一种基于 PEPA 流近似方法的动态自省模型

吕宏武，王慧强，马春光，林相君，赵 倩
（哈尔滨工程大学 计算机科学与技术学院，哈尔滨 150001）

摘 要：现有的基于自然语言或框图的自省模型已不能满足系统验证和分析的需求. 本文采用流近似方法将PEPA转化为常微分方程组(ODEs)，提出了一种基于性能评价进程代数(PEPA)语言的动态自省模型，且具有形式化推理验证和量化分析的功能，避免传统马尔可夫链求解方法状态空间爆炸的问题. 分析结果显示，缩减被管理组件检测过程的延迟时间、减小执行指令序列长度对于自省效率的提高具有重要作用.

关键词：自省；自律计算；性能评价进程代数；常微分方程
中图分类号：TP 319 **文献标志码**：A **文章编号**：0254－0037(2010)05－0710－06

自省机制对系统内部状态和环境改变进行感知和分析为系统结构演化和自动适应提供依据，是自律计算实现的基础[1]；Anthony 等[2]设计了一种运行时可配置软件结构，设计时添加动态决策点使软件具备对环境信息自省的能力. 马晓星等[3]提出了一种基于 edNC 图文法的自省模型；Wang 等[4]提出了一种基于 Pi 演算的自省模型；Strassner 等[5]通过 UML 框图定义了一个自律网络自省策略模型，提高了网络的环境感知和自管理能力. 不难看出，现阶段自省模型的描述方法普遍缺乏量化分析手段，难以满足系统分析需求，因此，迫切需要建立一种支持推理验证和量化分析的自省模型，以确定影响自省的关键性问题.

由于性能评价进程代数(performance evaluation process algebra，PEPA)在形式化验证之外，还具有性能分析的能力，本文将其作为形式化建模语言，此外，考虑到大型或超大型分布式系统(如 Internet，Bitcom 网络)状态空间巨大，通常 PEPA 求解方法可能面临空间爆炸问题[6]，利用 Hillston[7] 提出的流(fliud-flow)近似方法对 PEPA 状态空间向量化处理，形成常微分方程组(ODEs)，以支持大型状态空间分析.

本文利用 PEPA 语言对自省过程进行描述，提出了一种支持动态变更的自省模型；在此基础上，通过对模型参数效应的分析确定影响动态自省过程的关键因素.

1 PEPA 的流近似方法

PEPA 是一种典型进程代数[6]

$$P ::= (a,\lambda).P \mid P + P \mid P \underset{L}{\triangleright\!\triangleleft} P \mid P/L \mid A \tag{1}$$

式中，$(a,\lambda).P$ 为前缀操作，其含义是组件 $(a,\lambda).P$ 以变迁速率 λ 执行动作 a 后，转化为组件 P；$P+P$ 为选择操作，其含义是在组件 $P+Q$ 中，组件 P 和 Q 有且仅有一个被执行；$P \underset{L}{\triangleright\!\triangleleft} P$ 为合作操作，其含义是在组件 $P \underset{L}{\triangleright\!\triangleleft} Q$ 中，组件 P 和组件 Q 通过动作集合 L 进行合作，一般用来描述2个组件之间的同步；P/L 代表隐藏操作，其含义是动作集合 L 中的动作在组件 P 中可以被视为内部动作；A 代表常量，用于对组件的定义.

Hillston[7] 提出利用流近似的方法把 PEPA 转化成 ODEs，从而避免传统连续时间马尔科夫链(continuous-time Markov chain，CTMC)求解方法的状态空间爆炸；Bradley 等[8]和 Hayden 等[9]对其进一步

收稿日期：2009-12-10.
基金项目：国家自然科学重大计划基金资助项目(90718003)；国家自然科学基金资助项目(60973027，60373000)；国家“八六三”基金资助项目(2007AA01Z401).
作者简介：吕宏武(1983—)，男，山东日照人，博士生.

完善. 这对于异构、复杂的大型分布式系统而言，可以更为快速和高效地实现系统形式化建模与量化分析.

定义 1[7]　对于任意的 PEPA 模型 M，具有 n 种组件类型 $C_i(i=1,2,\cdots,n)$，每种组件都有 N_i 个不同的派生(derivation)，M 的数据矢量形式 $V(M)$ 是一个具有 $N=\sum_{i=1}^{n} N_i$ 个输入的向量，输入 v_{ij} 记录了当前状态下组件类型 C_i 第 j 个派生实例的个数.

在时刻 t，把 v_{ij} 记为 $N(C_{ij},t)$，变迁速率 $\rho_\alpha(C_{ij},P(t))$ 为系统 $P(t)$ 中当动作 α 发生时，C_{ij} 组件减少的速率，$\rho_\beta(C_{ij},P(t))$ 为系统 $P(t)$ 中当动作 β 发生时，组件 C_{ij} 增加的速率. 由于被动动作的存在，组件变迁速率不能完全由该组件自己决定，还涉及与之合作的相应组件. PEPA 模型可以转化为

$$\frac{\mathrm{d}v_{ij}(t)}{\mathrm{d}t}=-\sum_{k:\,C_{ij}\xrightarrow{(\alpha,\cdot)}C_{ik}}\rho_\alpha(C_{ij},P(t))+\sum_{k:\,C_{ik}\xrightarrow{(\beta,\cdot)}C_{ij}}\rho_\beta(C_{ik},P(t))\quad(j=1,2,\cdots,N_i,\ i=1,2,\cdots,n)\tag{2}$$

2　基于 PEPA 流近似的自省模型

自省实现结构可以抽象为元层(meta-leve1)和基层(base-leve1)2 个部分[3]. 基层包括系统组件和相关资源在内的实体部分，主要负责实现系统的功能，通常称为被管理单元(managed element，MDE)；元层在基层结构基础上的一种更高层次抽象操纵，可以反映基层的状态和结构，指导基层自适应.

元层可以简化为一个简单的自律反馈控制环 MAPE-K[4](monitor-analysis-plan-execute 和 knowledge). 在设计实现过程中，由于分析和计划 2 个部分紧密相关，通常被合并为 1 个部件，在本文中称其为决策器(Decide). 在基层中，MDE 可以看作是许多完成特定功能的服务器，每个都能提供特定的服务，相互之间通过自律反馈结构进行协作，而不直接交互(避免组件的内部合作行为，满足 PEPA 流近似建模条件). 从整体上把基层和元层看做完整的系统，可以对外提供各种服务，如图 1 所示.

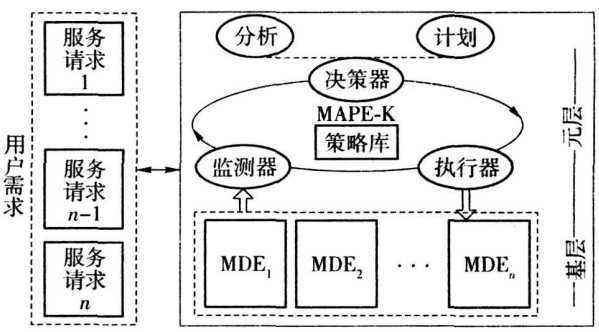


图 1　动态自省模型

Fig. 1　Dynamic self-reflection model

自省模型可以从 MAPE-K 环和 MDE 2 个主要部分对其进行描述.

MAPE-K 环是自省过程的施动者，按周期对 MDE 的状态信息分析与处理，并指导其自适应. 由于在 1 个自省周期内，MEPE-K 环发送完指令序列后，时钟周期还没有结束，必须等待下一个时钟周期开始时刻才能执行，因此，把 monitor 动作设置为被动变迁速率⊤. 当已有策略不能判定时，进行自学习进程 Learn，在执行进程 Execute 中，包括指导 MDE 执行的动作 execute 和剔除损坏 MDE 的动作 patch，其中被剔除的单元将会被重置或修复并由系统组成备用 MDE 序列来统一管理.

Monitor = (monitor，⊤). Decide；

Decide = (decided，r2). Execute + (uncertain，r3). Learn；

Learn = (leaning，r4). Decide；

Execute = (execute，r5). Monitor + (patch，r6). Monitor；

MDE 是对各种不同类型组件的一种抽象，MDE 执行、进入或退出都受到自律反馈控制结构的调度，可以在必要时关闭、开启某些服务或功能，以降低开销和减少系统的漏洞. MDE 按照所处状态的不同，可以划分为处于监测态(执行结束态)的组件 Element0、处于等待指令态的组件 Elenment1 和处于执行态的组件 Element2 以及处于被剔除态的组件 Patch 和处于备用态的组件 Reserve. 由于 MDE 被动地接受

MAPE-K 执行指令序列，因此，动作 execute 和 patch 的变迁速率为$\top$. 被剔除的 MDE，会以一定变迁速率被重置或修复从而转化为备用 MDE 而成为 Reserve 组件. 因为系统规模通常不易急速变化，处于备用态的组件将被逐渐加入到自省系统中. MDE 各派生的相互关系转化为

Element0 = (monitor, w1). Element1;

Element1 = (execute, $\top$). Element2 + (patch, $\top$). Patch;

Element2 = (work, w4). Element0

Patch = (reset, w5). Reserve;

Reserve = (reuse, w6). Element0

动态自省系统可以表述为

$$\mathrm{Monitor}[M] \underset{\{\mathrm{monitor},\ \mathrm{execute},\ \mathrm{patch}\}}{\triangleright\!\triangleleft} \mathrm{Element0}[N] \tag{3}$$

式中，M 为自省机制可同时处理 MDE 的容量；N 为初始时 MDE 的数量.

在 PEPA 模型的基础上，进行流近似转化. 对 PEPA 模型的状态进行数值向量化处理，状态映射为

Monitor$\rightarrow C_{11}$, Analysis$\rightarrow C_{12}$, Plan$\rightarrow C_{13}$, Execute$\rightarrow C_{14}$

Element0$\rightarrow C_{21}$, Element1$\rightarrow C_{22}$, Element2$\rightarrow C_{23}$Patch$\rightarrow C_{24}$, Reserve$\rightarrow C_{25}$

在时刻 t 模型可以向量化

$$P_1(t) = (N(C_{11}, t), N(C_{12}, t), N(C_{13}, t), N(C_{14}, t))$$

$$P_2(t) = (N(C_{21}, t), N(C_{22}, t), N(C_{23}, t), N(C_{24}, t), N(C_{25}, t))$$

$$P(t) = P_1(t) \triangleright\!\triangleleft P_2(t)$$

假定 MDE 的数量 N 巨大，根据 PEPA 流近似的方法，可以认为 MDE 的数量是连续变化的. 利用 Hayden 方法[9]消去被动动作，v_{ij}的导数均为连续的，方程组的解存在. 对应的 ODEs 为

$$\begin{cases}
\dfrac{\mathrm{d}v_{11}}{\mathrm{d}t} = -w_1 \min(v_{21}, v_{11}N) + r_5 \min(v_{14}, v_{22}N) + r_6 \min(v_{14}, r_{22}N) \\
\dfrac{\mathrm{d}v_{12}}{\mathrm{d}t} = -r_2 v_{12} - r_3 v_{12} + w_1 \min(v_{21}, v_{11}N) + r_4 v_{13} \\
\dfrac{\mathrm{d}v_{13}}{\mathrm{d}t} = -r_4 v_{13} + r_3 v_{12} \\
\dfrac{\mathrm{d}v_{14}}{\mathrm{d}t} = -r_5 \min(v_{14}, Nv_{22}) - r_6 \min(v_{14}, Nv_{22}) + r_2 v_{12} \\
\dfrac{\mathrm{d}v_{21}}{\mathrm{d}t} = -w_1 \min(v_{21}, v_{11}N) + w_4 v_{23} + w_6 v_{25} \\
\dfrac{\mathrm{d}v_{22}}{\mathrm{d}t} = -r_5 \min(v_{14}, v_{22}N) - r_6 \min(v_{14}, r_{22}N) + w_1 \min(v_{21}, v_{11}N) \\
\dfrac{\mathrm{d}v_{23}}{\mathrm{d}t} = -w_4 v_{23} + r_5 \min(v_{14}, v_{22}N) \\
\dfrac{\mathrm{d}v_{24}}{\mathrm{d}t} = -w_5 v_{24} + r_6 \min(v_{14}, r_{22}N) \\
\dfrac{\mathrm{d}v_{25}}{\mathrm{d}t} = -w_6 v_{25} + w_5 v_{24}
\end{cases} \tag{4}$$

3 模型参数效应分析

ODEs 的建立为自省能力的量化分析提供了基础，通过对自省模型 ODEs 的求解得到各状态的近似稳态概率.

假设 N_0(Monitor, t) 为 MAPE-K 结构在初始时刻可以用于对被管理单元处理的能力，即一个时间片内

可以同时接受处理的 MDE 个数，N_0(Element0，t) 为初始时刻 MAPE-K 控制的被管理单元数量，N_0(Reserve，t) 为初始时刻备用 MDE 的数量. 考虑一种最常见的情况

$$N_0(\text{Monitor},t)=N_0(\text{Element0},t)>0,N_0(\text{Reserve},t)=0,$$

代表一种规模受限的计算系统，例如多处理器自律系统，通常情况下，初始时刻每个处理器已经开始工作，不存在备用的处理器，但是当某个处理器发生故障时，可以通过自律反馈结构在无人为干预的情况下，移出故障处理器，对其进行修复并当作备用资源被再次利用.

由于模型对应的方程组(4)是常系数一阶齐次常微分方程，存在通解. 当模型各参数取值如表 1 时，求方程组的数值解，各组件数量随时间的变化曲线如图 2 所示.

表 1　模型参数取值

Table 1　Parameters of the model

参数	r_2	r_3	r_4	r_5	r_6	w_1	w_4	w_5	w_6
取值	0.7	0.3	1	0.9	0.1	1	1	0.2	0.2

由图 2 可见，一个自省周期内 MDE 处于等待指令态，即 Element1 形式的概率较大. 此时组件等待 MAPE-K 环发送指令而不执行，属于无效时间开销，应尽量减少 Element1 组件的数量. 对于确定的自律计算系统或自律单元，MAPE-K 环中检测和执行的变迁速率受环境或任务目标的影响通常是固定的，难以进行人为调节，因此，本文不再对参数 r_2 和 r_4 做过多讨论. 主要探讨改变 MDE 检测变迁速率 w_1 和执行变迁速率 w_4 对自省效率的影响，寻找减少 Element1 组件数量的方法，以提高系统效率.

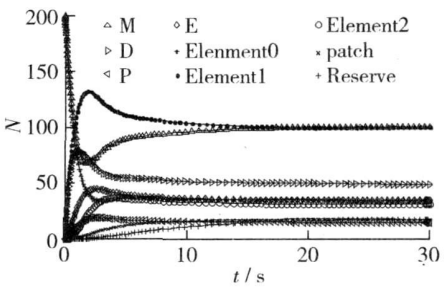
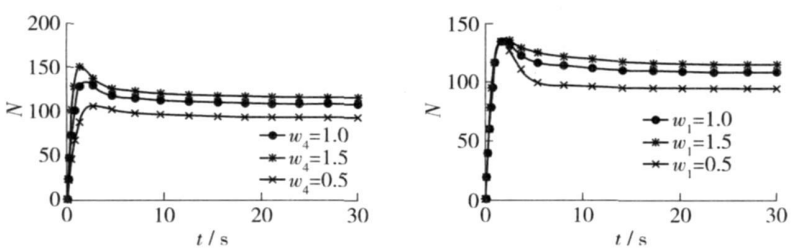


图 2　组件数量随时间序列的变化

Fig. 2　Time series plot of components in the model

图 3(a)是 w_1 分别为 0.5，1.0 和 1.5 时，处于等待执行态的 MDE 数量，可见，随着检测变迁速率的减小，其数量不断减少. 由于检测时间的缩短减少了自省检测过程的延迟时间，提高了 MAPE-K 的处理能力，因此 Element1 的数量减少.

图 3(b)是 w_4 分别为 0.5，1 和 1.5 时对 Element1 的影响，Element1 的数量随着动作执行变迁速率的减小而减少. 执行指令缩短，导致同一时间周期内 MAPE-K 用于生成指令的决策时延减少、MDE 处于等待指令态的概率减少，系统时间主要用于 MDE 执行指令. 由于指令序列的长度受限，不可能无限制缩短，因此，在保证自适应和自管理的目标下，应尽量减少指令长度.



(a) w_1 对 Element1 的影响　　(b) w_4 对 Element1 的影响

图 3　w_1 和 w_4 对 Element1 组件数量的影响

Fig. 3　The impacts of w_1 and w_4 on the population of Element1

当把 w_1 和 w_4 都缩短为 0.1 时，系统的近似稳态概率如图 4 所示，此时 Element1 组件的数量约等于总量的 10%，大多数的被管理组件是 Element2 或 Element0，表现为处于执行态或执行结束态等待新的自省周期开始.

在自省系统中，MDE 被剔除的变迁速率 r_6 代表组件损毁或业务淘汰的速率，对系统性能也有较大影响. 主要分析 r_6 对于 Element1 组件数量的影响. 当在其余参数不变的情况下，增大 r_6 将会使 Patch 组件数量有所增加，如图 5 所示. 随着移除组件数量的增多，组件重用概率不变，因此 MDE 处于剔除状态的概率变大.

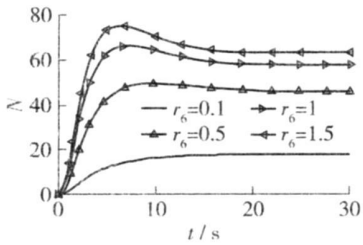


图 4 当 $w_1 = w_4 = 0.1$ 时各组件数量随时间序列的变化

Fig. 4 Time series plot of components in the model when $w_1 = w_4 = 0.1$

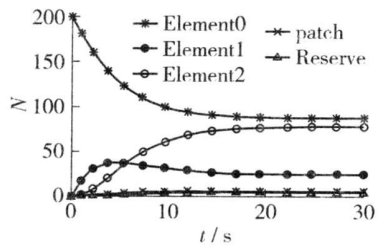


图 5 被管理组件剔除速率 r_6 对 Patch 组件数量的影响

Fig. 5 The impacts of r_6 on the population of Patch

PEPA 流近似求解极大地减少了自省模型分析过程的开销，表 2 列出了利用流近似方法和传统 CTMC 方法对本文提出的动态自省模型求解时间开销的对比，其中 CTMC 方法利用了 PEPA 工具 PEPA Eclipse-Plugin[10]，操作平台是 XP 操作系统，CUP 主频 3.2 GHz，内存 1 GB.

表 2 求解时间对比

Table 2 The comparison of solution time

M	N	状态数	CTMC 方法求解时间/s	ODEs 方法求解时间/s
5	5	2^{10}	0.828	0.53
10	10	2^{30}	312.8	0.53
20	20	2^{40}	—	0.53
100	100	2^{200}	—	0.53
1 000	1 000	$2^{2\,000}$	—	0.55

4 结束语

针对现有自省模型缺乏形式化描述尤其是性能分析的手段，提出了一种基于 PEPA 的动态自省模型，并利用流近似方法把 PEPA 模型转化为 ODEs，避免了大型和超大型分布式系统状态空间爆炸的问题，与传统的 CTMC 求解方法相比有效地降低了求解复杂性和开销. PEPA 的流近似解法，能直观、准确地刻画和反映自省过程，易于模型的求解. 且降低被管理组件检测过程的延迟时间、执行指令序列长度和提高自省效率.

参考文献：

[1] STERRITT R, PARASHAR M, TIANFIELD H, et al. A concise introduction to autonomic computing[J]. Advan-ced Engineering Informatics, 2005, 19(3): 181-187.

[2] ANTHONY R, PELC M, WARD P, et al. A Run-Time Configurable Software Architecture for Self-Managing Systems[C]// The 5th IEEE International Conference on Autonomic Computing. Chicago: IEEE Computer Society Press, 2008: 207-208.

[3] 马晓星，张小蕾，吕建. 自省的动态软件体系结构描述与实现[J]. 南京大学学报：自然科学版，2004，40(2)：146-155.

MA Xiao-xing, ZHANG Xiao-lei, LV Jian. Description and implementation of dynamic software architectures: a reflective

approach[J]. Journal of Nanjing University: Natural Science, 2004, 40(2): 146-155. (in Chinese)

[4] WANG Hui-qiang, LV Hong-wu, Feng Guang-sheng. A self-reflection model for autonomic computing systems based on π-calculus[C]// The Third Internatio-nal Conference on Network and System Security (NSS 2009). Gold Cost: IEEE Computer Society Press, 2009: 310-315.

[5] STRASSNER J, SAMUDRALA S, COX G, et al. The design of a new context-aware policy model for autonomic networking [C]// The 5th IEEE International Conference on Autonomic Computing . Chicago: IEEE Computer Society Press, 2008: 119-128.

[6] HILLSTON J. Tuning systems: From composition to performance[J]. Computer Journal, 2005, 48(4): 385-400.

[7] HILLSTON J. Fluid flow approximation of PEPA models[C]// Proceedings of the 2nd International Conference on Quantitative Evaluation of Systems, Torino: IEEE Computer Society Press, 2005: 33-42.

[8] BRADLEY J T, GILMORE S T, HILLSTON J. Analysing distributed Internet worm attacks using continuous state-space approximation of process algebra models[J]. Journal of Computer and System Sciences, 2008, 74: 1013-1032.

[9] HAYDEN R A, BRADLEY J T. Fluid semantics for passive stochastic process algebra cooperation [C] // International Conference on Performance Evaluation Methodologies and Tools, Athens: ACM Press, 2008: 1-10.

[10] PEPA Club. The PEPA Eclipse Plug-in[OL]. [2009-05-05]. http://www.dcs. ed. ac. uk/pepa/tools/

A Dynamic Self-reflection Model Based on Fluid Flow Approximation of PEPA

LÜ Hong-wu, WANG Hui-qiang, MA Chun-guang, LIN Xiang-jun, ZHAO Qian

(College of Computer and Science Technology, Harbin Engineering University, Harbin 150001, China)

Abstract: The existing self-reflection models based on natural language or diagram have not been able to meet the requirements of verification and analysis. The PEPA model is converted into Ordinary Differential Equations (ODEs). A dynamic self-reflection model based on Performance Evaluation Process Algebra (PEPA) which provides an insight into quantitative analysis besides reasoning is proposed to describe the process of self-reflection. Using a fluid flow approximation method to avoid a state-space explosion of Markov chains, a traditional solution method. The analysis results demonstrate that it is important to improve the efficiency of self-reflection by decreasing the latency time of monitor as well as shortening the length of instruction sequences.

Key words: self-reflection; autonomic computing; PEPA; ordinary differential equations

(责任编辑　张士瑛)