Computer Science and Application 计算机科学与应用, 2020, 10(10), 1753-1764
Published Online October 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.1010186

Hans汉斯

基于排队论的在线交易处理服务器配置决策方案

张　乐[1*]，连增申[2]，曾国荪[1#]，丁春玲[3]

[1]同济大学计算机科学与技术系，上海
[2]北京捷软世纪信息技术有限公司，北京
[3]同济大学化学科学与工程学院，上海
Email: #zhangle_tj@163.com

收稿日期：2020年10月5日；录用日期：2020年10月20日；发布日期：2020年10月27日

摘　要

电子商务频繁，在线交易量指数式增长，要求网络处理平台不断扩容升级。针对在线交易中心，服务器配置不准确，适应性差的问题，本文研究一种基于排队论的服务器配置方法。首先，采用G/M/c的排队论模型对交易请求的处理过程进行仿真建模，给出交易请求响应时间的估算公式，由此设计交易服务速度的计算方法。然后，通过实验建立服务器服务速度与服务器物理配置之间的对应关系，给出服务器物理配置的决策方案。实际应用表明，本文方法相可满足各种在线交易的服务器集群配置需求。

关键词

在线交易，服务器扩容，排队论，硬件性能配置

Online Transaction Processing Server Configuration Decision Scheme Based on Queuing Theory

Le Zhang[1*], Zengshen Lian[2], Guosun Zeng[1#], Chunling Ding[3]

[1]Department of Computer Science and Technology, Tongji University, Shanghai
[2]Beijing Agile Century Information Technology Co., Ltd., Beijing
[3]College of Chemical Science and Engineering, Tongji University, Shanghai
Email: #zhangle_tj@163.com

Received: Oct. 5th, 2020; accepted: Oct. 20th, 2020; published: Oct. 27th, 2020

*第一作者。
#通讯作者。

文章引用: 张乐, 连增申, 曾国荪, 丁春玲. 基于排队论的在线交易处理服务器配置决策方案[J]. 计算机科学与应用, 2020, 10(10): 1753-1764. DOI: 10.12677/csa.2020.1010186

Abstract

Frequent e-commerce and exponential growth in online transaction volume require continuous expansion and upgrade of network processing platforms. Aiming at the problem of inaccurate server configuration and poor adaptability in online trading centers, this paper studies a server configuration method based on queuing theory. First, the G/M/c queuing theory model is used to simulate the processing process of the transaction request, and the estimating formula of the transaction request response time is given, thereby designing the calculation method of the transaction service speed. Then, the corresponding relationship between server service speed and server physical configuration is established through experiments, and a decision-making plan for server physical configuration is given. Practical application shows that the method in this paper can meet the server cluster configuration requirements of online transactions in various Chinese companies.

Keywords

Online Transaction, Server Expansion, Queuing Theory, Hardware Performance Configuration

1. 引言

移动互联网的普及，使得人们可以通过无处不在的互联网进行金融业务，大大地增加了参与在线交易的用户基数。另一方面，电商也在不断地推出各类活动，例如双十一、六一八大促等，进一步地提升在线交易的规模。然而瞬时猛增的交易量，不可避免给现有部署的服务器更大的处理压力，从而产生宕机，卡顿，延迟巨大这些问题。最近，我们受北京捷软公司委托，开展项目“金融大数据实时流式计算及存储架构”研究，旨在针对金融领域产生的大数据进行实时的流式计算，对其进行高效存储和管理，提出适应于金融大数据的计算模型和存储架构，构建实时流式计算平台。然而在构建交易平台的时候，旧有的服务器在性能和数量上难以承受交易请求到达数量增加带来的压力。为了解决这些问题，对服务器配置的改造和升级就被提上了日程。

服务器扩容和改造升级一直是计算机应用领域的一个研究热点。早期的服务器配置估算主要采用TPC、SPEC 和 HPCC 等的服务器评测体系的方式。例如刘文清[1]应用 TPC 评测体系给出数据库服务器和应用服务器配置的估算方案。陈英达[2]利用 TPC 和 SPEC 两类评测体系对服务器配置的估算方式开展了详细的研究，并给出相应的估算方法。然而 TPC、SPEC 和 HPCC 这些方法忽视了交易请求存在多种到达规律，也无法描述交易系统处理交易请求这一动态的过程，无法保证服务器配置计算的准确度。为了克服上述缺点，有人利用排队论建立交易请求处理过程的仿真模型。Derek [3]等人将单台服务器系统视为 M/G/1/N 排队模型，用于发现适用于各种服务时间分布的最佳配置，通过实验可得，该模型可以更好地提供如何投资于增加服务器速度的策略。Young [4]等人将单台服务器系统提供的服务建模为 M/M/1 排队模型，求出服务请求的平均响应时间的公式，实验表明，该方法有效地提高了服务器配置策略的准确率。然而 Derek 和 Young 仅仅是对单台服务器系统的情况进行探究，缺乏对多台服务器系统情况的分

析。为了克服以上缺点，Hamzeh [5]等人将云服务器建模为 M/G/c/c+r 的排队系统，基于该模型他们求得请求响应时间和其他重要的绩效指标，用于运营商确定服务器数量，将单台服务器系统的排队论建模推广到多台服务器系统，在进行数值实验和仿真验证之后，可以表明，该文献所提出的近似方法能为系统中的平均任务数，阻塞概率，立即服务概率以及响应时间分布特征(例如均值，标准差，偏度)提供高精度的结果。Junwei 等人[6]将多服务器系统视为 M/M/c 排队模型，在此基础上得到新到达的服务请求的等待时间的概率密度函数，提高响应时间计算的准确度。YJ [7]等人将有限容量的云服务器系统建模为 M/M/c/K 排队系统，论证了系统容量控制和利用率对等待时间，丢失概率和最终到达率的各种性能的影响，实验表明，相比传统方法，该文献提出的最佳利润控制策略可以使得云提供商在服务器数量上做出更为准确的决策。Jing 等人[8]将多服务器系统视为 M/M/c 排队模型，在考虑到市场需求，请求的工作量，服务器级协议，服务器的租赁成本，能耗成本等诸多因素的基础上，提出了利润最大化的最优配置策略，实验表明，该配置策略优于传统方法。廖倩文[9]将多服务器的云计算中心看作为一个 M/M/c 排队问题，利用泊松分布描述批量到达云计算中心的用户服务请求变化特点，建立基于排队论的批量到达系统模型，然后给出系统稳态运行指标的计算方法，给出云中心的服务器配置的参考意见。张江强[10]在该文献的基础上，建立了具有两类用户请求的排队模型，分析系统的稳态概率分布、平均队长和平均响应时间等性能指标，提出云计算中心的服务器数量的优化方案。

虽然以上文献都利用排队论在一定程度上提升服务器配置估算的准确度，但都是将交易请求的到达规律简单地假设为泊松分布，适应性较差，与日趋复杂的在线交易环境不符，也没有探究服务速度与服务器物理配置之间的对应关系，无法形成服务器配置估算的标准流程。

因此，本文通过分析在线交易请求特点，提炼交易系统设计过程中存在的问题，首先采用 G/M/c 的排队论模型对交易系统进行仿真建模，给出交易系统性能计算公式，然后通过实验得到服务器服务能力与服务器硬件配置之间的对应关系，最终给出服务器配置方案。经实验可知，该方法可以适应多类实际交易系统设计问题的需要。

2. 在线交易过程和问题提出

2.1. 在线交易系统框架

图 1 为在线交易处理系统框架，用户向交易系统发出交易请求，在交易系统内部排队，继而形成任务队列，交易请求在被交易系统处理后，存储于数据库服务器当中。这里将从单机架构和集群架构进行分析。当 $c = 1$ 时，该系统为单机架构，当 $c > 1$ 时，该系统为集群架构。单机架构，顾名思义，交易请求的整个处理过程都是运行在一台服务器上。在为单机架构的交易系统配置服务器时，只需要考虑服务速度这一要素。集群架构，与单机架构相对应，同时有多台服务器处理交易请求。因此，在为集群架构的交易系统配置服务器时，既需要考虑服务器速度，还需要考虑服务器的数量。为了突出重点，我们做出了以下假设：1) 单机架构的交易系统，其处理效率由服务器的处理速度决定；2) 集群架构的交易系统，其处理效率由服务器的处理速度和个数决定。

2.2. 交易系统设计中存在的问题

在线交易系统的设计过程中，存在着以下几个难点。1) 高峰时期交易请求规模大：随着互联网+的推广，在线交易业务猛增，在线交易存在短时间高并发的现象。例如淘宝的双十一购物节，12306 的春运放票等等。以天猫 2019 年的双 11 为例，活动当天，订单创建峰值达到 54.4 万笔/秒。体现在 0 点这一时刻，购买商品的用户同时向淘宝平台发出交易请求，海量的交易请求蜂拥而至，呈现爆发的态势，给在线交易平台产生了巨大的压力。因此，在为交易系统配置服务器的时候，首先需要考虑的便是其高峰

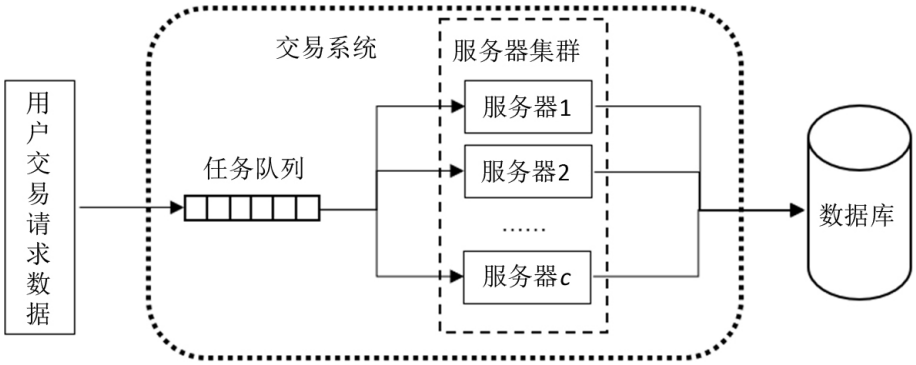

Figure 1. Online transaction processing system framework
图 1. 在线交易处理系统框架

时期的承受能力。2) 高实时性：为了保证发起在线交易请求的用户的良好的体验，在线交易请求的处理完成时间必然是越短越好。如果交易系统处理交易请求的时间超过 3 秒钟，停留在支付界面的用户会有明显的延迟感。因此，在线交易请求的处理是一项具有极高实时性的业务。3) 交易数据到达随机没有规律：发送交易请求的用户彼此之间是独立的个体，因此促使不同用户提出在线交易请求的因素也是多种多样。在设计交易系统的过程中，非常有必要找出高峰时期交易请求的到达规律。4) 交易请求处理规则繁多：金融在线交易的形式越来越多，支付手段也极为丰富，给用户带来了极大的便利，但同时也存在着各种交易风险，如用户信息泄漏、交易信息被篡改等，给用户的利益带来严重的损害。本项目在对交易请求的特点进行分析之后，为每一类交易请求都制定了 10 个以上的风险规则。同时，这么多的风险规则这也增加了交易请求的处理复杂度。5) 处理服务器的选型和配置困难：本文是围绕着交易系统的服务器配置展开研究的，因此，处理服务器的选型便是重中之重。

2.3. 研究动机

本文的研究基于“金融大数据实时流式计算及存储架构研究”这一项目，旨在针对金融领域产生的大数据进行实时的流式计算，并对其进行高效存储和管理，提出适应于金融大数据的计算模型和存储架构。该项目的具体目标是建立基于 Cassandra 的流大数据数据库，构建实时流式计算平台，支付机构总量 1 亿个人客户以上，确保每日高峰时期每秒并发 10,000 笔交易持续两小时，满足系统的平均响应时间 w 小于 2 秒，连续运行三十日以上。同时还要根据 10 个以上的交易规则，判断风险分数，即时提交给交易系统，进行交易风险事件判断，实现风险事故比率在万分之一以下，显著提升在线交易质量。然而在构建交易平台的时候，旧有的服务器在性能和数量上满足难以承受交易请求到达数量增加带来的压力。为了达成这些目标，我们有必要对服务器配置的扩容和改造升级展开研究。

3. 交易系统仿真建模

3.1. 在线交易过程的排队论假设

在交易系统中，用户请求源源不断的到来，并被 c 台服务器处理，这一过程可以视为一个排队过程。本文研究的交易系统内的 c 台服务器是同构的。我们利用排队论的相关理论工具来研究在线交易过程[11]，特别关注以下几个变量参数：

c，服务窗口个数，即服务器个数；

λ，交易请求到达速度，即单位时间内交易系统平均到达的交易请求个数；

μ，服务速度，即单位时间内一个服务器能够处理完成交易请求的平均个数；

w，交易请求在系统中平均逗留时间，即交易请求响应时间。

3.2. 在线交易请求到达规律建模

在非高峰时期，用户的行为不容易受外界影响，故可以认为每一条交易请求到达交易系统的事件是随机并且独立同分布的，我们可以假设单位时间内到达系统的交易请求个数服从泊松分布。在高峰时期，用户容易受到外界事件的影响，整个用户群体的趋向性较为明显，表现为集中地在同时在一个较短的时间段内提出交易请求。由于本文的研究目的是因为业务大增，提出服务器扩容升级的方案，所以我们聚焦研究高峰时期在线交易的规律。相较于非高峰时期，高峰时期的交易请求到达规律存在以下特点：1) 数据到达的规模较大；2) 数据到达的规模是可预见的，在某一个相对较小的范围内波动；假设泊松分布的参数为λ，实际的物理含义则是单位时间内到达系统的交易请求的平均个数，其期望和方差均为λ，显而易见的是，它无法满足对数据离散程度有更进一步要求的场景。又因随着λ值的增大，泊松分布可近似为正态分布，而正态分布优点在于我们可以通过调整描述离散程度的参数σ来拟合数据分布的离散程度。因此在本文中，我们假定高峰时期的交易请求到达规律符合正态分布。我们定义$X(y)$为$(y', y'+y)$时间内到达交易系统的交易请求个数，它服从参数为均值为λy，标准差为σy的正态分布，其累积分布函数为：

$$P\left(X(y)\le k\right)=\begin{cases}0 & (k<0)\\ \int_{-\infty}^{k}\dfrac{1}{\sqrt{2\pi}\sigma y}\mathrm{e}^{\left(-\frac{(x-\lambda y)^2}{2(\sigma y)^2}\right)}\mathrm{d}x & (k\ge 0)\end{cases} \tag{1}$$

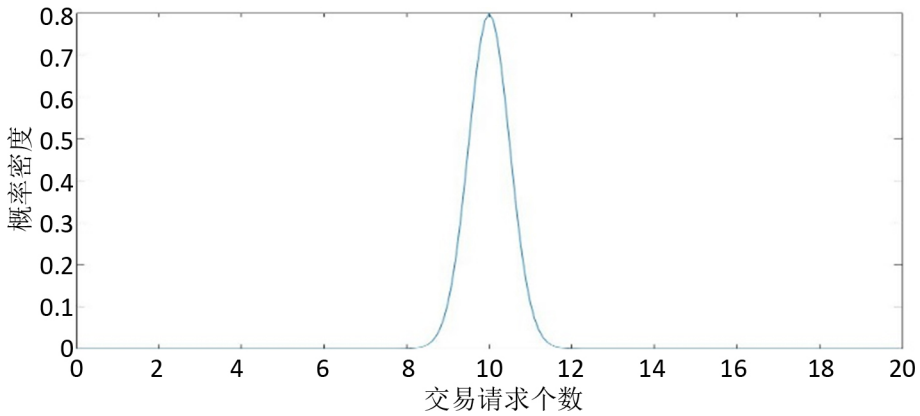

Figure 2. Normal distribution of the number of transaction requests
图 2. 交易请求个数的正态分布

图 2 为交易请求个数的在高峰时期的分布，其中$\lambda=10$，即交易请求到达速度为 10，$\sigma=0.5$，即正态分布的标准差为 0.5，$y=1$，图中只呈现了$0\le X(y)\le 20$时的概率分布。从图 2 可以看到，高峰时期，当交易请求个数服从“很陡”的分布时，正态分布可以很好地描述这类分布情况。

3.3. 交易处理的服务规律建模

一般而言，不妨假设每一个服务器的执行时间Y_1，服从参数为μ的负指数分布，意为一个服务器在单位时间内处理的平均交易请求个数为μ，即 3.1 节所提到的服务速度为μ，其累积分布函数如下：

$$P\left(Y_1\le y\right)=\begin{cases}1-\mathrm{e}^{-\mu y} & (y\ge 0)\\ 0 & (y<0)\end{cases} \tag{2}$$

3.4. 排队性能指标的计算

在 3.1 和 3.2 节中，假设高峰时期，交易请求的到达分布服从正态分布。在 3.3 节我们假设服务时间服从负指数分布。因此，整个交易系统的服务过程可以认为是 G/M/c 排队系统[11]，即交易请求到达分布服从平均到达速度为 λ 的正态分布，其相应的间隔时间分布函数为 $A(y)$，服务时间服从参数为 μ 的负指数分布。其中，λ 为单位时间内，交易系统到达的交易请求平均个数，μ 为单位时间内，单个服务器所能处理的交易请求平均个数，即 3.1 节所提的服务速度，c 为交易系统的服务器个数，σ 为标准差。

根据公式(1)，可以求得高峰时期交易请求到达间隔时间 Y_2 的累积分布函数 $A(y)$ 为：

$$\begin{aligned} A(y) &= P(Y_2 \le y) = 1 - P(Y_2 > y) = 1 - P(X(y) = 0) \\ &= 1 - \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}\sigma y} \mathrm{e}^{\left(-\frac{(x-\lambda y)^2}{2(\sigma y)^2}\right)} \mathrm{d}x \ (y > 0) \\ &= \int_{0}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma y} \mathrm{e}^{\left(-\frac{(x-\lambda y)^2}{2(\sigma y)^2}\right)} \mathrm{d}x \ (y > 0) \end{aligned} \tag{3}$$

受文献[12]的启发，可得 G/M/c 系统的交易请求响应时间 w 的计算公式可近似为：

$$w = \frac{\alpha^c}{c\mu(1-\alpha)} + \frac{1}{\mu} \tag{4}$$

其中，α 为如下函数 $f(\alpha,\mu,\lambda,c)=0$ 时的解，并且 α 在区间(0,1)，$f(\alpha,\mu,\lambda,c)=0$ 存在唯一解，函数 f 的解析公式如下：

$$f(\alpha,\mu,\lambda,c) = -\alpha + \int_0^{+\infty} \mathrm{e}^{-(c\mu - c\mu\alpha)^y} A'(y)\mathrm{d}y = \int_0^1 \int_0^1 \left[-\alpha + \frac{x^2 - x\lambda y - \sigma^2 y^2}{\sqrt{2\pi}\sigma^3 y^4} \mathrm{e}^{-\left(\frac{(x-\lambda y)^2}{2(\sigma y)^2} + (c\mu - c\mu\alpha)^y\right)} \right] \mathrm{d}x\mathrm{d}y \tag{5}$$

4. 在线交易服务速度的计算

4.1. 服务速度估算的基本思想

在第 3 节中，我们将交易请求到达速度、交易请求响应时间、服务器个数和服务速度这四者统一到公式(4)中。然而本文的研究动机是服务器配置的决策，实现这一目标的先决条件就是要求出服务速度的值，因为无法通过简单的变换和解析方法，直接将公式(4)中的服务速度 μ 计算出来。为此，我们通过反复迭代的数值计算方法，以便近似估算服务速度 μ。

数值计算方法的本质是将可能的备选值不断代入目标计算公式，迭代逼近目标解。一般地，不断迭代不断修正，有限次迭代之后，可以得到解的近似值。针对公式(4)，我们可以发现，响应时间 w 和服务速度 μ 两者存在着负相关的关系，即随着 μ 的单调递增，w 会跟着单调递减。遵循这个思想，合理控制 μ 的递增步长，在已知响应时间 w 和到达速度，以及保证求解精度的前提下，我们可以得出服务响应速度的估算值。具体地，在第一次迭代的时候，对 μ 设置一个初值，即一个比较小的值，不妨令 $\mu=\lambda$，代入到公式(4)中，计算出其对应的响应时间；如果不符合精度要求，则将 μ 增加一个步长 $\Delta\tau$，$\Delta\tau$ 是一个很小的值，确保不会引起计算震荡和发散；然后进入下一轮的迭代计算。

在每一次的迭代过程中，需要利用中间变量 α，因此需要先求出中间变量 α。α 的计算过程较为复杂，下面阐述 α 的求解过程。

正如公式(5)，由于 α 为函数 $f(\alpha,\mu,\lambda,c)=0$ 的解，并且在区间(0,1)存在唯一解，而 α 是变量 μ、λ 和

c 的隐函数，并且公式(5)中包含二重积分，我们采用数值积分的方法来近似求出积分值。一般地，常用的数值积分方法有两类，分别为基于梯形法和基于拉格朗日插值法的数值积分方法，两者计算复杂度相近，而后者在精度上高于前者。因此，本文将采用基于拉格朗日插值法的数值积分方法来计算函数 $f(\alpha,\mu,\lambda,c)$，其主要思想是用拉格朗日插值多项式代替被积函数，通过级数求和近似代替定积分。参考和引用文献[13]中方法，可以得到下面公式(6)。

$$f(\alpha,\mu,\lambda,c)\approx\sum_{j=0}^{n-1}\sum_{i=0}^{n-1}\left[-\alpha+\frac{x_i^2-x_i\lambda y_j-\sigma^2y_j^2}{\sqrt{2\pi}\sigma^3y_j^4}\mathrm{e}^{-\left(\frac{(x_i-\lambda y_j)^2}{2(\sigma y_j)^2}+(c\mu-c\mu\alpha)^{y_j}\right)}\right]a_ib_j \tag{6}$$

不妨取 $n\times n$ 个插值点均匀分布在(0,0)到(1,1)的矩形积分平面上，由此令 x_i，$i,j=0,1,\cdots,n-1$，为均匀分布在横坐标(0,1)上的 n 个点，有 $x_i=(2i+1)/2n$。同理，令 y_j 为均匀分布在纵坐标(0,1)上的 n 个点，有 $y_j=(2j+1)/2n$。x_i 和 y_j 组合即可构成 $n\times n$ 个插值点。由公式(6)可知，$f(\alpha,\mu,\lambda,c)$ 近似于其积分平面内 $n\times n$ 个插值点的被积函数值乘以求积系数 a_i 和 b_j 的和，其中系数 a_i 和 b_j 为最高次数为 $n+1$ 的多项式，其形式如下所示。

$$a_i=\int_0^1\prod_{\substack{k=0\\k\neq i}}^{n-1}\frac{x-x_k}{x_i-x_k}\mathrm{d}x \tag{7}$$

$$b_j=\int_0^1\prod_{\substack{k=0\\k\neq j}}^{n-1}\frac{y-y_k}{y_j-y_k}\mathrm{d}y \tag{8}$$

4.2. 服务速度的数值计算算法

根据 4.1 节已经论述服务速度的数值计算的一般过程和基本方法，在线交易服务速度反复迭代的数值计算算法的伪代码如下：

算法 1：服务速度计算算法 Service Speed Compute，简写为 SSC

输入：交易请求平均响应时间 w 秒，交易请求平均到达速度 λ 个交易请求/秒，服务器个数 c，收敛精度控制参数 ϵ，服务速度自增步长 $\Delta\tau$，坐标轴的插值节点个数 n，计算期间参数自增步长 $\Delta\xi$，交易请求到达离散程度 σ；

输出：单个服务器的服务速度 μ，单位为“交易请求数/秒”

SSC()

{ $\mu\leftarrow\lambda$， $t\leftarrow 2w$;//服务速度、平均响应时间初始化

while $t>w$//不满足平均响应时间要求，循环迭代计算 μ 值

{ $f\leftarrow 2\epsilon,\alpha\leftarrow\Delta\xi$， $\mu\leftarrow\mu+\Delta\tau$；//公式(6)中变量赋予初值，调整 μ 值

while $|f|>\epsilon$ //每次迭代计算中间参数 α 值，当 $|f|\le\epsilon$ 时，得到一个 α 值

{ $f\leftarrow 0$， $\alpha\leftarrow\alpha+\Delta\xi$;//按照增量 ξ，调整 α 值

for i = 0 to n-1

{for j = 0 to n-1

{ $a\_de\leftarrow 1$， $a\_nu\leftarrow 0$， $b\_de\leftarrow 1$， $b\_nu\leftarrow 0$;

for d = 0 to n-1

{ $a\_de\leftarrow a\_de\times\left(\frac{2i+1}{2n}-\frac{2d+1}{2n}\right)$;//计算 a_i 的分母

$b\_de \leftarrow b\_de \times \left(\frac{2j+1}{2n} - \frac{2d+1}{2n}\right)$;//计算 b_j 的分母

for p = 0 to n-d //生成 k_q 所有可能的值

{for q = 0 to n-1

{if $p \le q \le p+d-1$

$k_q \leftarrow 0$;

else

$k_q \leftarrow 1$;

}

$a\_nu \leftarrow a\_nu + \left(\prod_{\substack{q=0 \\ q\neq i}}^{n-1}\left(-\frac{2\times q+1}{2n}\right)^{k_q}\right) \Big/ (d+1)$;//计算 a_i 的分子

$b\_nu \leftarrow b\_nu + \left(\prod_{\substack{q=0 \\ q\neq j}}^{n-1}\left(-\frac{2\times q+1}{2n}\right)^{k_q}\right) \Big/ (d+1)$;//计算 b_j 的分子

}

}

$a_i \leftarrow a\_nu / a\_de$, $b_j \leftarrow b\_nu / b\_de$;

$x_i \leftarrow \frac{2i+1}{2n}, y_j \leftarrow \frac{2j+1}{2n}$;

$s \leftarrow \frac{x_i^2 - x_i\lambda y_j - \sigma^2 y_j^2}{\sqrt{2\pi}\sigma^3 y_j^4} \mathrm{e}^{-\left(\frac{(x_i-\lambda y_j)^2}{2(\sigma y_j)^2} + (c\mu - c\mu\alpha)^{y_j}\right)} - \alpha$;

$f \leftarrow f + s \times a_i \times b_j$;//按公式(6)计算

}// for j

} // for i

} // while $|f| > \epsilon$

$t \leftarrow \frac{\alpha^c}{c\mu(1-\alpha)^2} + \frac{1}{\mu}$;//将 μ 和 α 代入公式(4)，求出响应时间 t

} // while t>w

return μ;

}

4.3. 实际求解结果

我们的研究工作是以项目：“金融大数据实时流式计算及存储架构研究”为背景，旨在针对金融领域产生的大数据进行实时的流式计算，并对其进行高效存储和管理，提出适应于金融大数据的计算模型和存储架构。根据项目具体要求，有交易请求到达的平均速度 $\lambda = 10000$ 个/秒，并且要求交易系统的平均响应时间 $w \le 2$ 秒。我们不妨令 $\sigma = 100$，收敛精度控制参数 $\epsilon = 0.1$，坐标轴的插值节点个数 $n = 10$，服务速度自增步长 $\Delta\tau = 1$，计算中间参数自增步长 $\Delta\xi = 0.1$。

4.3.1. 单机环境服务速度的计算

单机环境中，交易系统由一台服务器构成。在单机环境中，交易系统的所有交易请求都由一台服务器处理，因单个服务器较为容易管理，这种情况一般存在于对于性能要求不高的业务场景，多为交易系

统起步时的配置。然而随着用户数的增长，旧有的单台服务器难以承受交易请求到达数量增加带来的压力，服务器配置的更新便提上了日程。在购买新的服务器时，我们期望新的服务器能够拥有更快的服务速度，使之能够在单位时间内处理更多的交易请求。由此，在单机环境中，我们的项目更为关注服务器的服务速度。因为现实应用中要求交易请求的到达的平均速度 $\lambda=10000$ 个/秒，并且要求交易系统的平均响应时间 $w\le 2$ 秒，为了计算服务器的服务速度，我们不妨调用算法 1：SSC，则可得到交易系统要求的服务速度 $\mu\approx 11500$ 个交易请求/秒。

4.3.2. 集群环境服务速度的计算

集群环境中，交易系统由多台服务器构成。在集群环境中，交易系统的所有交易请求会由多台服务器处理。在单机环境的服务器配置的更新换代到达瓶颈时，交易系统一般会转为应用多台服务器。然而随着业务场景的复杂度的进一步加深，用户数的进一步的增长，旧有的多台服务器难以承受日益增加的压力，我们不得不考虑增加服务器的数量，或者提升服务器的配置。在增加服务器的数量或者提升服务器的配置时，我们期望新的服务器集群能够拥有更快的服务速度。由此，在集群环境中，我们的项目同时关注服务器的服务速度和服务器的数量。又因集群环境中，交易系统的服务器数量可以有很多的选择，而在确定不同数量的服务器个数之后，服务速度的计算方式都是相同的。当目标服务器个数 $c=2$ 时，我们只需要调用算法 1 即可求得每台服务器的服务速度。与之类似，当 c 等于 4，8 乃至 100 等情况时，服务速度的计算方式都是相同的，事实上配置任意个服务器的数量都可以，原理是相同的。为了方便讨论，不失一般性，本文关于集群的服务速度的计算，不妨假设服务器个数 $c=8$，因为这是交易计算中心最基本的配置和做法，因此我们以 $c=8$ 为典型例子。因为现实应用中要求交易请求的到达的平均速度 $\lambda=10000$ 个/秒，并且要求交易系统的平均响应时间 $w\le 2$ 秒，为了计算服务器的服务速度，我们不妨调用算法 1：可得到每台服务器的服务速度 $\mu\approx 1450$ 个交易请求/秒。

5. 交易系统服务器的物理性能配置

在第 4 节中，已经求出了符合要求的交易系统服务速度(单机和集群)，即服务能力要求，但是这个服务速度指标 μ 不能直接对应服务器的硬件配置参数，即不能根据 μ 的值直接去升级或购买新的服务器，因为升级或购买新的服务器时，只能依据服务器硬件的主频，或每秒执行指令的条数作为主要指标依据[13]。为了解决这个问题，我们可以利用已有的多台、多类型计算机设备，通过实验来确定 μ 和服务器的硬件配置之间的对应关系。

5.1. 单机交易服务速度μ和硬件配置性能指标实验数据

我们采用多台配置不同的 Dell 服务器进行试验，其指令执行速度从 200,000,000 次浮点加法计算/秒到 700,000,000 次浮点加法计算/秒不等，内存 32GB，总存储量 1TB，运行的操作系统为 Red Hat Linux，交易系统运行于大数据处理平台 Storm 上[14]，数据库采用 Mysql。交易请求数据通过 Java 的模拟测试框架 Mockito 生成，模拟生成的数据会被交易系统使用，平均每一个交易请求会需要五次查询和一次数据库插入操作。为了测试在不同配置的服务器上的交易系统的极限服务速度，我们预先将 100 万个交易请求读入内存当中，并测试其处理完全部交易请求所需的时间，得出平均服务速度。

我们用服务器每秒所能执行的指令条数 ρ 来量化服务器本身的物理配置。因而，求解服务器配置问题可以转化为：已知低要求，低配置场景下的 μ-ρ 对应关系，推导出高要求高配置场景下的 μ-ρ 对应关系。在实际研究工作中，我们已有的、老旧的服务器往往在性能上远远无法满足与日俱增的在线交易需求，因此如何根据已有的、低端服务器资源，分析推导出服务速度 μ 和指令执行速度 ρ 的对应关系是必须解决的问题。

为了建立服务速度 μ 和指令执行速度 ρ 的对应关系，我们利用同济大学高性能计算中心的十台配置不同的机器上运行交易系统程序，得到表 1 中的试验数据，指令执行条数的单位是 10^8 浮点加法指令数/秒，服务速度的单位是交易请求个数/秒。

Table 1. Correspondence table
表 1. 对应关系表

指令执行条数	服务速度 μ
1.7193	1397
2.1840	1644
2.8314	2218
3.3457	3128
4.2321	4099
4.9637	5219
5.4400	6033
6.0268	7110
6.5293	7836
6.8268	8281

表 1 为指令执行速度 ρ 和服务速度 μ 的对应关系所制成的表格。从表 1 中我们可以看到，当服务器的指令执行速度提升之后，其服务速度也会随之提升，但不是简单的线性关系。

5.2. 通过数据拟合获得 μ-ρ 解析关系

多项式函数拟合因其计算过程简便，模型容易理解，和对于小数据量拟合效果好而被广泛的应用于离散数据点的曲线拟合[15]。

在本文中我们应用多项式函数来对数据进行拟合，我们定义执行速度 ρ 和服务速度 μ 之间存在如下近似关系：

$$\rho = p_0 + p_1\mu^i + p_2\mu^2 + \cdots + p_M\mu^M = \sum_{i=0}^{M} p_i\mu^i \tag{9}$$

在经过多次试验之后，我们发现，当 $M = 3$ 时，多项式函数对于数据的拟合效果最好。系数 p_3，p_2，p_1 和 p_0 的值可以通过最小二乘法得出，分别为 8.57×10^{-12}，-1.56×10^{-7}，15.32×10^{-4} 和−0.03。因此，μ-ρ 的对应关系可以由公式(10)来表示。

$$\rho = 8.57\times10^{-12}\mu^3 - 1.56\times10^{-7}\mu^2 + 15.32\times10^{-4}\mu - 0.03 \tag{10}$$

5.3. 服务器硬件配置的最终确定

如果采用单机服务器，根据我们项目的要求，规定高峰时期，交易系统能够在交易请求到达速度为 10,000 条每秒的情况下，将请求响应时间 w 控制在 2 秒之内。从 4.3.1 节可知，单机服务器的目标服务速度 μ 为 1.15 万个交易请求/秒。将 1.15 万个交易请求/秒代入公式(10)，可得单机服务器的目标性能应当达到 9.98×10^8 次浮点加法指令数/秒。

又因集群环境中，交易系统的服务器数量可以有很多的选择，而在确定不同数量的服务器个数之后，服务速度的计算方式都是相同的。因此，当 c 等于 2，4，8 乃至 100 等情况时，服务速度的计算方式都

是相同的，事实上配置任意个服务器的数量都可以，原理是相同的。

因此，若采用集群服务器，不妨以服务器个数 $c = 8$ 的集群环境为例，给出集群环境下，服务器配置的最终确定方案。从 4.3.2 节可知，当 $c = 8$ 时，集群环境的单台服务器的目标服务速度 μ 为 1450 个交易请求/秒。将 1450 个交易请求/秒代入公式(10)，可得服务器的目标性能应当达到 1.89×10^8 次浮点加法指令数/秒。

5.4. 项目实际应用效果

通过以上的研究，无论是单机或者是集群架构，我们最终都可以确定服务器的配置。“金融大数据实时流式计算及存储架构”项目已经结题，并生成相关验收报告。我们的研究已经通过专家评审，符合项目委托方的要求，受到客户的满意，已证明确实可以指导服务器配置扩容和改造升级。

6. 结束语

本文提出了一种基于排队论的在线交易系统的服务器配置决策方法。在该方法中，我们利用排队论来对交易请求的处理过程进行建模，求出交易请求平均响应时间的计算公式，根据公式推导，给出服务器的服务速度的计算算法，接着，本文通过实验求出低要求，低配置场景下服务器的服务速度与服务器物理配置之间的对应关系，借助多项式拟合，计算得到高要求，高配置场景下的服务器物理配置。针对传统排队论建模产生的不准确、不令人满意的问题，我们假设交易请求的到达规律符合正态分布，相比负指数分布，正态分布更符合高峰时期交易请求的到达规律。为了提升该模型的可用范围，我们利用排队论的 G/M/c 模型来进行建模，该模型适用于多种交易请求的到达规律，通过该模型得出的计算公式可同时应用于单机与集群环境的服务器配置。但是，在复杂的分布式环境的服务器配置扩容与改造升级方面，还不完善，下一步将充分利用排队论网络模型增强服务器配置决策的适用范围和准确度。

基金项目

国家社科基金项目(17BTQ086)，国家自然基金项目(62072337)，CCF 信息系统开放课题(CCFIS 2018-01-03)，实验教改项目(1380104112)。

参考文献

[1] 刘文清. 应用系统服务器的配置估算与调整[C]//国网信息通信有限公司、中国电机工程学会电力信息化专委会. 2008 电力行业信息化年会会议论文集. 2008: 4.

[2] 陈英达, 钟苏生, 周开东, 等. 信息系统资源估算方法研究[J]. 信息技术与信息化, 2017(11): 133-136.

[3] Doran, D., Lipsky, L. and Thompson, S. (2010) Cost-Based Optimization of Buffer Size in M/G/1/N Systems under Different Service-Time Distributions. *Ninth IEEE International Symposium on Network Computing and Applications*, Cambridge, 15-17 July 2010, 28-35. https://doi.org/10.1109/NCA.2010.11

[4] Lee, Y.C., Wang, C., Zomaya, A.Y., *et al*. (2012) Profit-Driven Scheduling for Cloud Services with Data Access Awareness. *Journal of Parallel and Distributed Computing*, **72**, 591-602. https://doi.org/10.1016/j.jpdc.2011.12.002

[5] Khazaei, H., Misic, J. and Misic, V.B. (2011) Performance Analysis of Cloud Computing Centers Using m/g/m/m + r Queuing Systems. *IEEE Transactions on Parallel and Distributed Systems*, **23**, 936-943. https://doi.org/10.1109/TPDS.2011.199

[6] Cao, J., Hwang, K., Li, K., *et al*. (2012) Optimal Multiserver Configuration for Profit Maximization in Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, **24**, 1087-1096. https://doi.org/10.1109/TPDS.2012.203

[7] Chiang, Y.J. and Ouyang, Y.C. (2014) Profit Optimization in SLA-Aware Cloud Services with a Finite Capacity Queuing Model. *Mathematical Problems in Engineering*, **2014**, Article ID: 534510. https://doi.org/10.1155/2014/534510

[8] Mei, J., Li, K., Ouyang, A., *et al*. (2015) A Profit Maximization Scheme with Guaranteed Quality of Service in Cloud Computing. *IEEE Transactions on Computers*, **64**, 3064-3078. https://doi.org/10.1109/TC.2015.2401021

[9] 廖倩文, 潘久辉, 王开杰. 基于排队理论的云计算中心性能分析模型[J]. 计算机工程, 2015, 41(9): 51-55.

[10] 张江强, 赵宁, 刘文奇. 具有两类请求的云计算中心服务器数量的优化[J]. 智能系统学报, 2017, 12(5): 601-607.

[11] 孙荣恒. 排队论基础[M]. 北京: 科学出版社, 2002.

[12] Yeh, K.C. and Kwan, K.C. (1978) A Comparison of Numerical Integrating Algorithms by Trapezoidal, Lagrange, and Spline Approximation. *Journal of Pharmacokinetics & Biopharmaceutics*, **6**, 79-98. https://doi.org/10.1007/BF01066064

[13] 从树. 服务器硬件选型测试平台的设计与实现[D]: [硕士学位论文]. 武汉: 华中师范大学, 2017.

[14] 孙大为, 张广艳, 郑纬民. 大数据流式计算: 关键技术及系统实例[J]. 软件学报, 2014, 25(4): 839-862.

[15] 刘慧婷, 张旻, 程家兴. 基于多项式拟合算法的 EMD 端点问题的处理[J]. 计算机工程与应用, 2004(16): 84-86 + 100.