基于加权类比的软件成本估算方法

赵小敏 曹光斌 费梦钰 朱李楠
(浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 软件成本估算是软件项目开发周期、管理决策和软件项目质量中最重要的问题之一。针对软件研发成本估算在软件行业中普遍存在不准确、难以估算的问题,提出一种基于加权类比的软件成本估算方法,将相似度距离定义为具有相关性的马氏距离,通过优化的粒子群算法优化后得到权值,并用类比法估算软件成本。实验结果表明,该方法具有比非加权类比、神经网络等非计算模型方法更高的精确度。实际案例测试表明,该方法在软件开发初期基于需求分析的软件成本估算比专家估算有更精确的评估结果。

关键词 软件成本估算,马氏距离,加权类比,粒子群优优化
中图法分类号 TP391 **文献标识码** A

Software Cost Estimation Method Based on Weighted Analogy

ZHAO Xiao-min CAO Guang-bin FEI Meng-yu ZHU Li-nan
(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract Software cost estimation is one of the most important issues in the cycle of development, management decision, and in the quality of software project. Aiming at the common problems of software cost estimation in the software industry, such as inaccuracy of cost estimation and estimation difficulty, this paper presented a weighted analogy-based software cost estimation method. In this method, the similarity distance is defined as the Mahalanobis distance with correlation, and the weight is obtained by particle swarm optimization. The software cost is estimated by analogy method. The result shows that this method has high accuracy compared with non-computational based model methods such as non-weighted analogy and neural networks. At the same time, the actual cases show that this method is more accurate than expert estimation in software cost estimation based on demand analysis at the early stage of software development.

Keywords Software cost estimation, Mahalanobis distance, Weighted analogy, Particle swarm optimization

1 引言

IT 从业人员经常会被问到开发一个网站多少钱,开发一个手机 App 多少钱,开发一个系统需要多少钱? 要回答这个问题,需要对软件的研发成本进行评估。软件的功能需求和非功能需求是决定软件成本的主要因素,甚至不同的开发团队对具体明确需求的软件开发的预算也是不同的。

软件成本估算是通过一套流程或模型对软件的研发成本进行评估的行为。软件成本估算贯彻整个软件开发周期,包括需求分析、系统设计、代码编写、系统测试以及系统实施和维护等阶段。过低或过高的软件成本估算都会影响整个项目的管理,导致项目交付延迟和软件项目质量低下,浪费人力、物力资源等。合理的成本估算有利于软件项目的管理与成本控制,还可以规范软件服务外包市场,减少恶意竞标的情形。

当前软件成本估算方法大致分为基于计算模型方法和非计算模型方法两大类。基于计算模型的软件成本估算方法主要以 IFPUG 功能点和 COCOMO 模型计算方法为代表。这类估算方法主要是提供一个或者多个算法模型,将软件成本的估算演化为一系列影响软件成本的特性参数通过算法模型计算结果的过程。由于计算模型提出的时间大多较早,软件设计开发的流程较多年前有较大变化,因此基于计算模型的估算方法已经很难用于准确估算现阶段国内软件的研发成本。非计算模型的软件成本估算方法不再采用计算公式来估算成本,而是利用大量历史数据对待估算的软件项目成本进行估算,如专家估算方法、神经网络、类比估算方法或其他机器学习估算方法等。神经网络估算方法是将影响软件成本的特性参数作为输入来训练神经网络,进而使用神经网络来估算成本或者工作量[1-2]。通常,为了达到快速收敛以及避免局部最优解的目的,国内外学者使用人工蜂群、粒子群等算法对神经网络进行优化[3-4]。神经网络估算法在历史数据数量较少的情况下,容易达到过拟合的状态,极大地影响估算结果。类比估算方法是通过将待估算的软件项目与已完成的项目进行对比来预测软件项目的成本。从广义角度来讲,专家估算方法也是一种类比方法[5]。较神经网络估算法而言,类比方法在历史数据数量不足的情况下仍能保证一定的估算精确度。现有类比法主要以欧氏距离作为衡量项目之间差异的标准,

本文受国家自然科学基金(61701443)资助。

赵小敏(1976-),男,博士,副教授,CCF 会员,主要研究方向为无线传感器网络、信息安全和软件成本评估,E-mail:zxm@zjut.edu.cn;**曹光斌**(1992-),男,硕士生,主要研究方向为软件成本评估;**费梦钰**(1992-),女,硕士生,主要研究方向为软件成本评估;**朱李楠**(1982-),男,博士,讲师,主要研究方向为云制造、制造业信息化等,E-mail:zln@zjut.edu.cn(通信作者)。

而忽略了影响成本的特性参数之间的相关性。本文提出一种基于加权马氏距离的优化粒子群类比评估方法,其以基于相关性协方差的马氏距离作为衡量项目相异度距离的标准,并用优化粒子群算法计算类比的权值,从而最终完成成本评估。

2 研究现状

类比法因具有直观、易于理解、样本学习能力强等符合人们习惯思维的特点,被广泛应用于软件成本估算[6]。基于类比法的软件成本评估方法用于衡量两个软件项目之间的相似性的指标主要有欧氏距离与非欧氏距离,这两类距离的定义为两个软件项目中多个影响成本的特性参数相异差的累加和。对于各个影响成本的特性参数,通常采用对特性参数加权的方式进行累加,以达到更准确的估算结果。类比法的各个成本影响着特性参数的权值,当前主要有专家人为赋值和基于机器学习的方法。专家人为赋值的方法对软件成本估算的准确性依赖于专家个人的能力水平及其对软件行业的理解,在不同领域的软件成本估算中无法保证结果的精确性。基于机器学习的成本特性参数权值估算方法采用启发式算法等对权值进行赋值,通常在精确性上较专家人为赋值方法有一定的优势。

Papatheocharous 等通过神经网络训练,将 Garson 模型引入影响软件成本的特性参数权重中[7]。该方法在实际应用中无法避免神经网络的缺陷,在样本特性参数维度过高以及样本数量不足的情况下,神经网络的训练结果存在过拟合现象,从影响了成本估算的准确性。吴登生等采用 PSO 粒子群算法优化类比法中的属性权值[8]。而杨抒等在此基础上,采用一种惯性权重属性非线性递减的粒子群算法对影响成本的各个特性参数进行权值拟合[6],该方法在局部求解方面较惯性权重线性递减的 PSO 算法有一定优势,避免了求解结果早熟和陷入局部最优解的情况。除了修改惯性权重因子,Dizaji 等采用一种混沌粒子群算法估算软件成本[9],该方法可引导粒子快速跳出局部最优,加快收敛。

上述方法在定义软件项目之间的相似程度时均采用了加权欧氏距离。欧氏距离的适用范围仅限定在正交空间,即影响软件成本的各个特性参数之间互相独立,而不具备相关性。而在实际的软件评估案例评判中,各个特性参数的属性之间存在多重关联,从而使得估计两个软件项目之间的相似度上存在较大的误差,如业务操作数与实体数量存在一定的正相关关系。针对该问题,文中提出一种基于加权马氏距离的类比方法,采用优化 PSO 算法优化权重,最后采用 Desharnais 数据集对该算法进行精确度验证。

3 基于加权马氏距离的 PSO 优化算法

3.1 项目之间的相异度距离

欧氏距离也称欧几里得距离(Euclidean Metric),是一种简单、直观的距离表示法,它表示空间中两个点之间的距离。在二维空间中,两个点的加权欧氏距离即这两个点的连线长度。欧氏距离的具体定义如式(1)所示:

$$Dis(X_i,X_j)=\sqrt{\sum_{k=1}^{n}w_k(x_{ik}-x_{jk})^2} \tag{1}$$

其中,X_i 和 X_j 表示两个 n 维样本数据的向量,x_{ik} 和 x_{jk} 分别表示项目 i 与项目 j 在第 k 维度软件成本影响特性参数的取值,w_k 表示第 k 维度特性参数的权值。显然,在各个维度的特性参数相互独立时,欧氏距离是有效的,而在实际软件成本的估算中,该距离不能有效表示软件项目之间的真正相似程度。

针对欧氏距离不能很好地定义特性参数之间的联系,无法很好地估算项目之间的相异度距离的问题,本文采用带协方差矩阵的马氏距离(Mahalanobis Distance)来定义软件项目之间的相异度。马氏距离即用于表示数据的协方差距离,与欧氏距离不同,其考虑了各个参数特性之间的联系;同时,在相异度距离的空间形状上呈现不规则性,更接近估算软件之间的真实相异度距离,也更容易使用户理解不同特性参数之间的联系。对特性参数进行加权可以降低非重要特性参数的权重,突出重要特性参数对相异度距离的贡献度,去除样本的部分冗余信息。

记软件成本的特性参数为向量 $X=(x_1,x_2,x_3,\cdots,x_n)$,其中 n 表示软件成本特性参数的维度。各个成本特性参数在所有训练样本中的均值为 $\mu=(\mu_1,\mu_2,\mu_3,\cdots,\mu_n)$。则成本特性参数 x_i 与 x_j 之间的协方差定义如式(2)所示:

$$Cov(x_i,x_j)=\frac{\sum_{k=1}^{n}(x_{ki}-\mu_i)(x_{kj}-\mu_j)}{n-1} \tag{2}$$

软件成本特性参数作为多维的列向量,其协方差矩阵的定义如式(3)所示:

$$\Sigma=\begin{pmatrix} Cov(x_1,x_1) & Cov(x_1,x_2) & \cdots & Cov(x_1,x_n) \\ Cov(x_2,x_1) & Cov(x_2,x_2) & \cdots & Cov(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(x_n,x_1) & Cov(x_n,x_2) & \cdots & Cov(x_n,x_n) \end{pmatrix} \tag{3}$$

假设两个软件项目的特性参数向量为 $A=(a_1,a_2,a_3,\cdots,a_n)$ 和 $B=(b_1,b_2,b_3,\cdots,b_n)$,则这两个软件项目的相异度定义如式(4)所示:

$$Dis(A,B)=\sqrt{(A-B)^w\Sigma^{-1}w(A-B)^{-1}} \tag{4}$$

其中,w 为斜对角矩阵形式的权重因子,Σ 表示软件成本特性参数在样本总体的维度为 $n\times n$ 的协方差矩阵。

马氏距离解决了欧氏距离只在样本各个属性参数相互独立的情况下才能计算的问题,定义了表示属性参数之间的相关度大小的协方差矩阵 Σ,当协方差矩阵 Σ 为单位矩阵时,马氏距离的定义与欧氏距离的定义相同。因此,欧氏距离是马氏距离在各个属性参数相互独立条件下的一种特殊情况。

马氏距离虽然考虑了属性参数之间的相关度,但未考虑各个属性参数对相异度距离的贡献度。考虑到属性参数对相异度距离的影响不同,本文引用了加权的马氏距离[10],具体定义如式(4)所示。对于权重的赋值,本文采用优化粒子群算法 PSO 来计算。

3.2 标准 PSO 算法

PSO 算法[11]最早由 Ebarhart 和 Kennedy 于 1995 年提出,它是从生物群体觅食行为中得到启发,被应用于优化求解问题的一种算法。在 PSO 算法中,将每个优化求解问题的潜在解记为一个 d 维向量的粒子,所有的粒子都由同一个目标函数决定适应度,适应度决定这个潜在解的优劣。此外,每个粒子都有一个速度,每个粒子根据当前粒子的最优解和整个种群的最优解决定自己“飞行”的方向和速度。在这种个体行

为和种群全局行为的相互作用下，粒子们在整个解空间中寻找最优解。

基本的 PSO 算法的形式如式(5)和式(6)所示：

$$v_k^{t+1}=w\times v_k^t+c_1\times rand()\times(pbest_k-present_k)+c_2\times rand()\times(gbest-present_k) \tag{5}$$

$$x_k^{t+1}=x_k^t+v_k^{t+1} \tag{6}$$

其中，v_k^t 表示编号为 k 的粒子在时刻 t 的速度，形式为 d 维向量；x_k^t 表示编号为 k 的粒子在时刻 t 的解向量；w 代表惯性因子；c_1 和 c_2 代表学习因子参数；$rand()$ 代表 0～1 之间的随机数；$pbest_k$ 表示粒子 k 在当前解空间找到的最优解；$gbest$ 表示在种群的解空间中找到的最优解；$present_k$ 表示粒子 k 的当前解。PSO 粒子群优化算法的步骤往往是先初始化一群粒子，每个粒子拥有随机的位置和速度，即相应的解向量和下一次迭代的粒子位置的权重。计算每个粒子的适应度，算法将个体在迭代过程中适应度最优的作为个体最优解，把种群在迭代过程中适应度最优的作为全局最优解。以式(5)与式(6)作为迭代计算公式，反复迭代算法直到符合收敛条件。具体收敛条件如式(7)所示：

$$|f(gbest^t)-f(gbest^{t-1})|<\varepsilon \tag{7}$$

其中，$gbest^t$ 表示在 t 次迭代后的全局最优解，$f(gbest^t)$ 表示对该最优解相应的适应度函数求值，ε 表示自定义的一个较小的数。

3.3 适应度函数

估算结果的评价标准即用于判定粒子群算法中粒子好坏的适应度函数。在软件成本估算方法中，往往将平均相对误差(Mean Magnitude Related Error，MMRE)作为判定标准，计算公式如式(8)和(9)所示：

$$MRE_i=\left|\frac{E_i-\hat{E}_i}{E_i}\right| \tag{8}$$

$$MMRE=\frac{1}{n}\sum_{i=1}^{n}MRE_i \tag{9}$$

式中，MRE_i 表示软件项目 i 的相对误差，E_i 表示软件项目 i 的实际成本结果，$\hat{E}_i$ 表示软件项目 i 的估算成本；n 表示数据集中的软件项目总数。

3.4 基于优化 PSO 算法的权值计算

粒子群算法容易陷入局部最优解而过早收敛。通过修改惯性因子 w 以及学习因子 c_1 和 c_2，可以在一定程度上改进粒子群算法。实验表明，当惯性因子采用[0.5，1]之间的随机值时，其比线性递减策略的精度更高，收敛速度更快。对于学习因子，采用非线性反余弦加速，c_1 先大后小，c_2 先小后大，使得粒子初期在解空间求解的过程注重个体的最优解，后期更加注重种群的最优解[12]。具体定义如式(10)以及式(11)所示：

$$c_1=c_{1e}+(c_{1s}-c_{1e})\left[1-\frac{\arccos(-2t/t_{\max}+1)}{\pi}\right] \tag{10}$$

$$c_2=c_{2e}+(c_{2s}-c_{2e})\left[1-\frac{\arccos(-2t/t_{\max}+1)}{\pi}\right] \tag{11}$$

其中，c_{1e} 和 c_{2e} 表示迭代终值，c_{1s} 和 c_{2s} 表示迭代初始值。c_{1s} 取值 2.5，c_{2s} 取值 0.5，c_{1e} 取值 0.5，c_{2e} 取值 2.5。

本文采用马氏距离的 PSO 算法优化方法。提出一种基于加权类比的软件成本估算方法。基本思路是将权重定义为向量形式的粒子，通过 PSO 算法不断迭代优化权重向量，从而得到最佳的权重向量。基本流程如算法 1 所示。

算法 1　基于马氏距离的软件成本类比法

输入：样本数据集 dataSet，待估算软件项目输入 projectData

输出：待估算软件项目成本估算结果 effort

Step 1　初始化：个体向量取值范围 paramRange，种群大小 paramNumber，迭代总次数 n，学习因子相关参数 c_{1e}，c_{2e}，c_{1s}，c_{2s} 和类比项目数 k。

Step 2　计算 dataSet 的协方差矩阵 Σ。

Step 3　用每个粒子作为权值计算每一个 $data_j$($data_j\in$ dataSet)对于其他历史样本的相异度距离。

Step 4　选取相异度距离小的 k 个样本计算适应度函数。

Step 5　更新种群的最优解 gbest 及每个个体的最优解 pbest。

Step 6　若符合收敛条件，则将 gbest 作为类比法的权值 w，否则更新所有粒子的速度以及位置向量，并跳转至 Step 3。

Step 7　用权值 w 拟合估算新软件项目的成本 effort。

4　实验及结果分析

采用 Desharnais 数据集来验证几种类比方法的有效性。Desharnais 数据集距今虽有 20 余年[13]，但其数据对方法的验证还是具有相当的可靠性。Desharnais 数据集共有 81 组数据，其中 4 组数据有个别参数缺失。去除这 4 组数据，取剩余的 77 组软件特性属性参数完整的样本数据。Desharnais 数据集中有 10 个软件特性属性参数，因此定义权值以及 PSO 的种群个体为 $w=$[TeamExp，ManagerExp，YearEnd，Length，Transactions，Entities，PointsAdjust，Envergure，PointsNonAjust，Language]。实验采用 MATLAB R2014a 版软件对方法进行验证实验，粒子的个数取 30，最大迭代次数取 300，学习因子采用非线性反余弦加速，c_{1s} 取值 2.5，c_{2s} 取值 0.5，c_{1e} 取值 0.5，c_{2e} 取值 2.5。该数据集的协方差矩阵如式(12)所示，实验结果如表 1 所列。

$$\Sigma=\begin{pmatrix}
1.000 & 0.398 & -0.154 & 0.243 & 0.156 & 0.314 & 0.273 & 0.304 & 0.303 & -0.121\\
0.398 & 1.000 & 0.069 & 0.243 & 0.172 & 0.235 & 0.248 & -0.097 & 0.215 & 0.241\\
-0.154 & 0.069 & 1.000 & -0.028 & 0.099 & 0.038 & 0.097 & -0.010 & 0.088 & 0.330\\
0.243 & 0.243 & -0.028 & 1.000 & 0.608 & 0.478 & 0.711 & 0.267 & 0.702 & -0.020\\
0.156 & 0.172 & 0.099 & 0.608 & 1.000 & 0.187 & 0.886 & 0.334 & 0.88 & 0.128\\
0.314 & 0.235 & 0.038 & 0.478 & 0.187 & 1.000 & 0.621 & 0.235 & 0.601 & -0.056\\
0.273 & 0.248 & 0.097 & 0.711 & 0.886 & 0.621 & 1.000 & 0.377 & 0.986 & 0.075\\
0.304 & -0.097 & -0.010 & 0.267 & 0.334 & 0.235 & 0.377 & 1.000 & 0.508 & -0.199\\
0.303 & 0.215 & 0.088 & 0.702 & 0.880 & 0.601 & 0.986 & 0.508 & 1.000 & 0.039\\
-0.121 & 0.241 & 0.330 & -0.020 & 0.128 & -0.056 & 0.075 & -0.199 & 0.039 & 1.000
\end{pmatrix} \tag{12}$$

表 1 实验结果

软件成本估算方法	类比项目数	*MMRE*
加权马氏距离类比	1	0.4367
	2	0.4671
	3	0.4796
加权欧氏距离类比	1	0.4629
	2	0.4631
	3	0.4673
模糊神经网络		0.4897
支持向量回归机(SVR)		0.4712

由式(12)可知,Transactions 与 PointsAdjust 的相关程度为 0.886;Transactions 与 PointsNonAjust 的相关程度为 0.886;PointsAdjust 与 PointsNonAjust 的相关程度高达 0.986,它们之间具有较强的线性正相关性;而 Envergure 与 Language 的相关程度为 −0.199,表明它们之间有一定的负相关性。由表 1 可以看出,在类比项目数 $k=1$ 时,加权马氏距离 PSO 权值优化类比法的平均误差 *MMRE* 为 0.4367,而加权欧氏距离的 *MMRE* 为 0.4629。由于欧氏距离未考虑特性参数之间的联系,加权马氏距离 PSO 权值优化类比法较加权欧氏距离有更好的表现。当类比项目数 $k=3$ 时,有较大概率采用相似度不高的项目作为类比项目,导致误差增大。同时,加权马氏距离 PSO 权值优化类比法在与其他基于机器学习的软件成本估算法进行比较时,模糊神经网络的 *MMRE* 为 0.4897,支持向量回归机(SVR)的 *MMRE* 为 0.4712。不管是模糊神经网络在高维度低样本数量的训练,还是支持向量回归机在样本维度的增加上,都有一定的过拟合风险,因此基于马氏距离的 PSO 权值优化类比法在软件成本估算的精度方面有一定的优势。

同时,从表 1 的 *MMRE* 结果来看,当类比项目数 k 增加时,两种类比法的平均误差 *MMRE* 值都有一定的升高。在类比项目数 k 取 1 时,加权马氏距离类比法得到最小的平均误差 0.4367。因此,在实际软件成本估算中,建议类比项目数 k 值取为 1。

5 实例分析

将浙江省财政厅委托专家评估的软件项目作为案例来对本文提出的基于加权类比的软件成本估算方法做进一步的实例验证。这些软件项目仅有需求说明,由专家来评估成本并将其作为政府采购预算的参考。为验证本文所提方法的有效性,选取其中 3 个软件项目进行评估,分别为地方政府性债务管理系统、学生综合管理平台、港航规费稽征系统,将其分别标记为项目 1、项目 2 和项目 3。其中,用于描述软件成本的特性参数主要取自《软件研发成本度量规范》[14] 所定义的模型参数,包括功能模块操作、行业生产率、系统业务用途、团队背景、开发语言、应用类型、质量特性等参数。使用模型参数能使类比方法的结果更具说服力,同时也便于用户填写软件项目的参数。在模型参数中,开发语言、团队背景等参数属于字符型参数,以往对于字符型参数距离的定义无法正确评估两个相异较大的参数的距离,只要不相同则定义为一个固定的值。借鉴《软件研发成本度量规范》所定义的模型参数,对字符型参数进行数值梯度化定义。该定义主要体现在各个字符型参数所在不同梯度的相异度距离的差异不同,具体相异度距离会在加权后得到修正。由此,将所有字符型参数转化为数值型参数。具体特性参数的取值如表 2 所列,3 个软件项目的估算结果如表 3 所列。

表 2 软件特性参数取值表

	项目 1	项目 2	项目 3
数据字段数	63	51	49
功能模块数	10	9	14
模块操作数	95	210	222
行业生产率	6.7	6.7	6.7
应用类型因子	1	1	1
质量调整因子	0.9	0.925	1.05
语言因子	1	1	0.6
开发团队因子	1	0.8	1

表 3 软件成本估算结果表

估算方法	项目 1	项目 2	项目 3
类比法	78.42	48.31	50.63
专家估算方法	71.20	35.58	45.61
实际采购预算	80	40	50

表 3 中结果显示,除项目 2 上类比法比专家估算方法的误差略大外,其余两个项目上的类比法较专家评估方法在精度上都有较大的优势。通过对相关样本进行分析,开发团队因子在计算软件项目的相异度距离时所占的权值较大,因此类比结果对该特性参数因子过于敏感,而与项目 2 相似的项目开发团队在因子取值上差异过大,是造成该项目类比估算结果与实际采购预算误差较大的原因。由于专家估算方法具有较大的人为因素,因此 3 个项目的专家估算结果都较实际采购预算低,偏于保守。类比法是根据历史数据与待估算软件项目相比较得出结果的方法,评估结果不依赖于个人经验,而依赖于历史数据,所以估算结果更接近实际采购预算。

结束语 本文针对软件成本普遍存在的不准确、难以估算的问题,提出一种加权类比的软件成本估算方法。经过 Desharnais 数据集的校验,该方法在精确度上较加权欧氏距离类比法、神经网络等方法有一定的提高。同时,文中通过实际案例验证了该方法的有效性。后续我们将继续研究类比法在实际应用中的参考模型的选取,以及针对历史数据集中包含个别特性参数的样本数量不足及该特性参数对相异度距离过于敏感等问题,实现更高的评估精确度。

参 考 文 献

[1] DAVE V S, DUTTA K. Neural network based models for software effort estimation: a review[J]. Artificial Intelligence Review, 2014, 42(2): 295-307.

[2] SARNO R, SIDABUTAR J, SARWOSRI. Comparison of different Neural Network architectures for software cost estimation [C]// International Conference on Computer, Control, Informatics and ITS Applications. IEEE, 2016: 68-73.

[3] WANI Z H, QUADRI S M K. Artificial Bee Colony-Trained Functional Link Artificial Neural Network Model for Software Cost Estimation[M]// Proceedings of Fifth International Conference on Soft Computing for Problem Solving. 2016: 729-741.

[4] BENALA T R, CHINNABABU K, MALL R, et al. A particle swarm optimized functional link artificial neural network (PSO-FLANN) in software cost estimation[C]// Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) Advances in Intelligent Systems and Computing. Springer Berlin Heidelberg, 2013: 59-66.

(下转第 531 页)

由表 6 中的测试数据可知，Clever-Mind 创建单一任务测试的时间与表 4 中 Kubernetes 创建容器的时间相比要长得多。为了找出耗时增加的原因，这里直接通过 Kubernetes 来创建相同的 TensorFlow 任务，经过多次测试发现，直接通过 Kubernetes 创建 TensorFlow 任务比 Clever-Mind 要快大约 700 ms，其实这是合理的，因为用户的请求需要经过 Clever-Mind 处理后才调用 Kubernetes API-Server 来创建任务，但是这与表 4 中的数据还是有较大出入，因此需要深入挖掘和分析。在创建任务时与表 4 中的唯一区别就是挂载了用户的 Storage，为了排除这个干扰项，在通过 Kubernetes 创建任务时不挂载任何目录，因为样例代码本身就存在于已构建好的 TensorFlow 镜像中，所以这里不挂载目录也不会有任何影响。在经过多次创建任务的测试后发现创建任务的速度有所提升，提升大约为 800 ms，虽然与表 4 中的结果还存在一点差距，但已相当接近。

在下一轮测试中，将 PS 和 Worker 的数量提升至 100，采用的方法很简单，只需要改变前端页面 PS 和 Worker 的数量即可，其他参数保持默认，测试结果如表 7 所列。

表 7　Clever-Mind 创建 100 个任务测试表

（单位：s）

TensorFlow 任务类型	最短耗时	最长耗时	平均耗时
PS	2.4	6.2	4.5
Worker	2.5	6.6	4.7

表 7 的数据表明，即使在创建 100 个容器的情况下，整体的耗时都是非常可观的。对于平台用户来说，简便地操作分布式 TensorFlow 是十分重要的，而快速分配资源、创建 TensorFlow 实例、高速训练则是 Kubernetes 分布式 TensorFlow 深度学习平台的优势所在。

结束语　本文主要围绕 Kubernetes 及 TensorFlow 这两个新兴的云计算及大数据框架展开研究，针对目前企业在使用分布式 TensorFlow 时无法高效地利用物理资源，并且工作效率过低，造成企业成本上升等问题，提出了容器技术的解决方案。传统的深度学习方式已经无法满足目前人工智能浪潮所带来的经济效益，高效、精确的构建及训练模型是深度学习最重要的核心步骤。因此，在对 TensorFlow 的使用现状进行了深入的分析及研究后总结得出，Kubernetes PaaS 云平台可以较好地帮助企业充分利用底层资源；而容器化 TensorFlow 可以使 TensorFlow 运行在 PaaS 平台上，最终实现在数秒内为开发人员提供上百个 TensorFlow 的分布式集群，从而达到快速构建深度学习环境的目的，极大提高了开发人员的效率。

最终测试创建 TensorFlow 容器所消耗的时间依然还有很大的优化空间，下一步将优化存储与 Kubernetes 集群之间的网络通信，并且尝试其他高性能存储系统，减少挂载带来的性能影响。

参 考 文 献

[1] ABADI M，AGARWAL A，BARHAM P，et al. TensorFlow：Large-Scale Machine Learning on Heterogeneous Distributed System[J]. arXiv：1603.04467v2，2016.

[2] 龚正，吴治辉，王伟，等. Kubernetes 权威指南：从 Docker 到 Kubernetes 实践全接触（纪念版）[M]. 北京：电子工业版社，2017：1-42.

[3] 浙江大学 SEL 实验室. Docker 容器与容器云[M]. 北京：人民邮电出版社，2016：1-27.

[4] 李航. 统计学习方法 [M]. 北京：清华大学出版社，2012：1-24.

[5] 李嘉璇. TensorFlow 技术解析与实战[M]. 北京：人民邮电出版社，2017：218-224.

[6] PEINL R，HOLZSCHUHER A F，PFITZER F. Docker Cluster Management for the Cloud-Survey Results and Own Solution [J]. Grid Computing，2016，14：265-282.

[7] Serving a TensorFlow Model[EB/OL]. https://www.tensorflow.org/serving/serving_basic.

[8] go-restful[EB/OL]. https://github.com/emicklei/go-restful.

[9] CHANG F，DEAN J，GHEMAWAT S，et al. Gruber. Bigtable：A Distributed Storage System for Structured Data[J]. ACM Transactions on Computer Systems (TOCS)，2008，26(2)：1-26.

[10] 朱林. Elasticsearch 技术解析与实战[M]. 北京：机械工业出版社，2017：6-10.

[11] https://github.com/kubernetes/examples/blob/master/staging/volumes/glusterfs/README.md.

[12] https://github.com/heketi/heketi.

[13] https://en.wikipedia.org/wiki/Network_File_System.

[14] SEYMOUR K，NAKADA H，MATSUOKA S，et al. Overview of GridRPC：A Remote Procedure Call API for Grid Computing [J]. Grid Computing，2002，2536：274-278.

[15] http://yann.lecun.com/exdb/mnist.

（上接第 504 页）

[5] BAJTA M E，IDRI A，FERNÁNDEZ-ALEMÁN J L，et al. Software cost estimation for global software development a systematic map and review study[C]// International Conference on Evaluation of Novel Approaches To Software Engineering. IEEE，2015：197-206.

[6] 杨抒，王业，乌尔柯西，等. 基于 C&S-PSO 的软件成本估算类比法特征权重优化[J]. 计算机系统应用，2015，24(7)：99-103.

[7] PAPATHEOCHAROUS E，ANDREOU A S. On the Problem of Attribute Selection for Software Cost Estimation：Input Backward Elimination Using Artificial Neural Networks[C]// Artificial Intelligence Applications and Innovations，Ifip Wg 12.5 International Conference. Springer Berlin Heidelberg，2010：287-294.

[8] 吴登生，李建平，蔡晨. 软件成本估算的粒子群算法类比模型及自助法推断[J]. 管理科学，2010，23(3)：113-120.

[9] DIZAJI Z A，KHALILPOUR K. Particle Swarm Optimization and Chaos Theory Based Approach for Software Cost Estimation[J]. International Journal of Academic Research，2014，6(3)：130.

[10] 王振丽. 基于加权 MP 马氏距离的 GS 方法研究[D]. 南京：南京理工大学，2016.

[11] KENNEDY J，EBERHART R. Particle swarm optimization[C]// IEEE International Conference on Neural Networks. IEEE，1995：1942-1948.

[12] 牛利勇，张帝，王晓峰，等. 基于自适应变异粒子群算法的电动出租车充电引导[J]. 电网技术，2015，39(1)：63-68.

[13] DESHARNAIS J M. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction[D]. Quebec：University of Montreal，1989.

[14] 中华人民共和国工业和信息化部. 软件研发成本度量规范：SJ/T 11463-2013[S]. 2013.