基于随机 Petri 网的高性能计算系统作业调度及 InfiniBand 网络互连的性能分析

李智佳[1,2] 胡 翔[1,2] 焦 莉[1] 王伟锋[1,2]

(中国科学院软件研究所 北京 100190)[1] (中国科学院大学计算机与控制学院 北京 100049)[2]

摘 要 基于模型的分析技术在系统研究和设计中发挥着重要作用,它具有简单灵活、可扩展性强、高效等优点,其中随机 Petri 网在性能评价方面得到了广泛的应用。使用随机 Petri 网为高性能计算机的作业调度系统进行抽象和建模,并将其与 InfiniBand 网络互连结构相结合来整体分析用户作业的延迟等性能指标。实验表明,该方法是可行的,且具有相对较高的精度。

关键词 随机 Petri 网,高性能计算,InfiniBand,作业调度,模型,性能评价

中图法分类号 TP302.7 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.007

Performance Evaluation of Job Scheduling and InfiniBand Network Interconnection in High Performance Computing System Based on Stochastic Petri Nets

LI Zhi-jia[1,2] HU Xiang[1,2] JIAO Li[1] WANG Wei-feng[1,2]

(Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]

(School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China)[2]

Abstract Model-based analysis technology plays an important role in the system research and design, because it has many advantages, such as simplicity, flexibility, scalability, and efficiency. Especially, stochastic Petri nets are widely used in performance evaluation. We proposed a method of synthetical performance evaluation of job scheduling and InfiniBand network interconnection based on stochastic Petri nets. The experiment results demonstrate that our method is feasible and comparatively accurate.

Keywords SPN, HPC, InfiniBand, Job scheduling, Model, Performance evaluation

随着人类面对的计算问题的规模和难度逐渐增大,高性能计算作为一种重要手段被广泛地应用到了科学研究和工程技术的各个领域,基于高性能计算理论的高性能计算系统也得到了飞速的发展。如何充分发挥高性能计算机的性能成为一个亟待解决的重点和难点问题。性能评价可为解决这一问题提供参考,其在系统的设计、实现、采购、部署、优化等生命周期过程中均能发挥重要的作用,有助于揭示系统性能瓶颈,从而指导系统性能优化。

作业调度系统作为纽带,向上连接着众多的用户,向下管理着庞大设备,是高性能计算系统的核心子系统。近年来,众多成熟的作业调度系统得到应用,如 LSF[11]、PBS、LoadLever、Legion、Maui 等,其中以 Platform 公司的商业软件 LSF 最为出名。在研究层面,许多文献关注于作业调度算法[12,17]、作业调度系统的设计[13]以及作业调度系统的性能评价[14-16]等方面的研究。网络互连的性能直接关系到高性能计算系统的整体性能。InfiniBand(以下简称 IB)是一种高速互连网络标准,其低延迟、高带宽的性能特征以及公开的协议标准都使其受到青睐,它是当今高性能计算集群节点间互连网络的主流。在作业调度性能评价方面,已有的众多研究并未将作业调度与集群系统的网络互连结构相结合,我们将通过模型分析的方式来研究典型的集中式作业调度系统的性能,并将其与 IB 互连网络相结合,来综合分析系统性能,如延迟等指标。

高性能系统性能评价的方法主要有:测量法(Measurement)、基准法(Benchmark)、仿真法(Simulation)和模型评价法(Model Evaluation)等[2,3,5,6,19,20]。其中,基于模型的性能评价在系统设计或采购阶段在性能预测、容量规划、软硬件采购等方面能发挥比较大的优势。随机 Petri 网(SPN)[7]是一种高级 Petri 网,特别适用于系统的性能评价,我们采用的方法将首先建立广义随机 Petri 网(GSPN)[8]模型,并通过精化的方法将其转化为随机高层网(SHLPN)[9];随后,再进行相关的性能评价。目前尚无使用随机 Petri 网对 IB 网络进行性能评价的研究,我们的方法还可以专门用来分析 IB 网络的性能。

到稿日期:2013-12-26 返修日期:2014-03-15 本文受国家科技重大专项(2102zx03039-004),中国科学院研究生科技创新专项资助。

李智佳(1986-),男,博士生,主要研究领域为 Petri 网的理论与应用,E-mail:lizj@ios.ac.cn;胡 翔(1989-),男,硕士生,主要研究领域为 Petri 网的理论与应用;焦 莉(1964-),女,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为并发理论与形式化方法;王伟锋(1989-),男,博士生,主要研究领域为实时系统的分析与验证。

本文第 2 节简要介绍随机 Petri 网；第 3 节介绍作业调度系统，建立抽象模型，使用随机 Petri 网技术分析其性能；第 4 节介绍 IB 网络互连，建立 IB 交换机的抽象模型，使用随机 Petri 网技术分析其性能；第 5 节将作业调度系统与网络互连相结合来综合分析系统性能；第 6 节通过实验验证我们方法的有效性；最后总结全文并提出下一步工作。

1 随机 Petri 网简介

一个随机 Petri 网(SPN)是一个六元组(S,T,F,W,M_0,λ)，其中，S 是库所的集合，代表系统所处的局部状态；T 是变迁的集合，代表导致系统状态发生改变的事件；$F\subseteq(S\times T)\cup(T\times S)$是流关系集合；$W:F\to N^+$是弧上的权函数；$M:S\to N$是 SPN 的标识函数，$M_0$ 代表初始状态标识；$\lambda=\{\lambda_1,\lambda_2,\cdots,\lambda_n\}$是变迁平均实施速率的集合，$\lambda_i$ 代表变迁 t_i 的平均实施速率。当变迁所有输入库所中的托肯数都大于等于相应的输入弧权值时，变迁可以实施。变迁实施后，SPN 从一个标识到达下一个标识，所有经由初始标识经过一系列变迁实施而到达的标识组成 SPN 的可达标识空间。一个 SPN 的可达标识空间同构于一个连续时间马尔科夫链(CTMC)。本文对 Petri 网及随机 Petri 网的相关基础知识不再作更进一步的介绍，文献[1,23]中有详细介绍。

2 作业调度系统建模与分析

2.1 作业调度系统简介

一般的调度系统由提交主机、主调度器及执行主机组成，用于接收用户提交的作业，为这些作业分配计算资源，并将它们分发到指定的主机去执行，之后，将执行结果反馈给用户。一般的作业调度系统都支持多种调度策略(如 LSF 中默认的是基于队列的先到先服务(FCFS)调度策略)，也支持公平调度及份额控制(Fairshare)、抢占式调度(Preemption)、独占式调度(Exclusive)、主机公平调度(HostParation)等调度策略。

2.2 作业调度系统抽象模型

根据 LSF 作业调度系统的工作原理[18]，我们采用文献[14,27]中介绍的方法，通过抽象得到其随机 Petri 网的模型，如图 1 所示。模型只包含 LSF 中的一个 default 到达队列 f，且单个的作业被分配到某一计算节点的单个 CPU 处理。作业到达后，首先被分配到指定的计算节点(u_i)，再分配具体的 CPU(d_{ij})用于执行作业。模型中，库所和变迁的含义如下：

f：代表作业等待队列，暂时保存未分配计算节点的作业；a_i：暂存被调度到计算节点 i 的作业；f_i：代表计算节点 i 的作业等待队列，暂时保存计算节点 i 内未分配处理器的作业；q_{ij}：代表计算节点 i 的处理器 j 的作业等待队列，保存等待处理器处理的作业。

c：代表客户端主机提交作业的过程；u_i：代表作业按照某种调度策略分配到计算节点 i；e_i：代表计算节点 i 将调度过来的作业添加到其等待队列的过程；d_{ij}：代表作业按照某种调度策略分配到计算节点 i 的处理器 j；s_{ij}：代表作业在处理器上处理的过程。

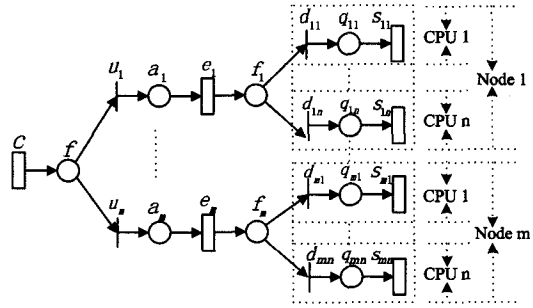

图 1 作业调度的抽象模型

2.3 模型精化与分析

本文采用对文献[27]中介绍的变迁可实施谓词和随机开关进行模型精化的方法来精化模型。图 2 是图 1 中的模型精化之后的模型。我们将原模型分解成多个子模型，子模型之间在结构上是独立的，每个子模型都代表了一个 M/M/1 排队系统，子模型中的变迁 d_{ij}、库所 q_{ij}、变迁 s_{ij} 分别代表计算节点 i 处理器 j 的排队系统的作业到达过程(到达速率为 λ_{ij})、作业等待队列(队列长度为 b_{ij})、作业处理过程(处理速率为 μ_{ij})。在下一节中，将介绍计算节点和处理器的调度算法，并给出相应的可实施谓词和随机开关的定义。

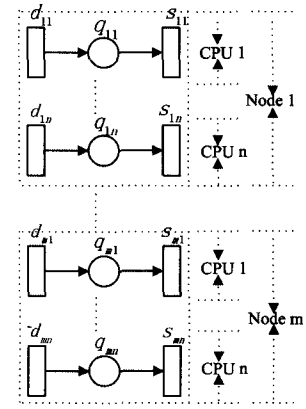

图 2 作业调度的精化模型

2.3.1 计算节点调度算法

采用总体最小期望等待时间调度，即在选择计算节点时，需要考虑每个计算节点内所有处理器等待队列的期望等待时间之和，将作业调度给总体最小期望等待时间的计算节点，从而获得更好的性能。

变迁 u_i 的可实施谓词 y_{ui} 用来约束变迁 u_i，判断其是否可实施，对于当前待分配的作业，在计算节点 i 的处理器队列不满的情况下，若分配给计算节点 i 的期望等待时间最小或者其他计算节点的处理器队列都满时，则按照调度算法将作业分配给计算节点 i，可实施的条件可用公式表示为：

$$y_{ui}:\left(\sum_{y=1}^{n}M(q_{iy})<\sum_{y=1}^{n}b_{iy}\right)\wedge\left(\left(\sum_{y=1}^{n}\frac{M(q_{iy})}{\mu_{iy}}=\min\left(\sum_{y=1}^{n}\frac{M(q_{1y})}{\mu_{1y}},\cdots,\sum_{y=1}^{n}\frac{M(q_{my})}{\mu_{my}}\right)\right)\vee\left(\forall k\neq i,\sum_{y=1}^{n}M(q_{ky})=\sum_{y=1}^{n}b_{ky}\right)\right)$$

变迁的 u_i 随机开关 p_{ui} 代表变迁 u_i 的可实施概率，即当多个计算节点同时满足可实施条件(y_{ui} 为 True)时变迁 u_i 实施的概率，其概率用公式可以表示为：

$$p_{u_i}(M)=\begin{cases}\dfrac{1}{|OSEDR(M)|}, & \text{if } i\in OSEDR(M)\\ 0, & \text{otherwise}\end{cases}$$

其中，$OSEDR(M)=\{k\mid \sum_{y=1}^{n}\frac{M(q_{ky})}{\mu_{ky}}=\min(\sum_{y=1}^{n}\frac{M(q_{1y})}{\mu_{1y}},\cdots,\sum_{y=1}^{n}\frac{M(q_{my})}{\mu_{my}})\wedge\sum_{y=1}^{n}M(q_{ky})<\sum_{y=1}^{n}b_{ky}\}$。

2.3.2 处理器调度算法

采用最小期望等待时间调度，即在选择某个计算节点内的处理器时，需要考虑该计算节点内每个处理器等待队列的期望等待时间，将作业调度给最小期望等待时间的处理器，从而获得更好的性能。

变迁 d_{ij} 的可实施谓词 $y_{d_{ij}}$ 用来约束变迁 d_{ij}，判断其是否可实施，对于当前待分配的作业，若分配给处理器 j 的期望等待时间最小，则按照调度算法将作业分配给处理器 j，可实施的条件可用公式表示为：

$$y_{d_{ij}}:(M(q_{ij})<b_{ij})\wedge((\forall k\neq j,\frac{M(q_{ij})}{\mu_{ij}}\leqslant\frac{M(q_{ik})}{\mu_{ik}})\vee(\forall k\neq j,M(q_{ik})=b_{ik}))$$

变迁 d_{ij} 的随机开关 $p_{d_{ij}}$ 代表变迁 d_{ij} 的可实施概率，即当多个处理器同时满足可实施条件（$y_{d_{ij}}$ 为 True）时变迁 d_{ij} 实施的概率，其概率用公式可以表示为：

$$p_{d_{ij}}(M)=\begin{cases}\dfrac{1}{|SEDR(M)|}, & \text{if } j\in SEDR(M)\\ 0, & \text{otherwise}\end{cases}$$

其中，$SEDR(M)=\{k\mid\frac{M(q_{ik})}{\mu_{ik}}=\min(\frac{M(q_{i1})}{\mu_{i1}},\cdots,\frac{M(q_{in})}{\mu_{ib}})\wedge M(q_{ik})<b_{ik}\}$。

2.3.3 作业调度系统的性能分析

对于标识 M，其稳定状态概率记为 $P(M)$。假定库所 q 代表一个容量为 b 的队列，则该位置的平均托肯数代表该队列的平均作业数目 $D(q)$，可以表示为：$D(q)=\sum_{y=1}^{b}y*P(M(q)=y)$。变迁 t 的利用率 $U(t)$ 等于使该变迁可实施的所有标识的稳定概率之和。如果该变迁代表处理器操作，则该变迁的利用率代表处理器的利用率，可以表示为：$U(t)=\sum_{M\in E}P(M)$，其中，E 是使得 t 可实施的所有可达标识的集合。变迁的吞吐量 $T(t)$ 等于变迁的利用率和平均实施速率之积，可以表示为：$T(t)=U(t)*\lambda$，其中，λ 是 t 的平均实施速率。在图 2 中的任意一个子模型中，根据排队论的知识，延迟时间 DT_{ij} 等于队列的平均作业数目除以表示离开队列的变迁的吞吐量，可以表示为：$DT_{ij}=D(q_{ij})/T(s_{ij})$，其中，$D(q_{ij})$ 是库所 q_{ij} 的平均托肯数，$T(s_{ij})$ 是变迁 s_{ij} 的吞吐量。调度的总延迟 $DT_{scheduling}$ 可以表示为：$DT_{scheduling}=(\sum_{i=1}^{m}\sum_{j=1}^{n}D(q_{ij}))/\sum_{i=1}^{m}\sum_{j=1}^{n}T(s_{ij})$。我们把队列为满时的概率定义为作业的丢失率 LR。当队列满时，到达的作业被抛弃掉，丢失率 LR 可以表示为：$LR=P(M(q)=b)$。有了以上公式，我们就可以对模型进行性能分析，这些性能指标都可以通过随机 Petri 网工具 SPNP[10] 求得。

3 IB 网络互连的建模与分析

3.1 IB 网络互连简介

IB 系统由信道适配器（Channel Adapter，CA）、交换机、路由器、线缆和连接器组成。CA 分为主机信道适配器（Host Channel Adapter，HCA）和目标信道适配器（Target Channel Adapter，TCA）。IB 交换机原理与其它标准网络交换机类似，但必须能满足 IB 的高性能和低成本的要求。IB 路由器用来把大网络分割为更小的子网，并用路由器连接在一起。HCA 是一个设备点，诸如服务器或存储设备的 IB 端节点通过该设备点连接到 IB 网络。TCA 是信道适配器的一种特别形式，多用于存储设备等嵌入式环境。

3.2 IB 交换机抽象模型

将交换机的功能抽象成两个部分：转发部分（将数据包转发给下一个交换机）和接收部分（将数据包转发到与该交换机连接的计算节点）。我们假定每个交换机的负载相同，即数据包的到达速率相同。

对于 $m\times m$ 2D-Mesh 的 IB 网络结构，由于其所在位置的不同，我们将区分 3 类交换机 S3（有 4 个相邻交换机）、S2（有 3 个相邻交换机）、S1（有两个相邻交换机）。对于交换机 S3，有 4 个端口可以允许数据包到达，如果转发该数据包，则选择 3 个端口转发数据包（即从交换机 1 转发到交换机 2 的数据包不会再重新转发回交换机 1），如果接收该数据包，则选择接收端口接收数据包。我们定义 4×3（4 个输入端口、3 个相邻的交换机）转发模型和 4×1（4 个输入端口，1 个与该交换机相连的计算节点接收数据包）接收模型来对该交换机建模。对于交换机 S2 和 S1，可以分别定义 3×2 转发模型和 3×1 接收模型、2×1 转发模型和 2×1 接收模型。我们只列出 4×3 转发模型和 4×1 接收模型，如图 3、图 4 所示。在转发模型中，从某个输入端口到达的数据包可以被转发给除其本身之外的 3 个输出端口。在接收模型中，从某个输入端口到达的数据包可以被转发到相应的接收端口。

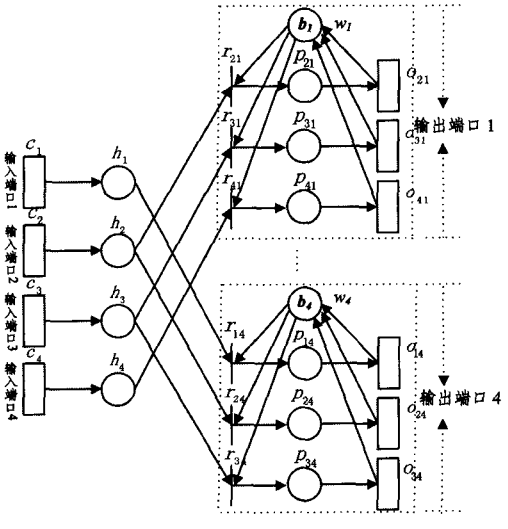

图 3 IB 交换机 S3 的 4×3 转发模型

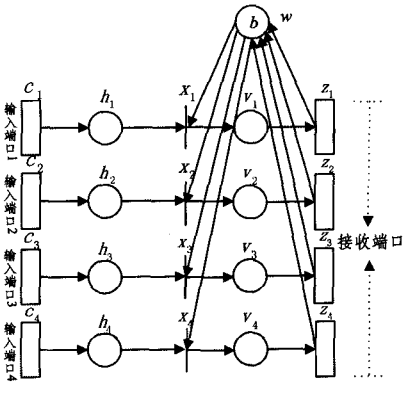

图 4 IB 交换机 S3 的 4×1 接收模型

模型中，库所和变迁的含义如下：

h_i：暂存从输入端口 i 到达未分配输出端口的数据包；p_{ij}：暂存从输入端口 i 到达被分配到输出端口 j 等待转发的数据包；v_i：暂存从输入端口 i 到达被分配到接收端口等待计算节点接收的数据包；w_j：代表输出端口 j 的数据包等待队列，托肯数 b_j 代表队列中的空闲位置个数；w：代表交换机中与计算节点相连的接收端口的数据包等待队列，托肯数 b 代表队列中的空闲位置个数。

c_i：代表数据包到达交换机输入端口 i 的过程；r_{ij}：代表从输入端口 i 到达的数据包被分配到输出端口 j；o_{ij}：代表从输入端口 i 到达的数据包经过输出端口 j 转发给相邻的交换机；x_i：代表从输入端口 i 到达的数据包按照某种调度策略被添加到接收端口的队列；z_i：代表数据包被接收的过程，即从输入端口 i 到达的数据包被转发给与该交换机相连的计算节点接收。

3.3 模型精化与分析

按照 2.3 节中作业调度系统的模型精化与分析的方法，类似地，我们可以对 IB 交换机模型进行精化和分析。对于 3.2 节中的模型，我们列出 4×3 转发模型和 4×1 接收模型的精化结果，如图 5、图 6 所示。其中，4×3 转发模型被精化成 12 个子模型，4×1 接收模型被精化成 4 个子模型，每个子模型代表一个排队系统。在下一节中，我们将会给出交换机的交换算法以及可实施谓词和随机开关。

3.3.1 IB 交换机的交换算法

本文中，我们不考虑交换机采用的路由算法，交换机采用随机均衡调度（转发），即对于从某个输入端口到达的数据包，交换机等概率地调度数据包到某一队列未满的输出端口进行转发。这种假设是为了方便分析，它对于实际情况也是一个比较好的近似。

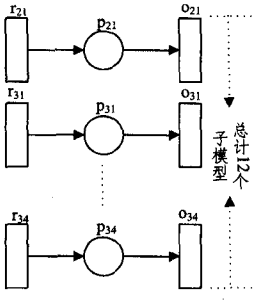

图 5　4×3 转发模型

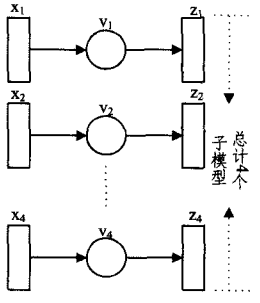

图 6　4×1 接收模型

转发模型中，变迁 r_{ij} 的可实施谓词 $y_{r_{ij}}$ 用来约束 r_{ij} 变迁，判断其是否可实施，对于从输入端口 i 到达的数据包，当输出端口 j 的等待队列中有空闲位置时，可以将该数据包分配给端口 j 进行转发，即变迁 r_{ij} 是可实施的，可实施的条件 $y_{r_{ij}}$ 可用公式表示为：$y_{r_{ij}}:\forall k\neq j,\sum_{k=1}^{4}M(p_{kj})<b_j$。转发模型中，变迁 r_{ij} 的随机开关 $p_{r_{ij}}$ 代表变迁 r_{ij} 的可实施概率，即对于从输入端口 i 到达的数据包，当多个输出端口的等待队列均有空闲位置时数据包被转发到端口 j 的概率。在随机均衡调度策略下，数据包在这些等待队列有空闲位置的输出端口中随机选择一个转发数据包，其概率 $p_{r_{ij}}(M)$ 用公式可以表示为：

$$p_{r_{ij}}(M)=\begin{cases}\dfrac{1}{|RR_i(M)|}, & \text{if } j\in RR_i(M)\\ 0, & \text{otherwise}\end{cases}$$

其中，$RR_i(M)=\{k\mid\forall k\neq i\wedge 0<k\leqslant 4\wedge y\neq k,\sum_{y=1}^{4}M(p_{yk})<b_k\}$ 表示等待队列中有空闲位置的输出端口集合。接收模型中，变迁 x_i 的可实施谓词和随机开关的定义与上述转发模型中的类似，故不再重复。

3.3.2 IB 交换机的性能分析

本文我们主要考虑的性能指标是交换延迟，计算方法同 2.3.3 节一致。对于 $m\times m$ 2D-Mesh 的 IB 网络结构，根据交换机的类型（S3、S2 或 S1）计算出每种类型交换机的转发延迟，然后对 3 种类型的交换机的转发延迟取平均数，记为数据包通过交换机的转发延迟。平均接收延迟通过同样的方法求取。

4 作业调度与网络互连的综合分析

接下来，以延迟为例说明如何将作业调度系统与网络互连结合起来综合分析。用户提交的作业等待时间包括：作业调度及作业被处理器处理的时间 $T_{scheduling}$、作业的传输时间 $T_{transmission}$。作业传输时间指当调度系统为作业分配好计算节点后，大小为 L 的作业从提交节点到处理器的网络传输时间。我们定义作业的延迟为：$Latency=T_{scheduling}+T_{transmission}$。其中，$T_{scheduling}$ 可以通过 2.3 节的结果求得。$T_{scheduling}=\frac{L_{job}}{L_{packet}}(\frac{2(m^2-1)}{3m}*\frac{L_{packet}}{W}+(\frac{2(m^2-1)}{3m}-1)*T_{forward}+T_{accept})$，其中，$L_{job}$ 表示作业的长度，L_{packet} 表示数据包的长度，L_{job}/L_{packet} 表示数据包的个数。$2(m^2-1)/3m$ 表示 $m*m$ 的 2D-Mesh 拓扑结构的平均跳数，W 表示带宽，L_{packet}/W 表示信道传输时间。$T_{forward}$ 与 T_{accept} 可以通过 3.3 节给出的方法求得。

从公式可以看出，作业延迟与调度的延迟 $T_{scheduling}$、作业大小 L_{job}、IB 数据包长度 L_{packet}、网络拓扑参数 m、IB 网络带宽 W、交换机-交换机延迟 $T_{forward}$、交换机-节点延迟 T_{accept} 有关。其中，作业大小、IB 数据包长度、网络拓扑参数、IB 网络带宽都是固定的，不需要求解；而 $T_{scheduling}$、$T_{forward}$、T_{accept} 则可以通过 2.3 节与 3.3 节的分析求得，在下一节中我们将通过实验求得 $T_{scheduling}$、$T_{forward}$ 及 T_{accept}。

5 实验

实验中我们采用的交换模型为 2×2（即两行两列交换机），调度模型为 4×2（即 4 个计算节点，每个计算节点 2 个处理器），4 个交换机分别连接到 4 个计算节点。在实验中，设置等待队列 b 的长度为 4，交换机的处理速率均为 60packets(jobs)/s[1]，节点 1 的两个 CPU 处理速率为 60 jobs/s，节点 2 的两个 CPU 处理速率为 50jobs/s，节点 3 的两个 CPU 处理速率为 40jobs/s，节点 4 的两个 CPU 处理速率为 30jobs/s。通过 SPNP（随机 Petri 网软件包）[10] 可以对上述调度算法中的可实施谓词和随机开关进行设置，进而对该随机 Petri 网进行性能分析。以下是一些实验结果和分析。

[1] 为了与调度系统的处理单位（作业）保持一致，方便实验参数的描述与实验结果的图形绘制，这里也使用作业来替换交换机的处理单位（数据包），不影响对交换机的性能分析。

图 7 是延迟的分析结果。从图中可以看出，调度系统的作业延迟和网络互连的交换机转发和接收延迟，都随着作业到达速率的增加而增加。计算节点利用率的实验结果如图 8 所示。从图中可以看出，由于计算节点处理速率的不同，处理速率高的计算节点的利用率会低于处理速率低的计算节点的利用率。作业的丢失率和队列平均等待数的实验结果如图 9 和图 10 所示。对于当前的系统配置，实验结果显示其丢失率很低，队列平均等待数小于 1，说明队列容量的设置是合理的，但从图中可以看出，随着到达率的增加，作业的丢失率和队列平均等待数目快速增加。在实际的系统设计中，可以使用该方法检验不同的队列容量设置是否满足实际需求，进而找出满足需求的最优设置。

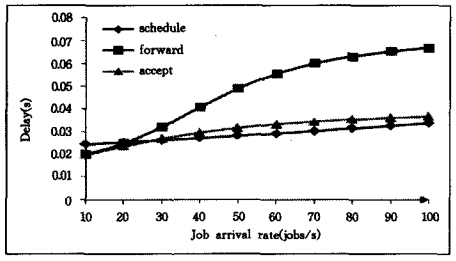

图 7　延迟时间(调度、转发、接收)

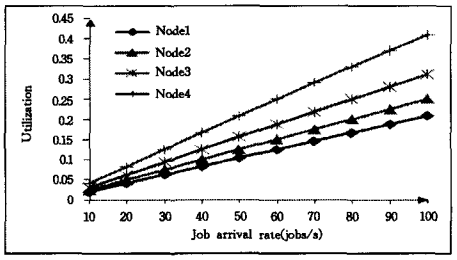

图 8　计算节点的利用率

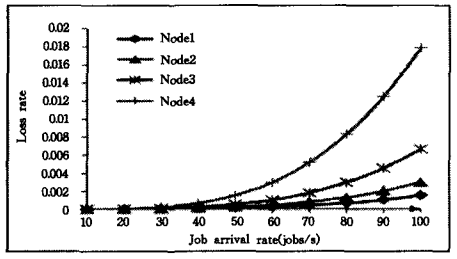

图 9　作业的丢失率

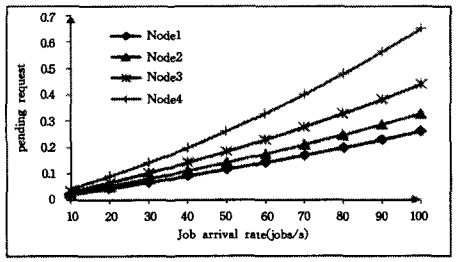

图 10　队列中平均等待作业的数目

我们的模型仿真结果与分析结果的比较如图 11 所示，可以看出，随着到达率的增加，仿真延迟的增长率没有分析的增长率稳定，增长曲线不圆滑，这从一个侧面反映了基于状态空间分析方法的优点(稳定性高)，但仿真的延迟与分析的延迟数值十分接近，具有较高的精度，因此所提模型可以用来对规模较大的系统进行建模分析。

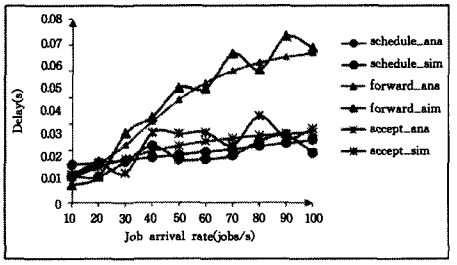

图 11　分析(ana)与仿真(sim)的结果对比

结束语　在本文中，我们将集群系统的作业调度与 IB 网络互连相结合，使用基于随机 Petri 网的性能评价技术对它们进行了性能分析，详细给出了延迟这一性能指标的计算方法，并通过实验验证了这一性能分析方法的可行性。在接下来的工作中，我们将进一步降低对集群系统的抽象程度，以提高模型的准确性；此外，我们还将比较本文的方法与其他性能评价方法在分析集群系统作业调度及 IB 网络互连性能上的优劣。

参 考 文 献

[1] Murata T. Petri nets: properties, analysis and applications[J]. Proceedings of the IEEE, 1989, 77(4): 541-580

[2] Petrini F, Kerbyson D J, Pakin S. The case of the missing supercomputer performance: achieving optimal performance on the 8, 192 processors of ASCI Q[C]// Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, 2003. Phoenix: ACM, 2003: 55

[3] Hu L, Gorton I. Performance evaluation for parallel systems: A Survey[R]. Sydney: University of NSW, 1997

[4] Ciardo G, Cherkasova L, Kotov V, et al. Modeling a scalable high-speed interconnect with stochastic Petri nets[C]// Proceedings of the Sixth International Workshop onPetri Nets and Performance Models, 1995. Durham: IEEE Computer Society Press, 1995: 83-92

[5] Jain R. The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling[M]. New York: John Wiley & Sons, 1991

[6] Brewer E A, Dellarocas C N, Colbrook A, et al. PROTEUS: a high-performance parallel-architecture simulator[C]// Proceedings of the 1992 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, 1992. Newport: ACM, 1992: 247-248

[7] Florin G, Natkin S. Evaluation based upon stochastic petri nets of the maximum throughput of a full duplex protocol[C]// Girault C, Reisig W, eds. Berlin: Springer-Verlag, 1982: 280-288

[8] Marsan M A, Conte G, Balbo G. A class of generalized stochastic Petrinets for the performance evaluation of multiprocessor systems[J]. ACM Transactions of Computer Systems, 1984, 2(2): 93

[9] Lin C, Marinescu D C. Stochastic high-level Petri nets and applications[J]. IEEE Transactions on Computers, 1988, 37(7): 815

[10] Ciardo G, Muppala JK, Trivedi KS. SPNP: Stochastic Petri Net Package[C]// Proceedings of 3rd International Workshop on Petri Nets and Performance Models, 1989. Kyoto: IEEE Computer Society, 1989: 142-150

(下转第 46 页)

挖掘语言特性，能够实现程序对计算资源的自动感知，并精细设计程序结构，实现数据初始化、数据划分、任务分解等功能。

Julia 语言提供了函数 nprocs()得到 Julia 当前运行环境中可用的处理单元个数。用 np=nprocs()感知系统环境的处理单元个数，则可将 np 变量用作数据划分和任务分配的参数，由程序自身适应不同运行环境下的计算资源数量而自动调优。

鉴于 Julia 数组的支持暂未能达到 C 语言中的灵活度，比如不能把带下标的子数组看作一个独立数组(因为是指针)，本文 Julia 程序中子数组的名字与处理单元个数相关联，在远程调用中需要严格对应。充分利用 Julia 程序的脚本语言特性，程序可先生成程序文件，并在后面程序执行过程中的适当位置包含进来(include 指令)，实现程序对处理单元个数的自适应效果，包括数据的自动划分和远程调用中的参数匹配等。

分布式集群上对大数组的划分和数据传输都是时间消耗的重要来源，且需要更多的存储空间，使计算规模受到限制。如果把数据文件存于共享目录，各处理器按照特定规则读取文件中的指定数据段，则可提升效率。我们测试了 4 核 CPU 并行读取数据文件情况下的程序执行效率，效率对比数据如表 2 所列。

表 2　不同读文件方式下的程序执行效率对比

时间(s)	主控读入	分布读
数据准备	1.212	1.096
分配	6.258	0.97,1.12,1.14,1.11
合成结果	1.56	1.58
合计	9.03	≈3.8

结束语　本文通过公交车次站间走行时间统计案例，尝试了 Julia 并行程序的精化过程，并对 Julia 在平台自适应、规模动态扩展等方面进行了探讨。Julia 作为脚本语言，其优秀之处在于其编程的容易性。通过为特定领域的应用程序开发高层算法库，隐式支持并行算法实现，能进一步提高其易用性。Julia 语言提供了丰富的高效的函数库，比如通常情况下使用 Julia 自带的 mean()函数对数组的指定维度求平均值，性能明显高于自编程序。Julia 能够方便地与其他语言接口(比如 C 语言和 Fortran 语言)，或调用外部程序，使其容易适应平台，充分利用平台的计算能力。如果用 CUDA/C 制作动态库，通过 Julia 程序调用，可利用节点上的 GPU 协处理器，加速应用程序中的关键算法。

参 考 文 献

[1] http://en.wikipedia.org/wiki/MIT_License

[2] http://julialang.org/

[3] Mathematics and Computer Science Division Argonne National Laboratory. MPICH User's Guide (Version 3.0.4)[OL]. [2013-10]. http://www.mpich.org/documentation/guides/. Apr. 24,2013

[4] 李润梅，刘建忠，朱凤华. 平行公交系统中的计算实验问题研究[J]. 自动化学报，2013,39(7):1011-1017

[5] http://bus.17u.com/bus/beijing/

[6] 张常有，张先轶. Julia 语言与并行计算[R]. 第 6 届 R 语言大会主题报告. 北京：人民大学

(上接第 37 页)

[11] Zhou S N,Zheng X H,Wang J W,et al. UTOPIA:a load sharing facility for large, heterogeneous distributed computer systems [J]. Software:Practiceand Experience,1993,23(12):1305-1336

[12] Mu'alem A W,Feitelson D G. Utilization,predictability,workloads,and user runtime estimates in scheduling the IBM SP2 with backfilling[J]. IEEE Transactionson Parallel and Distributed systems,2001,12(6):529-543

[13] Sherwani J,Ali N,Lotia N,et al. Libra:a computational economy-based job scheduling system for clusters[J]. Software:Practice and Experience,2004,34(6):573

[14] Shan Z G,Lin C. Modeling and performance evaluation of hierarchical job scheduling on the grids[C]//Proceedings of the 2007 International Conference on Grid and Cooperative Computing, 2007. Urumchi. IEEE Computer Society,2007:296-303

[15] Naik V K,Setia S K,Squillante M S. Performance analysis of job scheduling policies in parallel supercomputing environments[C]// Proceedings of the 1993 ACM/IEEE conference on Supercomputing,1993. New York:ACM,1993:824-833

[16] Moschakis I A,Karatza H D. Evaluation of gang scheduling performanceand cost in a cloud computing system[J]. The Journal of Supercomputing,2012,59(2):975

[17] Abawajy J H. An efficient adaptive scheduling policy for high-performance computing[J]. Future Generation Computer Systems,2009,25(3):364

[18] Platform Computing Corporation. Running Jobs with Platform LSF[OL]. http://www-03.ibm.com/systems/services/platformcomputing

[19] Palma J N. Performance evaluation of interconnection networks using simulation-tools and case studies[D]. Bilbao:University of the Basque Country,2009

[20] Sur S,Koop M J,Chai L,et al. Performance analysis and evaluation of MellanoxConnectXInfiniBand architecture with multi-core platforms[C]//Proceedings of the 15th Annual IEEE Symposium on High-Performance Interconnects, 2007. Stanford: IEEE,2007:125-134

[21] Mellanox. MellanoxInfiniBandTraining[OL]. http://www.mellanox.com/

[22] 超级计算机 TOP500 排名[OL]. http://www.top500.org/

[23] 林闯. 随机 Petri 网和系统性能评价(第 2 版)[M]. 北京：清华大学出版社，2005

[24] 陈永然. 面向高性能计算的性能评价模型技术研究[D]. 长沙：国防科学技术大学，2007

[25] 王翠萍. LSF 系统中作业调度的研究与优化[D]. 西安：西安电子科技大学，2009

[26] 曹宗雁. 高性能计算集群运行时环境的配置优化[J]. 科研信息化技术与应用，2011,2(6):52-61

[27] 林闯. 随机 Petri 网模型的精化设计[J]. 软件学报，2000,11(1):104-109