术缓解资源浪费。而且，VJM 并不依赖于资源预约机制，因此它可以通过标准的 GRAM 协议与绝大多数现有的局部调度器协作。最后，我们通过生物信息并行应用程序 mpiBLAST-G2，检验了模型的性能，验证了模型的合理性。

在今后的研究中，我们将深入研究各个科学领域的并行应用程序，总结出并行应用程序通用特征，并根据这些特征，开发基于 VJM 的调度算法，提高应用程序性能和资源利用率。而且，利用 CSF4 可扩展的调度框架机制，可以很方便地将网格并行应用调度算法以插件的形式在 CSF4 中实现。

6. 致谢

本文工作得到以下基金支持，中国国家自然科学基金，项目号 60703024；美国远距离医学与高级技术研究中心基金，项目号 W81XWH-07-2-0014；吉林省自然科学基金，项目号 20060532；吉林省杰出青年基金项目，项目号 20070122。

参考文献

[1] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Manchek, V. Sunderam, PVM:Parallel Virtual Machine—A User's Guide and Tutorial for Network Parallel Computing[J], MIT Press, Cambridge, MA, 1994.

[2] Message Passing Interface Forum, MPI:A message-passing interface standard[EB], Internat. J. Supercomput. Appl. 8(3/4)(1994) 165–414.

[3] Nicholas T. Karonis, Brian Toonen, Ian Foster, "A MPICH-G2: A Grid-enabled implementation of the Message Passing Interface" [J], Journal of Parallel and Distributed Computing.

[4] James, P. J, Portable Batch System: Exterernal Reference Specification Altair PBS Pro 5.3 [EB]. http://www.mta.ca/torch/pdf/pbspro54/pbsproers.pdf, March 2003.

[5] Sun Microsystems, Inc. Sun Grid Engine, http://gridengine.sunsource.net/

[6] Wei X, Ding Z, Yuan S, Hou C, LI H, "CSF4: A WSRF Compliant Meta-Scheduler",

International Conference 06' on Grid Computing and Applications, Las Vegas, USA, June 26-29, 2006.

[7] K. Park, H. Lee, Y. Lee, O. Kwon, H. Park, S. Kan, “An Effective Collective Communication method in Grids Using Two Level Latency-Optimal Tree”[M], Computational Science - ICCS 2003, LNCS, Volume 2660/2003.

[8] K. Czajkowski, I. Foster, and C. Kesselman. "Resource Co-Allocation in Computational Grids", Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8), pp. 219-228, 1999.

[9] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy. “A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation” Intl Workshop on Quality of Service, 1999.

[10] SILVER Design Specification, Jan 16, 2004. http://www.supercluster.org/projects/silver/.

[11] Rajkumar Buyya, David Abramson, and Jonathan Giddy, Nimrod-G Resource Broker for Service-Oriented Grid Computing[M], IEEE Distributed Systems Online, in Volume 2 Number 7, November 2001

[12] J. Frey, T. Tannenbaum, M. Livny, I. Foster and S. Tuecke, “Condor-G: A Computation Management Agent for Multi-Institutional Grids” [J], Journal of Cluster Computing, vol. 5, Number 3, July, 2002.

[13] Distributed Architecture Group from Universidad Complutense. Gridway, http://www.gridway.org/, 2006.

[14] Osamu Tatebe, Youhei Morita, Satoshi Matsuoka et al. Grid Datafarm Architecture for Petascale Data Intensive Computing [C]. Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pp.102-110, 2002.

[15] Wei X., Ding Z, Li WW. "GDIA: A Scalable Grid Infrastructure for Data Intensive Applications"[C], 2006 International Conference on Hybrid Information Technology, IEEE CS Press, Cheju Island, Korea,9-11 Nov, 2006.