探索计算系统异构性的描述*)

曾国荪　陈闵中
(同济大学计算机科学及工程系　上海200092)
(国家高性能计算机工程技术中心同济分中心　上海200092)

Inquiring the Description of Heterogeneity among Computational Systems

ZENG Guo-Sun　CHEN Hong-Zhong
(Dept. of Computer Science and Engineering, Tongji University, Shanghai, 200092)
(Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai 200092, China)

Abstract It is valuable work to construct computational system by means of network, so that its capacity and power can be enhanced and extended. But there is heterogeneity in common among computational systems. It is main obstacle for computational systems to cooperate well. This paper inquires the sound description of heterogeneity for computational systems, and presents several appoaches on definition of heterogeneity. That is significant to go a step further for hiding or exploring heterogeneity in the future.

Keywords Computational systems, Heterogeneity, Heterogeneity degree, Heterogeneity definition

1 引言

近年来，随着计算机和通信技术的高速发展，因特网、Web、网格正在变得越来越普及，包含科技、文化、商业、新闻、娱乐等多种形式和类别的信息系统像雨后春笋般出现，使得信息不仅可在大范围共享，而且还能开展分布计算、并行计算、异构计算、元计算等服务。因特网、Web、网格得以成功源于其开放性、分布性、自由性、共享性等特征。然而这些成功的因素也使得因特网信息资源伴随许多负面特性：(1)非控制性，因特网是一个多网络、无中心、无主管的分散型互联网结构，网上信息资源具有分布式特点，处于一种无中心控制、混乱无序的分布状态。(2)非均衡性，网上信息资源在不同学科专业领域、不同行业、不同地理位置上的分布差异很大，数量和质量的差别也很大。(3)异构性，信息资源在形式上没有一个统一的体系和结构，处于非结构化状况，更新周期不一。人类的种族、文化、语言也是千差万别的。(4)动态性，网络中的信息资源分散无序，变动和更换频繁。(5)自治性，网上的每个小型信息系统，都有各自的运行、管理和维护规则，甚至"占山为王，各自为政"，但其至少在局部是有生命和可靠的。

全球因特网上如此众多的信息系统，任意两者能够进行通信，存取关资源，远程执行指令，交互和合作等是最基本最自然的要求，但却是一个非常难于解决的课题，其最大的障碍就是信息系统的异构性。也就是说单个信息系统之间，在使用的计算机硬件平台、数据库系统、软件版本、信息内容等等方面都存在差异。在提倡个性化的时代，这种差异有不断加大的趋势，进一步增加了信息系统之间资源存取的复杂程度。众所周知，异构性是客观世界的一个普通属性，具有积极和消极两个方面，信息系统的异构性也是这样。本文正是研究计算系统异构性的客观规律，探索异构性的合理化描述，为开发和利用异构性以便增强信息系统的功能，屏蔽和抑制异构性以便拓宽信息系统的互操作和可用性等方面提供基础理论的支持。

2 异构性是普遍存在的

从哲学的角度来说，同构是相对，异构是绝对的。基于因特网的信息系统无处不存在异构性，主要体现在以下方面：

计算机硬件平台的异构性：信息系统进行数据处理的服务器可以分别是大型机、小型机、工作站、PC或嵌入式系统。这些机器除了生产厂商和型号的不同之外，关键是计算机体系结构和指令系统上的异构。对于大型机而言，多以追求高性能和并行计算为目的。例如单指令多数据流计算机SIMD，用来开发数据并行性；并行向量处理机PVP，用来开发规约运算并行性；对称多处理机SMP、大规模并行处理机MPP、工作站机群COW、分布共享存储多处理机DSM，多用来挖掘功能并行性和流水并行性。对于桌面电脑而言，强调的是易使用性。总之，每种机型都有其相对擅长和被选择的应用领域。在数据存储存取方面，CPU内部的寄存器、Cache、半导体内存、磁盘存储器、磁带、光盘等构成了存储系统的层次结构。尽管在同一存储系统中，数据的含义在各个层次上保持一致，但数据的格式、字节顺序、组织形式、存取模型、存取性能等都不一样，进一步放大了存储系统之间的差异性。在外部设备方面，所有的计算机输入、输出设备直接构成和展示了丰富多彩的计算机世界[1~3]。

基础操作系统的异构性：各个信息系统的基础操作系统可以是Unix、Windows NT、Linux等。Unix的OS内核可能基本相同，但被扩展成支持主从模型、事务模型、浮点执行模型、多线程模型等多种版本。操作系统之间的文件系统、命名规则、用户权限管理、文件类型、操作命令的解析和执行、进程之间的通信机制不可能相同。显然，NT下的可执行文件在Unix是无法执行的。期望统一的软件标准，一直是软件界的梦想，即便是消除操作系统之间的鸿沟，至今也无法实现。

*)本课题得到国家自然科学基金(60173026)和上海市自然科学基金(01ZD14067，025115037)资助。曾国荪　博士，教授，主要研究领域为并发理论，异构计算，计算网格。

数据库管理系统的异构性：信息经历了手工组织、文件存储到数据库管理的历史过程。当前每个信息系统都利用了一个或多个数据库系统。最流行的是关系型数据库系统，如Oracle、Sybase、SQL Server 等。也可采用由不同数据模型的数据库，如关系、模式、层次、网络、工程、面向对象、函数型数据库共同组成一个异构数据库系统。加入异构数据库系统之中的每个子数据库在此之前本身就已经存在，并且保留有自己的应用特性、完整性和安全控制机制。在数据库编程方面，能够选择 ODBC、JDBC 接口，也可嵌入宿主语言中。

通信网络的异构性：当前通信技术非常发达，可以采用电缆、光纤、微波来进行接入。总线、环、星型、树型等可以构造所希望的任一种网络拓扑结构，导致全球纷繁复杂的因特网。通信带宽、传输速率、路由和交换策略等，是标记和区别一个网络的品质因素。点对点、点对多、广播、广收、多对多、同步、异步等多种通信需求，说明了不能只采用固定单一的通信方式。由于历史和领域等原因，多种通信协议如：X.25、DDN、ISDN、帧中继等共网交相并存，阻碍了通信数据包在目的端口正确解析[4]。

应用程序和服务的异构性：应用程序是程序员利用某种语言和工具，为完成一定功能而开发出来的一个软件执行模块，它的生命周期一般依赖特定的操作系统平台和相应的环境，对于完成不同任务的应用程序之间无共性可言。而不同信息系统内功能相同的执行程序之间，也存在很大的差异。如编程思想，是采用面向结构，还是面向对象方法；还是采用共享内存，还是消息传递方式；是采用汇编语言，还是高级语言；是初级编程员，还是高级工程师等。应用程序和服务的异构性还表现在对非编程员和用户的不确定性。应用程序的类型不知道，程序的功能是一个黑盒，难于准确把握。程序的运行长度，内部包含子任务、子进程的数目，子任务之间的通信模式，原始处理数据的许可形式，子任务调度执行策略，任务脚本(profile)，优先权的要求，服务质量(QoS)的要求，死期(deadline)的要求等都可能是不定和未知的。总之，应用程序和服务的时间和空间属性被程序名封装而屏蔽和隐藏掉了[5]。

信息的异构性：多出现在广域、多组织、大规模信息资源重用的时候，特别是基于因特网的信息系统内部。表现在语法(syntax)、语义(semantics)、语用(pragmatics)等方面的冲突。例如术语和命名冲突：即源模型中的标识符可能是目的模型中的保留字，这时就需要重新命名。格式冲突：同一种数据类型可能有不同的表示方法和语义差异。结构冲突：如果两种数据库系统之间的数据定义模型不同，如分别为关系模型和层次模型，那么需要重新定义实体属性和联系，以防止属性或联系信息的丢失。尤其对自然语言理解和翻译，信息的异构性特别突出[6]。

3 异构性的定义

关于异构性的形式化定义，目前还没有统一和公认的方法，下面从几个侧面给出计算系统中异构性抽象定义的一些探索。

3.1 同构术语的代数定义

代数系统是离散数学的一个分支，常作为信息系统高度抽象的工具，其原理可帮助理解系统内部概念和本质规律。

定义1 设 $V_1=(A,\alpha_1,\alpha_2,\cdots,\alpha_n)$，$V_2=(B,\beta_1,\beta_2,\cdots,\beta_n)$ 是两个代数系统，其中 A，B 是非空对象集合，$n\geqslant 1$，α_i，β_i 是运算子。若对于 $i=1,2,\cdots,n$，α_i 和 β_i 运算子都具有相同的元数，则称 V_1 和 V_2 是同类型的代数系统。

定义2 设 $V_1=(A,\alpha_1,\alpha_2,\cdots,\alpha_n)$，$V_2=(B,\beta_1,\beta_2,\cdots,\beta_n)$ 是两个代数系统，对于 $i=1,2,\cdots,n$，α_i 和 β_i 是 k_i 元运算，函数 $\varphi:A\to B$，如果对所有的运算 α_i 和 β_i 都有：$\forall x_1,x_2,\cdots,x_{k_i}\in A$，$\varphi(\alpha_i(x_1,x_2,\cdots,x_{k_i}))=\beta_i(\varphi(x_1),\varphi(x_2),\cdots,\varphi(x_{k_i}))$ 成立，则称 φ 是代数系统 V_1 到 V_2 的同态映射，简称同态。

定义3 设 $V_1=(A,\alpha_1,\alpha_2,\cdots,\alpha_n)$，$V_2=(B,\beta_1,\beta_2,\cdots,\beta_n)$ 是两个代数系统，$\varphi:A\to B$ 是 V_1 到 V_2 的同态，并且 $\varphi:A\to B$ 是双射的，则称 φ 是同构，也称 V_1 同构于 V_2。

3.2 并行计算领域异构的定义

从抽象代数的观点看，如果代数系统 V_1 同构于 V_2，它们是没有区别的，是同一个代数系统。但上述定义过于严格，而且实际应用中寻找到定义3中要求的函数 φ 往往十分困难。因此，当难于找到这样的函数 φ 时，保守地认为两个系统是异构的。下面是并行计算领域异构的定义。

定义4 计算系统是异构的，如果系统中的处理器的类型、处理器的速度、处理器的数量、计算模式、内存容量、互连网络、通信模式等当中的一项或多项发生改变，将导致不同的执行性能[7]。

定义4包含两层含义，其一是说变化前和变化后的系统，如果执行同样一个应用任务，性能不一样，则前后系统是异构的。其二是说一大规模计算系统由许多不同种类型的计算模块和设备组成。后者含义包括前者。

以下我们用 $MH=(M_1,M_2,\cdots,M_m)$ 表示异构并行计算系统，其中 M_i 是特种类型的处理机或工作站，相互之间是异构的。M_i 的计算能力由其 CPU、I/O 和内存等综合决定。令 $S_i(A)$ 和 $T(A,M_i)$ 分别表示 M_i 单独解决和执行应用任务 A 的速度和完成时间，$T(A,MH)$ 表示整个异构系统合作完全 A 所需的时间，那么可用 $W_i(A)=S_i(A)/\max_{j=1}^{m}S_j(A)$，$i=1,2,\cdots,m$，来表示 M_i 运行程序 A 在计算能力上的权重，由此我们可定义异构加速比和异构度等概念。

定义5 异构并行计算系统 $MH=(M_1,M_2,\cdots,M_m)$ 执行程序 A 的加速比为：$SP(A)=(\min_{j=1}^{m}T(A,M_j))/T(A,MH)$。

定义6 异构并行计算系统 $MH=(M_1,M_2,\cdots,M_m)$ 执行程序 A 的并行度为：$P_{degree}(A)=(\sum_{j=1}^{m}T_j^{active})/T(A,MH)$，其中 T_j^{active} 为执行 A 期间，每个处理机 M_j 分别被激活的时间。

定义7 异构并行计算系统 $MH=(M_1,M_2,\cdots,M_m)$ 执行程序 A 的异构度为：$H_{degree}(A)=(\sum_{j=1}^{m}(1-W_j(A)))/m$。

定理1 异构并行计算系统 $MH=(M_1,M_2,\cdots,M_m)$ 执行程序 A 时，公式 $SP(A)=P_{degree}(A)\times(1-H_{degree}(A))$ 成立。

证明，略，参见文[8]。

3.3 程序的异构特征定义

程序千差万别，算法多种多样，这正是软件的魅力，又是程序设计和实现的困难点。异构性客观存在，正确开发异构性能获得好处。那么，我们应该从复杂的程序中抽象出哪些特征，以便实现异构计算呢？因为程序的设计和实现总是和机器捆绑在一起的，由此想到计算机的类型。Flynn 曾把机器分成四种，常用的只有 SISD、SIMD、MIMD 这三种，所以自然会想到将程序段和算法也抽象成 SISD、SIMD、MIMD 三种。因这方面的国际研究很少，我们初步这样定义，觉得有一定的合理

性。由于篇幅限制，我们将在后续研究和论文中报告基于粒度、模式、语义等其它程序异构特征的定义方法。

下面借用进程代数的表示方法和书记符号，展示一种程序异构特征的定义方法。

3.3.1 结构异构性定义

定义8 设有进程P和数据集D，D可划分为多个子集$D=\{d_1,d_2,\cdots,d_n\}$，$d_i\neq d_j$，$i,j=1,2,\cdots,n$，如果$P.D=P.d_1|P.d_2|\cdots|P.d_n$成立，则称P对D具有数据并行性[9]。

定义9 对于程序P和数据D，如果P对D具有数据并行性，且P中语句数$|P|>1$，则称P具有SPMD(单程序多数据)结构；若$|P|=1$时，称P具有SIMD结构，分别记为$P\vdash SPMD$、$P\vdash SIMD$。

数据并行性简称对不同数据作相同的处理，它是客观存在的。开发应用程序的数据并行性，关键在于数据集合的划分。面向数组的编程思想和程序设计语言正是为了利用数据并行性。许多矩阵运算问题，存在大量数据并行性。一般说来，具有统一、固定计算量的算法，潜藏数据并行性。

定义10 设有进程P和数据集D，P可划分为$P=\{p_1,p_2,\cdots,p_n\}$，$p_i\neq p_j$，D可划分$D=\{d_1,d_2,\cdots,d_n\}$，$d_i\neq d_j$，$i,j=1,2,\cdots,n$，如果$P.D=p_1.d_1|p_2.d_2|\cdots|p_n.d_n$成立，则称P对D具有任务并行性，或称P对D具有功能并行性[9]。

定义11 对于程序P和数据D，如果P对D具有任务并行性，则称P具有MIMD结构，记为$P\vdash MIMD$。

任务并行性简称对不同的数据独立地做不同的处理。它很直观，任何不同功能的两个操作同时执行，便可认为具有任务并行性。因此，任务并行性具有多种灵活的表现形式。在程序任务图中，fork节点的各个子节点，join节点的各个父节点，潜藏任务并行性。基于多道程序和多线程的设计思想，其目的是为了开发程序中的任务并行性。

直觉上，数据并行性和任务并行性可通过粒度上的调整相互转换，但是我们在定义10中要求$p_i\neq p_j$，这样能够严格区分这两种并行性，而不致于引起混乱。

定义12 对于程序P，如果$P=\alpha_1.\alpha_2.\cdots.\alpha_n$，则称P具有SISD结构，记为$P\vdash SISD$。

显然，不可并行化的程序，即通常所说的串行程序具有SISD结构。

定义13 设有程序P，数据集$D=\{d_1,d_2,\cdots,d_n\}$，$d_i\neq d_j$，$i,j=1,2,\cdots,n$，转换规则集$R=\{r_1,r_2,\cdots,r_m\}$，如果$P\xrightarrow{r_1}P'\xrightarrow{r_2}\cdots\xrightarrow{r_m}P'=P^*.d_1|P^*.d_2|\cdots|P^*.d_n$成立，则称P蕴含SIMD结构，记为$P\rightarrow SIMD$。

定义14 设有程序P，数据集$D=\{d_1,d_2,\cdots,d_n\}$，$d_i\neq d_j$，$i,j=1,2,\cdots,n$，转换规则集$R=\{r_1,r_2,\cdots,r_m\}$，如果$P\xrightarrow{r_1}P'\xrightarrow{r_2}\cdots\xrightarrow{r_m}P'=\{P_1^*,P_2^*,\cdots,P_n^*\}=P_1^*.d_1|P_2^*.d_2|\cdots|P_n^*.d_n$成立，则称P蕴含MIMD结构，记为$P\rightarrow MIMD$。

定义9和定义11的条件要比定义13和定义14强。在初始开展异构计算时，可暂不严格区别$P\vdash SIMD$和$P\rightarrow SIMD$，以及$P\vdash MIMD$和$P\rightarrow MIMD$。定义13和定义14中的规则集$R=\{r_1,r_2,\cdots,r_m\}$，有时表现为等效变换规则，有时表现为划分规则，有时表现为归约规则，总之要求保证程序语义不变。

上述基于结构的程序异构性定义，与机器，即执行模式无关。

3.3.2 性能异构性定义

定义15 对于程序P(包括代码和数据)，给定三台分别为SISD、SIMD、MIMD体系结构的计算机，分别记为M_{SISD}、M_{SIMD}、M_{MIMD}，令$\tau(P)|M_s$表示P在特定计算机s上所需的执行时间，则：

(1)当$\tau(P)|M_{SISD}=\min(\tau(P)|M_{SISD},\tau(P)|M_{SIMD},\tau(P)|M_{MIMD})$，在此特定条件下称P具有SISD优化的，记为$P\models SISD$。

(2)当$\tau(P)|M_{SIMD}=\min(\tau(P)|M_{SISD},\tau(P)|M_{SIMD},\tau(P)|M_{MIMD})$，在此特定条件下称P具有SIMD优化的，记为$P\models SIMD$。

(3)当$\tau(P)|M_{MIMD}=\min(\tau(P)|M_{SISD},\tau(P)|M_{SIMD},\tau(P)|M_{MIMD})$，在此特定条件下称P具有MIMD优化的，记为$P\models MIMD$。

定义16 对于程序P，任意给定三台分别为SISD、SIMD、MIMD体系结构的计算机，分别记为M_{SISD}、M_{SIMD}、M_{MIMD}，令$t(P)|M_s$表示P在特定计算机s上执行所需的CPU时钟周期数，则：

(1)当$t(P)|M_{SISD}=\min(t(P)|M_{SISD},t(P)|M_{SIMD},t(P)|M_{MIMD})$，称P具有理想SISD优化的，记为$P=SISD$。

(2)当$t(P)|M_{SIMD}=\min(t(P)|M_{SISD},t(P)|M_{SIMD},t(P)|M_{MIMD})$，称P具有理想SIMD优化的，记为$P=SIMD$。

(1)当$t(P)|M_{MIMD}=\min(t(P)|M_{SISD},t(P)|M_{SIMD},t(P)|M_{MIMD})$，称P具有理想MIMD优化的，记为$P=MIMD$。

定义15的条件比定义16条件要弱得多。在定义16中，t是时钟周期数，而非执行时间，这样可消除同类机器由于主频不一致造成的影响。程序具有SISD、SIMD、MIMD优化的特征是程序在特定条件和时刻的在线运行性质，离开了特定条件将没有意义。程序理想SISD、SIMD、MIMD优化的特征是全局的，反映了程序的内在性质，但难于测定，因为不能穷尽所有计算机的执行情况。程序具有SISD、SIMD、MIMD优化的特征是局部的，可以检测的。所以，验证程序异构性时，只能采用定义15。但我们注意到，一般不能保证$P\models SISD\Rightarrow P=SISD$，$P\vdash SIMD\Rightarrow P=SIMD$，$P\models MIMD\Rightarrow P=MIMD$。反之$P=SISD\Rightarrow P\models SISD$，$P=SIMD\Rightarrow P\models SIMD$，$P=MIMD\Rightarrow P\vdash MIMD$能够成立。程序具有哪种优化的性质随用以测定的一组计算机变化而变化，即与执行模式相关的。所以，区分程序具有SISD、SIMD、MIMD哪一种优化的特征，只有在某种特定的环境下才有意义。事实上，一个高性能计算实验室只可能有有限类型和数量的机型，只能从已有的计算资源上，谋求计算性能的优化。

另一方面，程序具有SIMD、MIMD结构的特征和程序具有理想SIMD、MIMD优化的特征之间究竟有什么关系？能否保证$P\vdash SIMD\Leftrightarrow P=SIMD$和$P\vdash MIMD\Leftrightarrow P=MIMD$成立？这些问题值得深入研究。

结语 本文详细陈述了计算系统普遍和大量存在异构性，指出异构性是信息系统互操作和充分发挥潜能的最大障碍。为了解决异构性的合理化描述问题，论文用抽象代数理论给出了同构的定义，它的反面即是异构。而且从特定的并行计算领域，给出了系统是异构的定义，并且提出了异构度这一新概念，该参数用来把握计算系统负载平衡是一个非常有力的参量。从机器的类型即执行模式出发，我们还给出了基于结构和基于性能的程序异构特征的定义方法，为开展异构计算提供了智能调度信息。

在后续的论文和将来的研究中，我们将介绍基于粒度、模式、语义等的程序异构特征定义方法，以及自动提取程度特征

(下转第22页)

(4)开发 DNA 海量存储器。DNA 具有天然的海量存储功能,因此,如果据此开发海量存储器,将在数据加密、解密以及 NP 等复杂性问题上取得突破性进展。

(5)开发各种用途的 DNA 芯片。针对计算机、半导体、光学合成等微加工技术以及复杂系统等领域开发特殊用途的 DNA 芯片必将引起研究者与应用者的极大兴趣。

(6)开发分子操作的高级语言。DNA 计算是一种新的计算方法,它以切割、粘贴、插入和删除等操作为基本运算,因此,开发以此基本运算为基础的分子操作的高级语言必将对 DNA 计算机的形成和推广使用起巨大作用。

参 考 文 献

1 Alderman L M. Molecular Computation of Solution to Combinatorial Problems[J]. Science, 1994, 266(11): 1021～1024

2 Richard L J. DNA Solution of Hard Computation Problems[J]. Science, 1995, 268(28): 542～545

3 Deaton R, et al. A DNA Based Implementation of an Evolutionary Search for Good Encoding for DNA Computation[A]. In: Proc. of 1997 IEEE Intl. Conf. on Evolutionary Computation, 1997, 13～16: 267～271

4 Manganaro G, et al. DNA Computing Based on Chaos[A]. In: Proc. of 1997 IEEE Intl. Conf. on Evolutionary Computation, 1997, 13～16: 255～260

5 Russo M F, et al. An Improved DNA Encoding Scheme for Neural Network Modeling [A]. 1994 Intl. Neural networks Society Annual meeting, 1994, 5～9: 354～359

6 Yoshikawa T, et al. Emergence of effective fuzzy rules for controlling mobile robots using DNA coding method[J]. In: Proc. 1996 IEEE Intl. Conf. on Evolutionary Computation, 1996. 581～586

7 Paun G, Rozenberg G, Saloman A. DNA Computing [M]. Springer, 1998. 10～41

8 邓少平,等. DNA 计算的一些基本问题. 科学(中文版),1996,(5): 51～54

9 Kari L. DNA Computing: Arrival of Biological Mathematics[J]. The Mathematical Intelligencer, 1997, 19(2): 9～22

10 马立人,等. 生物芯片[M]. 北京:化学工业出版社,2000

11 Adleman L M. On Constructing a Molecular Computing [R]: [Technical Report TR 79-387]. 1995

12 Boneh D, et al. On the Computational Power of DNA [R]: [Technical Report]. 1995

13 Quang Q, et al. DNA Solution of the Maximal Clique Problem[J]. Science, 1997, 278(17): 446～449

14 Adleman L M. Computing with DNA. Scientific American, 1998, 279(2): 54

15 Benenson Y, et al. Programmable and autonomous computing machine made of biomolecules[J]. Nature[J], 2001, 414(11): 430～434

16 Ravinderjit S, et al. Solution of a 20-Variable 3-SAT Problem on a DNA Computer[J]. Science, 2002, 296(4): 499～502

17 Liman W, et al. A DNA Computing readout operation based on structure-specific cleavage [J]. Nature Biotechnology, 2001, 19(11): 1053～1059

18 Dennis N. DNA-Based Computer Takes Aim at Genes [J]. Science, 2002, 295(2): 951

19 Adrian C. Hairpins Trigger an Automatic Solution[J]. Science, 2000, 288(5): 1152～1153

20 Leier A, Richter C, Banzhaf W, Rauhe H. Cryptography with DNA binary strands[J]. Biosystems, 2000, 57(1): 13～22

21 Bach E, et al. DNA models and algorithms for NP-compute problems[J]. Journal of Computer an System Sciences, 1998, 57(2): 172～186

22 Sakakibara Y. DNA computers: A new computing paradigm. Journal of Photopolymer Science and Technology[J], 1998, 11(4): 681～686

23 Yoshikawa T, et al. Effects of new mechanism of development from artificial DNA and discovery of fuzzy control rules[J]. In: Proc. 4th Intl. Conf. Soft Computing, 1996, 2: 498～501

24 Ogihara M, et al. Simulating Boolean circuits on a DNA computer. Algorithmica[J], 1999, 25(2-3): 239～250

25 Faulhammer D, et al. Counting DNA: Estimating the complexity of a test tube of DNA[J]. Biosystems, 1999, 52(1-3): 193～196

26 Garzon M H, et al. The bounded complexity of DNA computing [J]. Biosystems, 1999, 52(1-3): 63～72

27 Kari L. DNA computing in vitro and vivo[J]. Future Generation Computer Systems, 2001, 17(7): 823～834

28 Wasiewics P. DNA computing : Implementation of data flow logical operations [J]. Future Generation Computer System, 2001, 17(4): 361～378

29 Deaton R. DNA computing: A review [J]. Fundamenta Informaticase, 1998, 35(1～4): 301～307

30 Garzon M H. Bimolecular computing and programming [J]. IEEE Trans. On Evolutionary Computation, 1999, 3(3): 236～250

(上接第18页)

的算法,并且探索开发异构性即发挥异构性,以及屏蔽异构性即抑制异构性的场合和方法。

参 考 文 献

1 Ekmecic I, Tartalja I. A Survey of Heterogeneous Computing: Concepts and Systems. Proceeding of the IEEE, 1996, 84(8)

2 Freund R F, Siegel H J. Heterogeneous processing. Computer, 1993, 26(6): 18～27

3 曾国荪,陆鑫达. 异构计算中的负载共享. 软件学报,2000,11(4): 551～556

4 黄伟民,陆鑫达,曾国荪. 异构 BSP 模型及其通信协议. 电子学报,2000,28(8): 72～75

5 Zeng Guosun. Exploiting Maximum Parallelism in Loop Using Heterogeneous Computing. Chinese Journal of Electronics, 2001, 10(3): 340～344

6 Merrick I, Wood A M. Coordination with Scopes. In: Proc. ACM Symposium of Applied Computing, May 2000. 210～217

7 Zhang X, Yan Y. A Framework of Performance Prediction of Parallel Computing on Nondedicated Heterogeneous Networks of Workstations. In: Proc. of the 1995 Intl Conf. of Parallel Processing, Aug. 1995. 334～338

8 曾国荪. 自动提取程序的异构特征实现异构计算:[博士学位论文]. 上海交通大学计算机科学与工程系,2000

9 Zeng Guosun, Lu Xinda. Structured-Based Automatic Extraction of the Program Heterogeneity. In: Proc. of Conf. On Software: Theory and Practice, 16th World Computer Congress 2000, Aug. 2000. 985～986