# 研究综述

流水线最早用于工业生产，其目的在于通过专业化分工、降低环节耦合度，一方面提高整体生产效率，令一方面便于评估生产流程各个环节，从而为后期改进工艺流程提供支撑。

现代计算机操作系统也引入流水线概念，将一个完整的程序执行流程分为取指、译码、取操作数、执行指令、保存结果等步骤，并分配给不同的执行单元完成。如今软件工程的项目开发广泛采用模块化开发方式，也为后期用户提交作业的流水线处理奠定基础。软件测试作为软件开发生命周期的重要环节，旨在尽快发现软件漏洞，并检验软件是否满足既定的需求和技术要求。

现代软件架构的复杂度日益加剧，一个重要的表现在于软件面向的业务流程中的逻辑爆炸，包括如下几个方面：

1. 某个模块在多个流程中被复用；
2. 程序流程按顺序进行，程序模块运行有复杂的先后关系；
3. 某个模块要同时聚合多个模块的输入才能进行下一步处理和输出；
4. 某个模块的输出要传递给多个后继模块；
5. 后继模块的结果重新返回前置模块做进一步处理。

庞大的模块集群、复杂的模块间关系要求架构师提前测试架构，软件性能工程（SPE）应运而生，用以确定架构对最终软件性能的影响程度，是否能够达到既定的目的，乃至做进一步架构优化。

对软件架构的性能测试大致分为三个阶段，第一阶段是制定仿真方案，做业务流水线测试，通过对大量样本流程仿真，获得原始仿真数据；第二阶段通过一系列合理的指标评价仿真结果；第三阶段将仿真结果与评价指标相结合，绘制成各类直观的图表，供研究者参考。本文以上述三阶段的内容为线索，对业界的研究成果进行综述。

## 业务流水的仿真

在大型分布式系统中，软件的性能问题有时不能通过增加更多服务器和硬件得到解决，大型软件架构通常不能与可用资源线性相关，往往存在性能瓶颈限制了附加硬件的作用。

在软件行业的性能测试常常被推迟到系统开发的末期，软件架构师需要尽早利用性能模型，以对架构预实施性能分析，从而避免软件后期迭代的成本剧增。

模型驱动的软件预测，作为一种在早期开发阶段优化软件架构的手段[[[1]](#endnote-0)]，用于评估响应时间、吞吐量、负载率是否满足需求，并能够分析不同架构模型设计和验证性能相关的要求。

在过去30多年，研究者已近提出许多的方法来创建更精确的性能评估模型，包括排队网络[[[2]](#endnote-1)][[[3]](#endnote-2)]、随机过程代数[[[4]](#endnote-3)]、随机性佩特里网络[[[5]](#endnote-4)]等最著名的预测模型，然而由于其过高复杂性，这些模型很少被从业者使用，因此研究者转向研究开发者友好的模型，以弥补这些性能分析模型在语义上的差剧，即在计算机自动提取的低层视觉特征和人所理解的高层语义特征间的差异。

过去10年中，超过20中分析方法以上述方向为基础，大多数使用UML注释性模型作为设计模型，创建分层排队模型作为分析模型[[[6]](#endnote-5)]，对象管理组（OMG）发布了多个UML概要，从而向UML模型添加与性能相关的注释，然而这些概要还在修订当中，不知道能否在在大范围中实践。

基于组件的软件工程（CBSE）给软件驱动的性能预测方法提供了新的思路，CBSE致力于组件重用性、灵活性、节约成本，并减少软件系统的上市时间[[[7]](#endnote-6)]。软件开发者可以测试组件从而提供更加细节、与性能有关的说明。

已经有几项研究方法解决了确定软件组件性能的挑战，这一任务的难点在于组件性能同时受到诸多环境因素的影响，且这些因素往往不能提前被开发者了解，这些因素包括：

1. 执行环境

组件部署的平台包括组件容器、应用程序服务器、虚拟机、操作系统、软件资源、硬件资源等。

1. 用量概况

用户对组件服务的输入、用户对组件请求数量概况。

1. 服务要求

附加的外部执行时间（由于网络时延等引起），最终与组件自身服务时间相加得到整体响应时间。

Sitaraman等人在没有考虑对请求服务的调用情况下，通过扩展O积分对组件性能建模[[[8]](#endnote-7)]；Grassi等人开发了中间模型语言KLAPER[[[9]](#endnote-8)]，用于基于组件的系统，从而弥补了不同设计和分析模型间的差剧；Palladio组件模型（PCM）[[[10]](#endnote-9)]遵循了前人的多项研究方法，试图反应所有组件性能的影响，PCM没有使用UML作为设计模型，而自定义其元模型，从而简化为性能预测的必要概念。

对于不涉及到并发问题的流水线业务，可以使用随机过程代数（SPA）方法，该方法可以用来快速预测响应时间，此外，基于仿真模型的排队网络，相对于SPA具有更少的限制条件，但通常花费更多的时间，下面将简要概述这两种经典性能仿真方式。

### 基于SPA的仿真

Capra是一种经典的SPA模型[[[11]](#endnote-10)]，涵盖了并发过程、操作系统资源、调度策略，并且能够将任意用时分布纳入分析过程，采用混合方式分析和仿真来进行性能预测。

一个进程的基本实体是动作，在Capra中，动作集合被分割成事件动作集合和需求动作集合，事件动作代表最小的、不可分的行为，主要用于进程间联络；需求动作则代表资源的用量，例如一段时间的处理器和硬盘的使用。在执行需求动作时，可能还会发生其它活动，因此用随机变量*Da.r*代表由需求行动*a*向请求资源*r*发出的请求，由概率密度函数*fa.r(t)*指代其时间分布特征。

为了创建更加复杂的行为结构，Capra使用不同操作符用于顺序的和替代性的组合以及用于递归过程行为的规范。

进程*P·Q*指代了包含*P、Q*进程的序列组合，即先执行*P*，在*P*成功终止后执行*Q。*

替代性组合用于建模仅包含*P、Q*进程的执行过程。Capra通过确定下一个行动的决策位置区分了两种替代性组合，如果进程自身决定下一个行动，则会使用内部的或者概率选择，进程*P*** *Q*以概率** 选择进程*P*，并以概率1-** 选择进程*Q*，此外，*P*+*Q*用于建模不同可能行为的外部或非确定性选择，其中的选择基于由进程环境所触发的事件行为，例如其它同时运行的进程。

进程变量*X*用于定义递归性质的进程，在现实系统中，递归调用的数量通常受限，并且这一界限通常不是固定的某个值，而是取决于系统的参数，为了近似这一行为，Capra将递归调用建模为离散的随机变量，用概率质量函数（pmf）表示，进程*P\*(Iter)*描述了进程*P*的有限循环，其中由pmf *P(Iter = n)*表征的随机变量*Iter*代表了执行迭代次数的概率。

进程*P* ||*A Q*代表进程*P*和*Q*的并行组合，进程间通过集合*A*中的事件动作通信，两个进程会竞争可用资源，因此可能导致延迟执行。

一个需求动作序列的总体需求是单个需求的总和，因此描述序列需求的随机变量*DP**Q*,*r*  *DP*,*r*  *DQ*,*r*，两个随机变量之和是它们概率密度函数的卷积，所以*DP**Q*,*r* 的pdf是*fP**Q*,*r (t)* (*fP*,*r*  *fQ*,*r)(t)*，其中代表卷积运算。对于*P*** *Q的*概率选择，其pdf D*P****Q=* *fp*,*r* (*t*)*+* (1-** )*fQ*,*r*(*t*)

qt_temp无限递归可以看作由进程*P、Q*组成的长度为n的时间序列组合，在其中进行概率选择，其概率选择定义为pmf *P*(*I = n*)。因此函数

C:/Users/15337/AppData/Local/Temp/qt_temp.a16324qt_temp计算需求的pdf。

根据以上资源需求的组合，所需的仿真步数显著减少，这本质上是一种基于Deson-J框架的离散事件仿真，并得出使用场景中的响应时间作为概率分布。

### 基于排队网络的仿真

许多性能分析方法使用排队网络作为底层预测模型，因为它们非常适用于分析并发系统的交互作用。排队模型包含一个中央子系统，用于处理进入网络队列的任务。

经典的排队论中涉及到的重要参数包括：

1. 平均到达率（）：每秒作业进入网络的数量
2. 平均服务率（）：服务台每秒可处理的作业数量
3. 平均响应时间（E(*T*)）：作业从进入网络到离开网络的时间间隔

在先到先服务（FCFS）的服务策略下，一个完整的作业过程可以表述为作业到达排队网络，若有空闲的服务台则会获得服务，否则进入等待队列，直到服务台空闲，则获得服务，并在作业结束后离开网络。

在基本的排队论假设下，所有的非抢占式策略（例如FCFS、LCFS、随机等）具有相同的平均响应时间时间，即所有的非抢占式服务策略在改善网络性能方面几乎是等价的，此外几项基本的运筹定理简化的排队网络的分析，包括Little定理，强制流定理和瓶颈定理，其中瓶颈定理指出了在作业数量较多的封闭式网络中，如果不改善系统中平均服务时间最长的服务器，则一切改善基本收效甚微。

## 业务流水的评价

为了确保软件模型的质量，开发者不仅要测试功能属性，同时要确保附加的功能需求能够在已被开发的模型中满足，这些附加需求包括：性能、可靠性、可用性、私密性、安全性、可维护性、便携性等，它们需要在项目开发早期阶段被确定，从而避免由于再设计和再开发导致的成本增长。

许多业务信息系统对软件性能都有高要求；基于网络的信息系统在日益激烈的网路业务竞争中，必须确保给用户提供低时延、快速响应的高品质服务，此外，公司内部使用的软件对于确保高效业务流程至关重要。

表2-1 排队网络的性能指标

|  |  |  |
| --- | --- | --- |
| **符号** | **命名** | **含义** |
|  | 平均响应时间 | 也称系统平均时间，代表作业从进入到离开队列的时间 |
|  | 平均等待时间 | 也称为平均延迟时间，代表作业在队里中等待，而“被浪费”时间 |
|  | 系统期望 | 在排队网络中（包括队列和服务器内）作业数的期望 |
|  | 队列期望 | 在队列中作业数的期望 |
|  | 设备i吞吐量 | 设备i在一段时间内完成的作业数的期望 |
|  | 设备i利用率 | 设备i在一段时间内忙碌时间占比的期望 |
| Slowdown= | 作业放缓 | 将响应时间与作业规模关联 |

在排队网络中，有几项指标非常重要（见表2-1），其中响应时间作为衡量用户体验的指标，被广泛用于WEB服务应用等的测试及其扩展研究[[[12]](#endnote-11)][[[13]](#endnote-12)][[[14]](#endnote-13)][[[15]](#endnote-14)]；吞吐量和利用率（负载率）作为衡量服务器运行效率的指标，也经常在服务器集群[15][[[16]](#endnote-15)][[[17]](#endnote-16)]中被检验。

slowdown[[[18]](#endnote-17)][[[19]](#endnote-18)][[[20]](#endnote-19)]是当下流行的度量标准，通过将响应时间*T*与作业规模*S*关联，可以保证小规模作业不被过度延后，从而均衡了服务器负载。

除了上述指标外，还有许多其它性能指标值得关注。包括时间可变性和slowdown的可变性[[[21]](#endnote-20)]，在系统设计时会比平均指标更重要。

另一个重要指标是响应时间的尾部行为，定义为响应时间超过某个等级的x的概率，即P{T>x}，了解尾部行为将有利于设置服务水平协议（SLA），但尾部行为通常较难导出[[[22]](#endnote-21)]。

## 三、业务流水的仿真结果展示

高并发的任务量，导致仿真后时间序列步数繁多，绘制成的图表不易布局，有可能产生两种完全相反的问题，第一种问题是数据繁杂导致难以辨识，另一种则是虽然可以辨识数据，但却导致图表过大，且大部分的空间处于空白状态。

针对上述问题，英特尔公司开发的流程图分析器（FGA）软件[[[23]](#endnote-22)]通过用户交互界面，将流程图和仿真运行后的数据组织起来，通过用户框选关注的数据区域可以实现区域重新绘制，并且鼠标悬停的数据会与流程图节点产生关联，从而直观的分析每时刻任务运行状态。

FGA内部主要使用了三类图表，第一种是每线程任务随时间执行轨迹（如图3.1），这类图表用于综合性性能分析和优化并行程序[[[24]](#endnote-23)]，其中横坐标代表时间变化，纵坐标代表任务所处的线程，此外，颜色由冷色（绿色）转向暖色（红色），代表一项任务耗时由1毫秒至1微秒的变化区间。

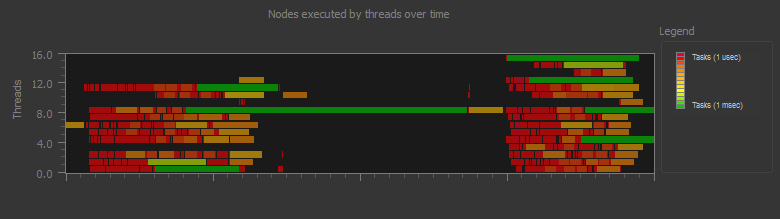


图3.1 每线程任务随时间执行轨迹

第二类图表是任务并发随时间变化直方图（图3.2），其中横坐标代表时间，纵坐标代表活跃的线程数量，而颜色区间则代表了每时刻运行节点述目的区间，通过这中组织方式，可以在图表中显示的数据维度也更加丰富，这类图表也广泛用于负载和性能分析[[[25]](#endnote-24)][[[26]](#endnote-25)]。

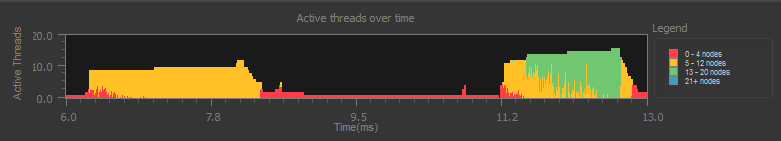


图3.2 任务并发随时间变化直方图

最后一类图表为图3.3中的第2部分窗口内展示的树状图，树状图是一种对整体进行细分的方法，在树状图中不同方块代表不同节点的用时，从而切分整个CPU时间，红色代表低并发量，绿色代表高并发量，通过这种简单的划分，可以更直观的找到那些耗时多而并发量低的瓶颈环节，被广泛用于各个领域的数据和性能分析[[[27]](#endnote-26)][[[28]](#endnote-27)]。

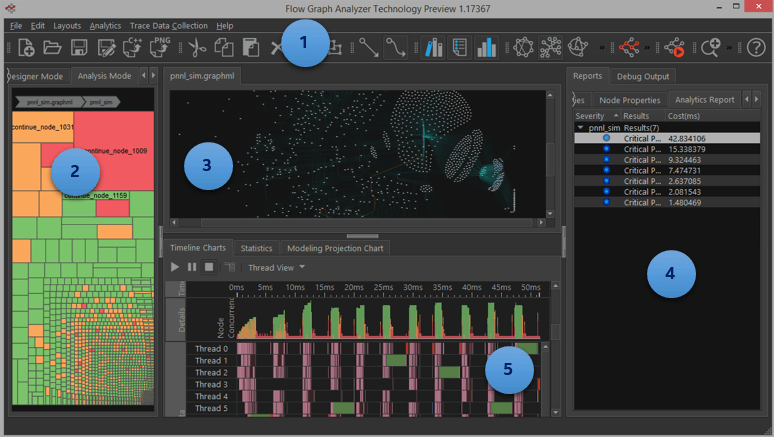


图3.3 FGA用户界面

通过树形图可以挖掘瓶颈节点，从而通过剩下的几个与时间相关的图表，进一步挖掘运行中的空闲时间，通过优化运行效率，从而消除空闲时间，或者通过调度可以并行执行并保持空闲内核繁忙的其他算法来解决。

此外对于长时间趋势的图，也可以使用考虑使用如图3.4所示的时间序列螺旋（Spiral Plot），该图从螺旋中心开始，然后向外发展，螺旋图功能多样，可以使用点或线条沿螺线路径方向显示时间序列，并且可以表征时间序列中存在的某些周期性特征[[[29]](#endnote-28)]。

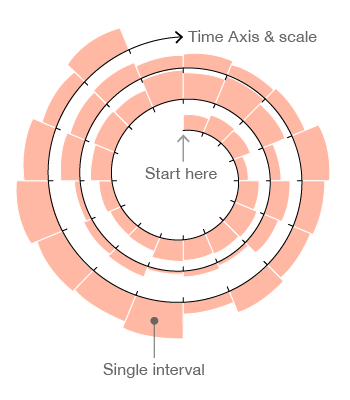


图3.4 时间序列螺旋图

## 参考文献

1. [] Smith, C. U. & Williams, L. G. (2001). Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley Professional [↑](#endnote-ref-0)
2. [] Bolch, G., Greiner, S., de Meer, H. & Trivedi, K. S. (2006). Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. Wiley-Interscience, 2nd Edition [↑](#endnote-ref-1)
3. [] Kounev, S. (2006). Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets. IEEE Transactions on Software Engineering, 32(7), 486-502. [↑](#endnote-ref-2)
4. [] Woodside, C. M., Neilson, J. E., Petriu, D. C. & Majumdar, S. (1995) The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software. IEEE Transactions on Computers,44(1), 20-34. [↑](#endnote-ref-3)
5. [] Kounev, S. (2006). Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets. IEEE Transactions on Software Engineering, 32(7), 486-502. [↑](#endnote-ref-4)
6. [] Balsamo, S. , DiMarco, A., Inverardi, P. & Simeoni, M. (2004). Model-Based Performance Prediction in Software Development: A Survey. IEEE Transactions on Software Engineering, 30(5), 295-310. [↑](#endnote-ref-5)
7. [] Szyperski, C., Gruntz, D. & Murer, S. (2002). Component Software: Beyond Object-Oriented Programming. Addison-Wesley [↑](#endnote-ref-6)
8. [] Sitaraman, M., Kuczycki, G., Krone, J., Ogden, W.F. & Reddy, A. (2001). Performance Specifications of Software Components. In Proceedings of the Symposium on Software Reusability 2001 (pp. 3-10). [↑](#endnote-ref-7)
9. [] Grassi, V., Mirandola, R. & Sabetta, A. (2006). Filling the gap between design and performance/reliability models of component-based systems: A modeldriven approach. Journalof Systems and Software, 80(4), 528-558. [↑](#endnote-ref-8)
10. [] Becker, S., Koziolek, H. & Reussner, R. (2007). Model-based Performance Prediction with the Palladio Component Model. In Proceedings of the 6th Workshop on Software and Performance WOSP’07 (pp. 56-67). ACM Press [↑](#endnote-ref-9)
11. [] Happe, J., Koziolek, H., & Reussner, R. H. (2007). Parametric Performance Contracts for Software Components with Concurrent Behaviour. In Electronical Notes of Theoretical Computer Science, Vol. 182 (pp. 91-106 ), Elsevier. [↑](#endnote-ref-10)
12. []刘承启, 林振荣, 黄文海. 基于LSTM的WEB服务响应时间大数据预测方法[J]. 四川大学学报(自然科学版), 2019, 056(001):71-77. [↑](#endnote-ref-11)
13. []郑晓霞,赵俊峰,程志文,谢冰.一种WebService响应时间的动态预测方法[J].小型微型计算机系统,2011,32(08):1570-1574. [↑](#endnote-ref-12)
14. []陈文文. 基于BP神经网络的云服务响应时间预测方法研究[D].东北大学,2013. [↑](#endnote-ref-13)
15. []王红斌. Web服务器集群系统的自适应负载均衡调度策略研究[D].吉林大学,2013. [↑](#endnote-ref-14)
16. []刘梅, 高岑, 田月,等. 基于Docker Swarm集群的调度策略优化算法[J]. 计算机系统应用, 2018. [↑](#endnote-ref-15)
17. []马晓光,刘钊远.一种适用于Docker Swarm集群的调度策略和算法[J].计算机应用与软件,2017,34(05):283-287. [↑](#endnote-ref-16)
18. [] Bansal N, Harchol-Balter M. Analysis of SRPT scheduling: Investigating unfairness.Proceedings of the 2001 ACM SIGMETRICS international conference on measurement and modeling of computer systems. 2001. p. 279–90. [↑](#endnote-ref-17)
19. [] Casanova H, Desprez F, Suter F. On cluster resource allocation for multiple parallel task graphs. J Parallel Distrib Comput 2010;70(12):1193–203. [↑](#endnote-ref-18)
20. [] Harchol-Balter M. Performance modeling and design of computer systems:queueing theory in action. Cambridge University Press; 2013. [↑](#endnote-ref-19)
21. [] Maccio, V. J., Hogg, J., & Down, D. G. (2018). On slowdown variance as a measure of fairness.Operations Research Perspectives, 5, 133–144. doi:10.1016/j.orp.2018.05.001 [↑](#endnote-ref-20)
22. [] O.J. Boxma and B. Zwart. Tails in scheduling Performance Evaluation Review, 34(4), 13-20.2007. [↑](#endnote-ref-21)
23. [] Tovinkere, Vasanth, and Michael Voss. "Flow graph designer: a tool for designing and analyzing Intel® threading building blocks flow graphs." 2014 43rd International Conference on Parallel Processing Workshops. IEEE, 2014. [↑](#endnote-ref-22)
24. [] Adhianto, Laksono, et al. "HPCToolkit: Tools for performance analysis of optimized parallel programs." Concurrency and Computation: Practice and Experience 22.6 (2010): 685-701 [↑](#endnote-ref-23)
25. [] Hernández-Orallo, Enrique, and Joan Vila-Carbó. "Web server performance analysis using histogram workload models." Computer Networks 53.15 (2009): 2727-2739 [↑](#endnote-ref-24)
26. [] Suleman, M. Aater, Moinuddin K. Qureshi, and Yale N. Patt. "Feedback-driven threading: power-efficient and high-performance execution of multi-threaded workloads on CMPs." ACM Sigplan Notices 43.3 (2008): 277-286. [↑](#endnote-ref-25)
27. [] Walker, Richard, and Jack Basiago. "System and method for tree map visualization for database performance data." U.S. Patent Application No. 10/375,393. [↑](#endnote-ref-26)
28. [] Scheibel, Willy, et al. "A Taxonomy of Treemap Visualization Techniques." VISIGRAPP (3: IVAPP). 2020. [↑](#endnote-ref-27)
29. [] Weber, Marc, Marc Alexa, and Wolfgang Müller. "Visualizing time-series on spirals." Infovis. Vol. 1. 2001. [↑](#endnote-ref-28)