(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 107086929 B
(45)授权公告日 2019.12.13

(21)申请号 201710247307.5
(22)申请日 2017.04.16
(65)同一申请的已公布的文献号
申请公布号 CN 107086929 A
(43)申请公布日 2017.08.22
(73)专利权人 北京工业大学
地址 100124 北京市朝阳区平乐园100号
(72)发明人 梁毅 侯颖 苏超 陈诚 丁治明
(74)专利代理机构 北京思海天达知识产权代理有限公司 11203
代理人 沈波
(51)Int.Cl.
H04L 12/24(2006.01)
H04L 12/861(2013.01)
H04L 12/26(2006.01)
H04L 29/08(2006.01)
审查员 刘江平

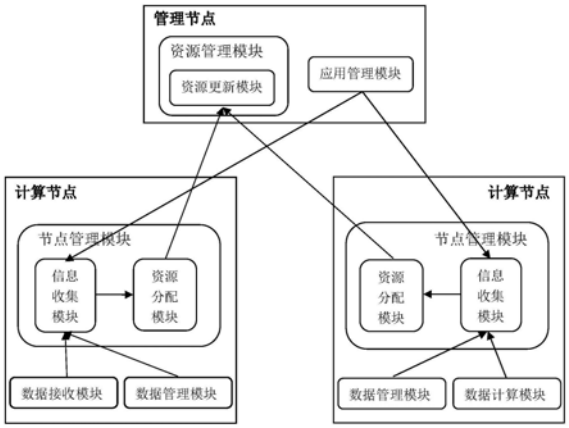
权利要求书4页 说明书11页 附图7页

(54)发明名称

一种基于排队建模的批量流式计算系统性能保障方法

(57)摘要

本发明公开了一种基于排队建模的批量流式计算系统性能保障方法，性能保障方法分为五个步骤，过程划分、组件选取、性能建模、延迟计算和瓶颈定位及优化。本方法针对批量流式计算系统在运行过程中负载强度具有明显波动性特征，抽取了批量流式计算系统中的关键组件，依据排队论理论构建系统的性能模型，并对模型进行数学解析；然后在系统运行过程中使用该模型计算不同负载强度下系统的数据处理延迟；在数据处理延迟无法满足数据处理时效性需求时，依据排队论原理定位性能瓶颈组件并给出优化配置建议。



CN 107086929 B

1.一种基于排队建模的批量流式计算系统性能保障方法，其特征在于：该性能保障方法分为五个步骤：过程划分、组件选取、性能建模、延迟计算和瓶颈定位及优化；其中，过程划分、组件选取和性能建模步骤是在批量流式计算系统在线运行前一次性完成，延迟计算、瓶颈定位及优化步骤则在系统在线运行中周期性执行；在性能建模时，对数据到达强度、组件服务时间等做如下假设：1)外部数据源的数据到达符合泊松分布；2)数据处理采用先来先服务的调度方式；3)数据到达时间间隔与组件服务时间的分布规律相互独立；4)网络是不可控因素，因此假设网络带宽足够；在本方法中，有几个基本的参数：数据块间隔$t_{blockInterval}$，批处理间隔$t_{batchInterval}$，用户期望数据处理延迟T_{max}，在线计算数据处理延迟周期t，数据在各组件的逗留时间占比阈值P_a，逗留时间标准差阈值σ_a，组件实例数量增加步长k_a；$t_{batchInterval}$取值在0.5～2秒之间；

上述方法的特征在于包括以下步骤：

(1)过程划分

将批量流式计算的数据处理过程划分为以下五个阶段：

①数据接收：从数据源获取数据并存入系统的内存缓冲区；

②数据块构建：周期性将当前缓冲区的数据封装为数据块，放入数据块队列中；

③数据块存储：将队列中的数据块存入内存或磁盘中，记录该数据块的元数据信息，并将该数据块的标识ID放入对应的数据流队列中；

④作业生成：根据设置批处理间隔，从数据流队列中提取所有未处理数据块，并根据数据的处理逻辑关系生成作业，提交到作业队列中；

⑤作业执行：将作业队列中的作业转化为并行任务，并分发给集群的任务执行器执行；

(2)组件选取

2.1)根据系统结构，选取步骤(1)中划分阶段所对应的核心组件，构建备选组件集合C，$C=\{c_i|1\leqslant i\leqslant 5\}$，设置随机组件集合$C_R$，$C_R=\varnothing$，以及常量组件集合$C_c$，$C_c=\varnothing$；其中，随机组件是指数据项在该组件的逗留时间存在随机性的组件，常量组件是指数据项在该组件的逗留时间近似为固定值的组件；

2.2)选取测试数据集$D=\{d_j|1\leqslant j\leqslant n\}$，并选取低、中、高三类不同的数据到达强度，构建数据到达强度集λ，$\lambda=\{\lambda_m|1\leqslant m\leqslant p\}$；

2.3)运行批量流式计算系统，并逐一按照λ中的到达强度向系统注入数据集D；对每个数据到达强度λ_m，$\lambda_m\in\lambda$，利用公式(1)计算该数据到达强度下数据项d_j，$d_j\in D$，在C中组件c_i，$c_i\in C$的逗留时间T_{ijm}；

$$T_{ijm}=T_{ifinish}-T_{istart} \quad (1)$$

其中，$T_{ifinish}$表示数据项d_j离开组件c_i的时间，T_{istart}表示数据项d_j到达组件c_i的时间；

2.4)对于λ中的每个数据到达强度λ_m，利用公式(2)计算该强度下数据项在组件c_i的平均逗留时间T_{im}；

$$T_{im}=\frac{\sum_{j=1}^{n}T_{ijm}}{n} \quad (2)$$

2.5)对于λ中的每个数据到达强度λ_m，利用公式(3)和公式(4)计算该强度下数据项在各

组件c_i的平均逗留时间占比P_{im}和标准差σ_{im}；

$$P_{im}=\frac{T_{im}}{T_m} \tag{3}$$

$$\sigma_{im}=\sqrt{\frac{\sum_{j=1}^{n}(T_{ijm}-T_{im})^2}{n}} \tag{4}$$

其中T_m为在数据到达强度λ_m下数据项在系统中的平均总逗留时间，即数据项在各组件平均逗留时间的总和；由公式(5)计算得到，

$$T_m=\sum_{i=1}^{5}T_{im} \tag{5}$$

2.6)利用公式(6)和公式(7)计算不同强度下数据项在各组件c_i的逗留时间的平均占比P_i和标准差σ_i；

$$P_i=\frac{\sum_{m=1}^{p}P_{im}}{p} \tag{6}$$

$$\sigma_i=\frac{\sum_{m=1}^{p}\sigma_{im}}{p} \tag{7}$$

2.7)对于C中每一个组件c_i，若$P_i\geqslant P_a$且$\sigma_i\geqslant\sigma_a$，则将组件$c_i$加入随机集合$C_R$，若$P_i\geqslant P_a$且$\sigma_i<\sigma_a$，则将组件$c_i$加入常量集合$C_c$；

(3)性能建模

3.1)对于任一$c_i\in C_c$，数据在该组件c_i的逗留时间设为常量T_i；

3.2)对于任一$c_i\in C_R$，则根据组件c_i的数据到达及服务特征，选取相应排队模型，并根据对应模型计算出数据在该组件的等待时间W_{qi}和服务时间S_i，数据在在该组件的逗留时间为$T_i=W_{qi}+S_i$；

3.2.1)参考步骤(1)中的过程划分，如果组件属于第①阶段，则数据以λ_0的泊松过程到达，视为单个服务台或多个服务台的M/M/1排队模型，利用公式(8)和公式(9)计算等待时间W_{qi}和组件服务时间S_i；

$$W_{qi}=\begin{cases}\dfrac{\lambda_i}{\mu_i(\mu_i-\lambda_i)} & k_i>\dfrac{\lambda_i}{\mu_i}\\ +\infty & k_i\leq\dfrac{\lambda_i}{\mu_i}\end{cases} \tag{8}$$

$$S_i=\frac{1}{\mu_i} \tag{9}$$

其中，λ_i表示该组件的数据到达速率，μ_i表示该组件的服务速率，k_i表示服务台数目；

3.2.2)参考步骤(1)中的过程划分，如果组件属于第②③④阶段，则数据在固定时间间隔到达，视为单个服务台/多个服务台的D/M/1排队系统，利用公式(10)计算等待时间W_{qi}，服务时间S_i可利用3.2.1)中的公式(9)计算得到；

$$W_{qi}=\begin{cases}\dfrac{1}{\mu_i}\cdot\dfrac{\delta}{1-\delta} & k_i>\dfrac{\lambda_i}{\mu_i}\\ +\infty & k_i\le\dfrac{\lambda_i}{\mu_i}\end{cases} \tag{10}$$

其中，δ是方程$\delta=e^{-\mu/\lambda(1-\delta)}$的最小绝对值根；

3.2.3)参考步骤(1)中的过程划分，如果组件属于第⑤阶段，则其服务特征满足一个作业被分为多个子任务并行处理，并且每个子任务需等到其他所以子任务被处理结束才能重新聚合并离开时，可看做Fork-Join模型，利用公式(11)计算服务时间S_i；

$$S_i=\max(T_{l0},T_{l1},\ldots,T_{lk}) \tag{11}$$

其中，T_{lk}表示第k个并行子任务的执行时间；

3.3)将数据项在各关键组件的逗留时间加总即得系统的平均数据处理延迟，如公式(12)：

$$T=\sum_{i=1}^{u}T_i \tag{12}$$

(4)延迟计算：在批量流式计算系统运行过程中，周期性根据公式(12)计算系统在线数据处理延迟；

4.1)计算每个组件i的数据到达速率λ_i、服务速率μ_i和服务台数量k_i，其中，k_i为当前可使用的组件实例数量；

4.1.1)若组件$c_i\in C_R$，且数据到达和服务特征满足M/M/1模型时，其到达速率和服务速率可利用公式(13)和公式(14)计算得到；

$$\lambda_i=\frac{n_{event}}{t} \tag{13}$$

$$\mu_i=\frac{1}{t_{event}} \tag{14}$$

其中，n_{event}表示在最近的时间周期t内到达的数据项总量，t_{event}表示对单个数据项的平均服务时间；

4.1.2)若组件$c_i\in C_R$，且数据到达和服务特征满足D/M/1模型时，其到达速率和服务速率可利用公式(15)和公式(16)计算得到；

$$\lambda_i=\frac{1}{t_{Interval}} \tag{15}$$

$$\mu_i=\frac{1}{t_{block}} \tag{16}$$

其中，$t_{Interval}$表示应用程序中设置的时间间隔，t_{block}表示对单个数据块的平均服务时间；

4.1.3)若组件$c_i\in C_R$，且服务特征满足Fork-Join模型时，其到达速率和服务速率可利用4.1.2)中的公式(15)和公式(16)计算得到；

4.2)将步骤4.1)中获取的参数值带入步骤3.3)的公式(12)中得到数据处理延迟T；

4.3)比较数据处理延迟T和用户期望数据处理延迟T_{max}，若$T\le T_{max}$，说明数据处理延迟可满足用户时效性需求，则执行步骤(6)，否则执行步骤(5)；

(5)瓶颈定位及优化

5.1)若T趋于无穷大,查找满足$k_i \le \frac{\lambda_i}{\mu_i}$的组件,将其服务台数目设置为$k_i = \left\lceil \frac{\lambda_i}{\mu_i} \right\rceil$,并跳转至步骤(4)重新计算数据处理延迟T,其中,⌈⌉为上取整符号;

5.2)若$T > T_{max}$,则选取逗留时间占比P_i最大的组件,将其服务台数目设置为$k_i = k_i + k_a$,并跳转至步骤(4)重新计算数据处理延迟T;其中,k_a为组件实例数量增加步长;

(6)回溯:在时间周期t后,判断应用程序是否结束,是则转至步骤(7),否则转到步骤(4);其中时间周期t是指相邻两次性能评估之间的时间间隔;

(7)结束:中止对系统的性能评估及优化。

2.根据权利要求1所述的一种基于排队建模的批量流式计算系统性能保障方法,其特征在于:本方法在Spark Streaming系统的每个计算节点上增设信息收集模块,用于实时地收集数据到达组件和离开组件的时间以及各组件数据到达率及组件实例数量,本方法将收集的数据到达和离开组件的时间信息构成步骤(2)中进行组件选取的基础,收集的各组件数据到达率及组件实例数量信息构成步骤(4)中进行延迟计算的依据;为了实现该方法,在Spark Streaming系统中增设资源分配模块,用于根据信息收集模块提供的信息,完成对组件实例数量的增加。

3.根据权利要求1所述的一种基于排队建模的批量流式计算系统性能保障方法,其特征在于:性能保障方法依附于现有Spark Streaming批量流式处理系统,通过新增相应的软件模块实现;该Spark Streaming批量流式处理系统的平台由多个计算机服务器组成或平台节点组成,服务器间通过网络连接;平台节点分为两类,包括一个管理节点和多个计算节点;所依附的平台包含以下核心软件模块:资源管理模块、节点管理模块、应用管理模块、数据接收模块、数据管理模块和数据计算模块;其中,资源管理模块负责维护平台中所有节点的资源信息,仅在管理节点上部署;节点管理模块负责启动和结束任务执行器,并监控本节点上数据在各组件的执行情况,每个计算节点上均部署一个节点管理模块;应用管理模块负责流应用的依赖解析以及作业的生成与调度,在Spark Streaming平台中提交和运行的每个流应用均对应一个应用管理模块;数据接收模块负责数据项的接收及数据块的构建,数据管理模块负责数据块的存储管理,数据计算模块负责作业的计算;上述软件模块中,资源管理模块和节点管理模块在Spark Streaming系统启动时即部署运行,应用管理模块、数据接收模块、数据管理模块和数据计算模块分别在相应的流应用提交运行时触发部署运行;

为实现本方法须在既有Spark Streaming系统中新增软件模块,包括信息收集模块、资源分配模块和资源更新模块;其中,信息收集模块负责对数据接收模块、数据管理模块、应用管理模块及数据计算模块中各组件数据到达率及组件实例数量、数据到达和离开各组件时间的收集,资源分配模块负责根据瓶颈定位及优化方法的判断信息,调整各组件的实例即服务台数量;信息收集模块、资源分配模块作为节点管理模块的子模块部署于各计算节点上;资源更新模块作为资源管理模块的子模块,部署于管理节点,负责收集计算节点上的组件实例数量变动信息,并修改维护各计算节点可分配的资源信息。

一种基于排队建模的批量流式计算系统性能保障方法

技术领域

[0001] 本发明属于分布式计算领域，具体涉及批量流式计算系统的性能分析与优化方法。

背景技术

[0002] 流式数据是大数据的一种重要数据类型，具有连续性、易失性、动态性等特征。大数据流式计算是针对流式数据的分析处理技术，以数据处理时效性为性能目标，快速挖掘流式数据价值。批量流式计算是大数据流式计算的一个重要分支。批量流式计算的核心技术特征是将接收的流式数据按时间顺序切分为多个小批次，并使用类MapReduce的批量计算周期地进行处理。批量流式计算在物联网传感器实时数据流处理、社交网络数据流分析等领域有广阔的需求和应用前景，已成为研究热点。

[0003] 流式数据的动态性和处理时效性需求驱动批量流式计算系统在线性能保障方法研究。在线性能保障是指在系统负载动态变化的前提下，通过自适应的性能分析和优化，保障系统稳定地达到预期的性能目标。目前，针对批量流式计算系统的性能保障方法尚处于初级阶段，主要集中在以批量流式计算的某一关键阶段存在性能瓶颈为假设前提，提出相应的性能优化方案。然而，批量流式计算系统是由数据接收、存储、处理等多阶段多组件构成的复杂系统，各阶段组件之间存在数据传递和性能依赖关系。如何针对动态变化的数据负载快速评估出系统的性能并能够在诸多组件中准确定位性能瓶颈是进行性能优化的前提，其本身也存在较大的技术挑战。目前，批量流式计算系统在线性能保障方法尚缺乏有效的性能评估和瓶颈定位方法，这使得所提出的性能优化方案在实际运用中存在盲目性。具体而言，既有性能保障方法存在如下问题：

[0004] (1)无法根据系统负载的变化快速评估系统性能，实施性能保障存在滞后性。既有方法在系统负载强度变化时，需要经过一段时间的观测统计才能确定在新的负载强度下系统的平均性能。这难以适应流式系统负载快速变化的特征，导致性能保障存在滞后性，不能在负载强度变化时及时对系统性能做出评估。

[0005] (2)没有综合考虑各阶段组件的性能依赖关系，无法准确定位性能瓶颈。现有技术在系统性能无法达到预期目标时，没有考虑各组件间的复杂的性能依赖关系，而是简单地选择某一关键组件实施优化方案，无法准确定位性能瓶颈，且无法对所采用的优化方案进行优化后的性能评估，降低了性能保障效率。

[0006] 综合而言，目前尚未存在基于准确性能评估和瓶颈定位的批量流式计算系统性能保障方法。

发明内容

[0007] 针对上述问题，本发明提出一种基于排队论的批量流式计算系统在线性能保障方法。本发明首先抽取批量流式计算系统中的关键组件，依据排队论理论构建系统的性能模型，并对模型进行数学解析；然后在系统运行过程中使用该模型计算不同负载强度下系统

的数据处理延迟;在数据处理延迟无法满足数据处理时效性需求时,依据排队论原理定位性能瓶颈组件并给出优化配置建议。

[0008] 排队论是人们研究大量服务过程的一门数学理论。排队论将排队系统抽象为输入过程、排队规则和服务规则三个部分,并利用排队系统的特征选取相应的排队模型,计算顾客在系统中的平均响应时间,包括等待时间和服务时间。排队论广泛应用于计算机网络、分布式系统、生产运输等资源共享的随机服务系统。本发明就是利用排队论原理对批量流式计算系统进行性能分析,使得系统的各个组件能有效运行,并发挥最大效益。

[0009] 本发明提供技术方案如下:

[0010] 本发明所述的性能分析和优化方法主要分为五个步骤:过程划分、组件选取、性能建模、延迟计算和瓶颈定位及优化。其中,过程划分、组件选取和性能建模步骤是在批量流式计算系统在线运行前一次性完成,延迟计算、瓶颈定位及优化步骤则在系统在线运行中周期性执行。在性能建模时,对数据到达强度、组件服务时间等做如下假设:1)外部数据源的数据到达符合泊松分布;2)数据处理采用先来先服务的调度方式;3)数据到达时间间隔与组件服务时间的分布规律相互独立;4)网络是不可控因素,因此假设网络带宽足够。在本方法中,有几个基本的参数:数据块间隔$t_{blockInterval}$,批处理间隔$t_{batchInterval}$,用户期望数据处理延迟T_{max},在线计算数据处理延迟周期t,数据在各组件的逗留时间占比阈值P_a,逗留时间标准差阈值σ_a,组件实例数量增加步长k_a。$t_{batchInterval}$一般取值在0.5～2秒之间。

[0011] 上述方法的特征在于包括以下步骤:

[0012] (1)过程划分

[0013] 将批量流式计算的数据处理过程划分为以下五个阶段:

[0014] ①数据接收:从数据源获取数据并存入系统的内存缓冲区;

[0015] ②数据块构建:周期性将当前缓冲区的数据封装为数据块,放入数据块队列中;

[0016] ③数据块存储:将队列中的数据块存入内存或磁盘中,记录该数据块的元数据信息,并将该数据块的标识ID放入对应的数据流队列中;

[0017] ④作业生成:根据设置批处理间隔,从数据流队列中提取所有未处理数据块,并根据数据的处理逻辑关系生成作业,提交到作业队列中;

[0018] ⑤作业执行:将作业队列中的作业转化为并行任务,并分发给集群的任务执行器执行;

[0019] (2)组件选取

[0020] 2.1)根据系统结构,选取步骤(1)中划分阶段所对应的核心组件,构建备选组件集合C,$C=\{c_i|1\leqslant i\leqslant 5\}$,设置随机组件集合$C_R$,$C_R=\varnothing$,以及常量组件集合$C_c$,$C_c=\varnothing$;其中,随机组件是指数据项在该组件的逗留时间具有随机性的组件,常量组件是指数据项在该组件的逗留时间近似为固定值的组件。

[0021] 2.2)选取测试数据集$D=\{d_j|1\leqslant j\leqslant n\}$,并选取低、中、高三类不同的数据到达强度,构建数据到达强度集λ,$\lambda=\{\lambda_m|1\leqslant m\leqslant p\}$;

[0022] 2.3)运行批量流式计算系统,并逐一按照λ中的到达强度向系统注入数据集D。对每个数据到达强度λ_m,$\lambda_m\in\lambda$,利用公式(1)计算该数据到达强度下数据项d_j,$d_j\in D$,在C中组件c_i,$c_i\in C$的逗留时间T_{ijm};

[0023] $$T_{ijm}=T_{ifinish}-T_{istart} \quad (1)$$

[0024] 其中，$T_{ifinish}$表示数据项d_j离开组件c_i的时间，T_{istart}表示数据项d_j到达组件c_i的时间；

[0025] 2.4)对于λ中的每个数据到达强度λ_m，利用公式(2)计算该强度下数据项在组件c_i的平均逗留时间T_{im}；

[0026] $$T_{im}=\frac{\sum_{j=1}^{n}T_{ijm}}{n} \tag{2}$$

[0027] 2.5)对于λ中的每个数据到达强度λ_m，利用公式(3)和公式(4)计算该强度下数据项在各组件c_i的平均逗留时间占比P_{im}和标准差σ_{im}；

[0028] $$P_{im}=\frac{T_{im}}{T_m} \tag{3}$$

[0029] $$\sigma_{im}=\sqrt{\frac{\sum_{j=1}^{n}(T_{ijm}-T_{im})^2}{n}} \tag{4}$$

[0030] 其中T_m为在数据到达强度λ_m下数据项在系统中的平均总逗留时间，即数据项在各组件平均逗留时间的总和；由公式(5)计算得到，

[0031] $$T_m=\sum_{i=1}^{5}T_{im} \tag{5}$$

[0032] 2.6)利用公式(6)和公式(7)计算不同强度下数据项在各组件c_i的逗留时间的平均占比P_i和标准差σ_i；

[0033] $$P_i=\frac{\sum_{m=1}^{p}P_{im}}{p} \tag{6}$$

[0034] $$\sigma_i=\frac{\sum_{m=1}^{p}\sigma_{im}}{p} \tag{7}$$

[0035] 2.7)对于C中每一个组件c_i，若$P_i\geq P_a$且$\sigma_i\geq\sigma_a$，则将组件c_i加入随机集合C_R，若$P_i\geq P_a$且$\sigma_i<\sigma_a$，则将组件c_i加入常量集合C_c；

[0036] (3)性能建模

[0037] 3.1)对于任一$c_i\in C_c$，数据在该组件c_i的逗留时间设为常量T_i；

[0038] 3.2)对于任一$c_i\in C_R$，则根据组件c_i的数据到达及服务特征，选取相应排队模型，并根据对应模型计算出数据在该组件的等待时间W_{qi}和服务时间S_i，数据在在该组件的逗留时间为$T_i=W_{qi}+S_i$；

[0039] 3.2.1)参考步骤(1)中的过程划分，如果组件属于第①阶段，则数据以λ_0的泊松过程到达，视为单个服务台或多个服务台(根据应用的配置决定)的M/M/1排队模型，利用公式(8)和公式(9)计算等待时间W_{qi}和组件服务时间S_i；

[0040] $$W_{qi}=\begin{cases}\dfrac{\lambda_i}{\mu_i(\mu_i-\lambda_i)} & k_i>\dfrac{\lambda_i}{\mu_i}\\ +\infty & k_i\le\dfrac{\lambda_i}{\mu_i}\end{cases} \tag{8}$$

[0041] $$S_i=\frac{1}{\mu_i} \tag{9}$$

[0042] 其中，λ_i表示该组件的数据到达速率，μ_i表示该组件的服务速率，k_i表示服务台数目；

[0043] 3.2.2)参考步骤(1)中的过程划分，如果组件属于第②③④阶段，则数据在固定时间间隔到达，视为单个服务台/多个服务台的D/M/1排队系统，利用公式(10)计算等待时间W_{qi}，服务时间S_i可利用3.2.1)中的公式(9)计算得到；

[0044] $$W_{qi}=\begin{cases}\dfrac{1}{\mu_i}\cdot\dfrac{\delta}{1-\delta} & k_i>\dfrac{\lambda_i}{\mu_i}\\ +\infty & k_i\le\dfrac{\lambda_i}{\mu_i}\end{cases} \tag{10}$$

[0045] 其中，δ是方程$\delta=e^{-\mu/\lambda(1-\delta)}$的最小绝对值根；

[0046] 3.2.3)参考步骤(1)中的过程划分，如果组件属于第⑤阶段，则其服务特征满足一个作业被分为多个子任务并行处理，并且每个子任务需等到其他所以子任务被处理结束才能重新聚合并离开时，视为Fork-Join模型，利用公式(11)计算服务时间S_i；

[0047] $S_i=\max(T_{l0},T_{l1},\ldots,T_{lk})$ (11)

[0048] 其中，T_{lk}表示第k个并行子任务的执行时间；

[0049] 3.3)将数据项在各关键组件的逗留时间加总即得系统的平均数据处理延迟，如公式(12)：

[0050] $$T=\sum_{i=1}^{u}T_i \tag{12}$$

[0051] (4)延迟计算：在批量流式计算系统运行过程中，周期性根据公式(12)计算系统在线数据处理延迟。

[0052] 4.1)计算每个组件i的数据到达速率λ_i、服务速率μ_i和服务台数量k_i，其中，k_i为当前可使用的组件实例数量；

[0053] 4.1.1)若组件$c_i\in C_R$，且数据到达和服务特征满足M/M/1模型时，其到达速率和服务速率可利用公式(13)和公式(14)计算得到；

[0054] $$\lambda_i=\frac{n_{event}}{t} \tag{13}$$

[0055] $$\mu_i=\frac{1}{t_{event}} \tag{14}$$

[0056] 其中，n_{event}表示在最近的时间周期t内到达的数据项总量，t_{event}表示对单个数据项的平均服务时间；

[0057] 4.1.2)若组件$c_i\in C_R$，且数据到达和服务特征满足D/M/1模型时，其到达速率和服务速率可利用公式(15)和公式(16)计算得到；

[0058] $$\lambda_i = \frac{1}{t_{Interval}} \quad (15)$$

[0059] $$\mu_i = \frac{1}{t_{block}} \quad (16)$$

[0060] 其中，$t_{Interval}$表示应用程序中设置的时间间隔，t_{block}表示对单个数据块的平均服务时间；

[0061] 4.1.3)若组件$c_i \in C_R$，且服务特征满足Fork-Join模型时，其到达速率和服务速率可利用4.1.2)中的公式(15)和公式(16)计算得到；

[0062] 4.2)将步骤4.1)中获取的参数值带入步骤3.3)的公式(12)中得到数据处理延迟T；

[0063] 4.3)比较数据处理延迟T和用户期望数据处理延迟T_{max}，若$T \leqslant T_{max}$，说明数据处理延迟可满足用户时效性需求，则执行步骤(6)，否则执行步骤(5)；

[0064] (5)瓶颈定位及优化

[0065] 5.1)若T趋于无穷大，查找满足$k_i \leq \frac{\lambda_i}{\mu_i}$的组件，将其服务台数目设置为$k_i = \left\lceil \frac{\lambda_i}{\mu_i} \right\rceil$，并跳转至步骤(4)重新计算数据处理延迟T，其中，⌈⌉为上取整符号；

[0066] 5.2)若$T > T_{max}$，则选取逗留时间占比P_i最大的组件，将其服务台数目设置为$k_i = k_i + k_a$，并跳转至步骤(4)重新计算数据处理延迟T；其中，k_a为组件实例数量增加步长；

[0067] (6)回溯：在时间周期t后，判断应用程序是否结束，是则转至步骤(7)，否则转到步骤(4)；其中时间周期t是指相邻两次性能评估之间的时间间隔；

[0068] (7)结束：中止对系统的性能评估及优化。

[0069] 本方法在Spark Streaming系统的每个计算节点上增设信息收集模块，用于实时地收集数据到达组件和离开组件的时间以及各组件数据到达率及组件实例数量，本方法将收集的数据到达和离开组件的时间信息构成步骤(2)中进行组件选取的基础，收集的各组件数据到达率及组件实例数量信息构成步骤(4)中进行延迟计算的依据；为了实现该方法，在Spark Streaming系统中增设资源分配模块，用于根据信息收集模块提供的信息，完成对组件实例数量的增加。

[0070] 性能保障方法依附于现有Spark Streaming批量流式处理系统，通过新增相应的软件模块实现；该平台由多个计算机服务器组成或平台节点组成，服务器间通过网络连接；平台节点分为两类，包括一个管理节点和多个计算节点；所依附的平台包含以下核心软件模块：资源管理模块、节点管理模块、应用管理模块、数据接收模块、数据管理模块和数据计算模块；其中，资源管理模块负责维护平台中所有节点的资源信息，仅在管理节点上部署；节点管理模块负责启动和结束任务执行器，并监控本节点上数据在各组件的执行情况，每个计算节点上均部署一个节点管理模块；应用管理模块负责流应用的依赖解析以及作业的生成与调度，在Spark Streaming平台中提交和运行的每个流应用均对应一个应用管理模块；数据接收模块负责数据项的接收及数据块的构建，数据管理模块负责数据块的存储管理，数据计算模块负责作业的计算；上述软件模块中，资源管理模块和节点管理模块在Spark Streaming系统启动时即部署运行，应用管理模块、数据接收模块、数据管理模块和

数据计算模块分别在相应的流应用提交运行时触发部署运行；

[0071] 为实现本方法须在既有Spark Streaming系统中新增软件模块，包括信息收集模块、资源分配模块和资源更新模块；其中，信息收集模块主要负责对数据接收模块、数据管理模块、应用管理模块及数据计算模块中各组件数据到达率及组件实例数量、数据到达和离开各组件时间的收集，资源分配模块负责根据瓶颈定位及优化方法的判断信息，调整各组件的实例即服务台数量；上述两个模块作为节点管理模块的子模块部署于各计算节点上；资源更新模块作为资源管理模块的子模块，部署于管理节点，负责收集计算节点上的组件实例数量变动信息，并修改维护各计算节点可分配的资源信息。

[0072] 在上述组件选取步骤的执行过程中，本发明需要对批量流式计算系统源码进行插桩，用于获取数据项在各组件的逗留时间，并根据该逗留时间计算各组件的逗留时间占比和标准差，作为步骤(2)中选取组件的依据，其中逗留时间占比越大说明在该组件花费时间越多，标准差越大说明数据在该组件逗留时间的波动性越大，所以选取逗留时间占比和标准差作为组件选取的依据。利用排队论建模方法可合理优化数据由于等待时间长而造成的拥堵等问题，通过对模型进行数学解析，可全面、准确的揭示系统运行规律，优化排队系统的到达间隔、服务台等重要参数，为系统稳定运行奠定基础；本发明能精确地、完善地把数据在批量流式计算系统中的总逗留时间详细的量化表达出来，建立性能模型，为用户定位瓶颈及优化提供了保障。

附图说明

[0073] 图1为本发明方法所依附的批量流式计算平台的部署图。

[0074] 图2为采用本发明方法的批量流式计算平台中新增软件模块及其交互关系图。

[0075] 图3为本发明方法的总体流程图。

[0076] 图4为组件选取流程图。

[0077] 图5为性能建模流程图。

[0078] 图6为瓶颈定位及优化流程图。

[0079] 图7是Spark streaming系统中的数据处理过程图。

[0080] 图8是Spark streaming随机组件的排队模型图。

具体实施方式

[0081] 下面结合附图和具体实施方式对本发明加以说明。

[0082] 本发明结合目前广泛使用的批量流式计算系统Spark Streaming，阐述所提出的性能保障方法具体实施方式。图1是本方法所依附的批量流式计算平台的部署图。该平台由多个计算机服务器(平台节点)组成，服务器间通过网络连接。平台节点分为两类：包括一个管理节点(Master)和多个计算节点(Slave)。本发明所依附的平台包含以下核心软件模块：资源管理模块、节点管理模块、应用管理模块、数据接收模块、数据管理模块和数据计算模块。其中，资源管理模块负责维护平台中所有节点的资源信息，仅在管理节点上部署；节点管理模块负责启动和结束任务执行器，并监控本节点上数据在各组件的执行情况，每个计算节点上均部署一个节点管理模块。应用管理模块负责流应用的依赖解析以及作业的生成与调度，在Spark Streaming平台中提交和运行的每个流应用均对应一个应用管理模块。数

据接收模块负责数据项的接收及数据块的构建，数据管理模块负责数据块的存储管理，数据计算模块负责作业的计算。上述软件模块中，资源管理模块和节点管理模块在Spark Streaming系统启动时即部署运行，应用管理模块、数据接收模块、数据管理模块和数据计算模块分别在相应的流应用提交运行时触发部署运行。

[0083] 图2是为实施本发明方法在所依附的Spark Streaming系统中需增加的软件模块及其交互关系图。阴影模块是为实现本发明方法须在既有Spark Streaming系统中新增的模块，包括信息收集模块、资源分配模块和资源更新模块。其中，信息收集模块主要负责对数据接收模块、数据管理模块、应用管理模块及数据计算模块中各组件数据到达率及组件实例数量、数据到达和离开各组件时间的收集，资源分配模块负责根据瓶颈定位及优化方法的判断信息，调整各组件的实例(服务台)数量。上述两个模块作为节点管理模块的子模块部署于各计算节点上。资源更新模块作为资源管理模块的子模块，部署于管理节点，负责收集计算节点上的组件实例数量变动信息，并修改维护各计算节点可分配的资源信息。

[0084] 下面结合图3发明总流程说明本发明方法的具体实施方法。在本实施方法中，基本参数的设置如下：数据块间隔$t_{blockInterval}$＝0.2s、批处理间隔$t_{batchInterval}$＝2s、用户期望数据处理时间T_{max}＝2s、在线计算数据处理延迟周期t＝60s、数据在各组件的逗留时间占比阈值P_a＝10％、逗留时间标准差阈值σ_a＝10、组件实例数量增加步长k_a＝1。具体实施方法可分为以下步骤：

[0085] (1)过程划分

[0086] 根据Spark Streaming数据处理流程，如图7所示，将Spark Streaming批量流式计算的数据处理过程划分为以下五个阶段：

[0087] ①数据接收：从数据源获取数据并存入系统的内存缓冲区；

[0088] ②数据块构建：周期性将当前缓冲区的数据封装为数据块，放入数据块队列中；

[0089] ③数据块存储：将队列中的数据块存入内存或磁盘中，记录该数据块的元数据信息，并将该数据块的标识ID放入对应的数据流队列中；

[0090] ④作业生成：根据设置批处理间隔，从数据流队列中提取所有未处理数据块，并根据数据的处理逻辑关系生成作业，提交到作业队列中；

[0091] ⑤作业执行：将作业队列中的作业转化为并行任务，并分发给集群的任务执行器执行；

[0092] (2)组件选取

[0093] 2.1)根据系统结构，选取步骤(1)中划分阶段所对应的核心组件，构建备选组件集合C，$C=\{c_i|1\leqslant i\leqslant 5\}$，其中，核心组件分别为数据接收器(Receiver，数据接收模块内)、数据块生成器(Block Generator，数据接收模块内)、数据块管理器(Block Manager，数据管理模块内)、作业生成器(Job Generator，应用管理模块内)、任务执行器(Executor，数据计算模块内)；随机组件集合C_R，$C_R=\varnothing$，以及常量组件集合C_c，$C_c=\varnothing$；

[0094] 2.2)选取测试数据集$D=\{d_j|1\leqslant j\leqslant n\}$，并选取低、中、高三类不同的数据到达强度，构建数据到达强度集λ，$\lambda=\{\lambda_m|1\leqslant m\leqslant p\}$，选取数据到达强度分别为$\lambda_1$＝2000events/s，$\lambda_2$＝6000events/s，$\lambda_3$＝10000events/s；

[0095] 2.3)运行批量流式计算系统，并逐一按照λ中的到达强度向系统注入数据集D。对每个数据到达强度λ_m，$\lambda_m\in\lambda$，利用公式(1)计算该数据到达强度下数据项d_j，$d_j\in D$，在C中组

件c_i，$c_i \in C$的逗留时间T_{ijm}；

[0096] $T_{ijm}=T_{ifinish}-T_{istart}$ (1)

[0097] 其中，$T_{ifinish}$表示数据项d_j离开组件c_i的时间，T_{istart}表示数据项d_j到达组件c_i的时间；

[0098] 2.4)对于λ中的每个数据到达强度λ_m，利用公式(2)计算该强度下数据项在任一组件c_i的平均逗留时间T_{im}；

[0099] $$T_{im}=\frac{\sum_{j=1}^{n}T_{ijm}}{n} \tag{2}$$

[0100] 2.5)对于λ中的每个数据到达强度λ_m，利用公式(3)和公式(4)计算该强度下数据项在各组件c_i的平均逗留时间占比P_{im}和标准差σ_{im}；

[0101] $$P_{im}=\frac{T_{im}}{T_m} \tag{3}$$

[0102] $$\sigma_{im}=\sqrt{\frac{\sum_{j=1}^{n}(T_{ijm}-T_{im})^2}{n}} \tag{4}$$

[0103] 其中T_m为在数据到达强度λ_m下数据项在系统中的平均总逗留时间，即数据项在各组件平均逗留时间的总和；由公式(5)计算得到，

[0104] $$T_m=\sum_{i=1}^{5}T_{im} \tag{5}$$

[0105] 在λ_1，λ_2和λ_3三个负载强度下，各组件的平均逗留时间计算结果如表1所示。

[0106] 表1各组件在不同负载强度下的数据平均逗留时间

[0107]

组件编号	负载强度λ_1（ms）	负载强度λ_1（ms）	负载强度λ_1（ms）
c_1	27.76	54.28	83.28
c_2	0.62	0.83	0.94
c_3	42.99	63.86	70.80
c_4	26.55	29.40	26.91

[0108]

c_5	191.89	242.41	343.60

[0109] 2.6)利用公式(6)和公式(7)计算不同强度下数据项在各组件c_i的逗留时间的平均占比P_i和标准差σ_i；

[0110] $$P_i=\frac{\sum_{m=1}^{p}P_{im}}{p} \tag{6}$$

[0111] $$\sigma_i = \frac{\sum_{m=1}^{p} \sigma_{im}}{p} \tag{7}$$

[0112] 各组件逗留时间平均占比P_i和标准差σ_i计算结果如表2所示。

[0113] 表2各组件逗留时间平均占比和标准差

[0114]

组件编号	逗留时间占比P_i	标准差σ_i
c_1	13.10%	20.56
c_2	0.20%	0.85
c_3	14.89%	33.17
c_4	7.27%	6.35
c_5	65.54%	64.95

[0115] 2.6)对于C中每一个组件c_i,若$P_i \geq P_a$且$\sigma_i \geq \sigma_a$,则将组件c_i加入随机集合C_R,若$P_i \geq P_a$且$\sigma_i < \sigma_a$,则将组件c_i加入常量集合C_c;此时生成随机组件集合$C_R = \{c_1, c_3, c_5\}$,常量组件集合为空;

[0116] (3)性能建模

[0117] 3.1)对于任一$c_i \in C_c$,数据在该组件c_i的逗留时间为常量T_i;此时,常量集合C_c为空,没有组件$C_i \in C_c$;

[0118] 3.2)对于$c_1, c_3, c_5 \in C_R$,则根据组件c_i的数据到达及服务特征,选取相应排队模型,并根据对应模型计算出数据在该组件的等待时间W_{qi}和服务时间S_i,数据在在该组件的逗留时间为$T_i = W_{qi} + S_i$;

[0119] 3.2.1)组件c_1属于第①阶段,则数据以λ_0的泊松过程到达,可看做单个服务台的M/M/1排队模型,利用公式(8)和公式(9)计算等待时间W_{qi}和服务时间S_i;

[0120] $$W_{qi} = \begin{cases} \dfrac{\lambda_i}{\mu_i(\mu_i - \lambda_i)} & k_i > \dfrac{\lambda_i}{\mu_i} \\ +\infty & k_i \leq \dfrac{\lambda_i}{\mu_i} \end{cases} \tag{8}$$

[0121] $$S_i = \frac{1}{\mu_i} \tag{9}$$

[0122] 其中,λ_i表示该组件的数据到达速率,μ_i表示该组件的服务速率,k_i表示服务台数目;

[0123] 3.2.2)组件c_3属于第③阶段,则数据在固定数据块间隔到达,可看做单个服务台的D/M/1排队系统,利用公式(10)计算等待时间W_{qi},服务时间S_i可利用3.2.1)中的公式(9)计算得到;

[0124] $$W_{qi} = \begin{cases} \dfrac{1}{\mu_i} \cdot \dfrac{\delta}{1-\delta} & k_i > \dfrac{\lambda_i}{\mu_i} \\ +\infty & k_i \leq \dfrac{\lambda_i}{\mu_i} \end{cases} \tag{10}$$

[0125] 其中，δ是方程$\delta=e^{-\mu/\lambda(1-\delta)}$的最小绝对值根；

[0126] 3.2.3)组件c_5属于第⑤阶段，则其服务特征满足一个作业被分为多个子任务并行处理，并且每个子任务需等到其他所以子任务被处理结束才能重新聚合并离开时，可看做Fork-Join模型，利用公式(11)计算服务时间S_i；

[0127] $S_i=\max(T_{l0},T_{l1},\ldots,T_{lk})$ (11)

[0128] 其中，T_{lk}表示第k个并行子任务的执行时间；

[0129] 3.3)根据Spark Streaming基本原理及各随机组件之间的关系，得出随机组件的排队模型图，如图8所示，将数据项在各关键组件的逗留时间加总即得系统的平均数据处理延迟，如公式(12)：

[0130] $$T=\sum_{i=1}^{3}T_i \tag{12}$$

[0131] (4)延迟计算：在批量流式计算系统运行过程中，周期性根据公式(12)计算系统在线数据处理延迟。

[0132] 4.1)根据发明内容4.1)中的方法，由当前可使用的组件实例数量和应用的配置参数，得$\lambda_1=10052$events/s，$\mu_1=9008$events/s，$k_1=1$；$\lambda_3=5$，$\mu_3=9.8$，$\delta_3=0.1$；$\lambda_5=0.5$，$\mu_5=1$，$\delta_5=0.7$，$k_5=4$；

[0133] 4.2)将步骤4.1)中获取的参数值带入步骤3.3)的公式中得到总逗留时间$T=+\infty$，说明存在服务速率小于数据到达速率的组件，数据在队列中的等待时间不短增加；

[0134] 4.3)此时，不满足$T\leq T_{max}$，执行步骤(5)；

[0135] (5)瓶颈定位及优化

[0136] 5.1)若T趋于无穷大，查找满足$k_i\leq\frac{\lambda_i}{\mu_i}$的组件，将其服务台数目设置为$k_i=\left\lceil\frac{\lambda_i}{\mu_i}\right\rceil$；此时，满足$k_i\leq\frac{\lambda_i}{\mu_i}$的组件为$c_1$，设置$k_1=\left\lceil\frac{10052}{9008}\right\rceil=2$，并跳转至步骤(4)重新计算数据处理延迟T；

[0137] 5.2)若$T>T_{max}$，则选取随机集合中逗留时间占比P_i最大的组件，将其服务台数目设置为$k_i=k_i+k_a$，并跳转至步骤(4)重新计算数据处理延迟T；其中，k_a为服务台默认的增加数值；

[0138] (6)回溯：在时间周期t后，判断应用程序是否结束，是则转至步骤(7)，否则转到步骤(4)；其中时间周期t是指相邻两次性能评估之间的时间间隔；

[0139] (7)结束：中止对系统的性能评估及优化。

[0140] 根据本发明所提出的性能保障方法，发明人对建模方法和瓶颈定位及优化做了相关的测试。验证结果表明，本发明方法可适用于典型流应用负载。采用本发明方法的批量流式计算系统，如Spark Streaming，可较好的保障系统性能。

[0141] 测试以数据在系统中的逗留时间为指标，体现本发明提出的性能建模方法的正确性，瓶颈定位及优化的有效性。性能测试运行于7个节点构成的集群系统，节点的硬件配置包括：Intel(R)Xeon(R)CPU E5-2660 0@2.2.GHz的CPU、16GB DDR3 RAM、1TB SATA硬盘，节点间采用千兆以太网互连，操作系统为Centos6.5。实验选用Hibench作为负载发生器，选取其中常见的Word Count应用为例进行测试。Word Count应用的到达间隔符合泊松分布，到

达强度为平均每秒发送6000个记录，即λ_0＝6000records/s；实验设置的任务执行器的数目为4。

[0142] 针对性能建模方法的测试

[0143] 通过发明内容所述方法计算数据在各组件的等待时间W_{qi}和服务时间S_i，得出数据在系统中的数据处理延迟理论值T，同时测量出数据处理延迟的实际值，本实验选取应用开始后2-2.5分钟，2.5-3分钟，3-3.5分钟，3.5-4分钟，4-4.5分钟，4.5-5分钟，5-5.5分钟七个时间段，分别表示为实验序列1、2、3、4、5、6、7，其结果如表3所示。

[0144] 表3实际与理论数据处理延迟的计算结果

[0145]

实验序列	1	2	3	4	5	6	7
实际数据处	453	458	453	439	443	458	448

[0146]

理延迟（ms）							
理论数据处理延迟（ms）	435.364	435.364	435.364	435.364	435.364	435.364	435.364

[0147] 从以上实验结果中可以看出，理论总逗留时间和实际总逗留时间基本吻合，其中理论时间小于实际时间，这是因为所建的模型是基于关键组件，计算理论时间时没有考虑到时间占比较小的组件。

[0148] 针对瓶颈定位及优化方法的测试

[0149] 该部分测试为Word Count应用设置了用户期望响应延迟T_{max}＝1s，分析了不同负载强度下应用的性能。实验结果如表4所示。

[0150] 表4在线测量与优化后的数据处理延迟计算结果

[0151]

实验序列	1	2	3	4	5	6	7
在线测量的数据处理延迟（ms）	1050	1038	1074	1083	1029	1039	1048
优化后的数据处理延迟（ms）	921	958	936	911	883	939	986

[0152] 以上实验结果中可以看出，在线计算的数据处理延迟大于用户期望的相应延迟，通过本发明提出的瓶颈定位及优化后，所得数据处理延迟已满足用户期望的响应延迟，且最低缩短5.92％，最高缩短了15.88％，平均缩短了11.20％。实验结果表明，瓶颈定位及优化技术可为批量流式计算系统提供性能保障。

[0153] 以上实施例仅用以说明本发明，而并非限制本发明所描述的技术方案。因此，一切不脱离本发明的精神和范围的技术方案及其改进，均应涵盖在本发明的权利要求范围当中。

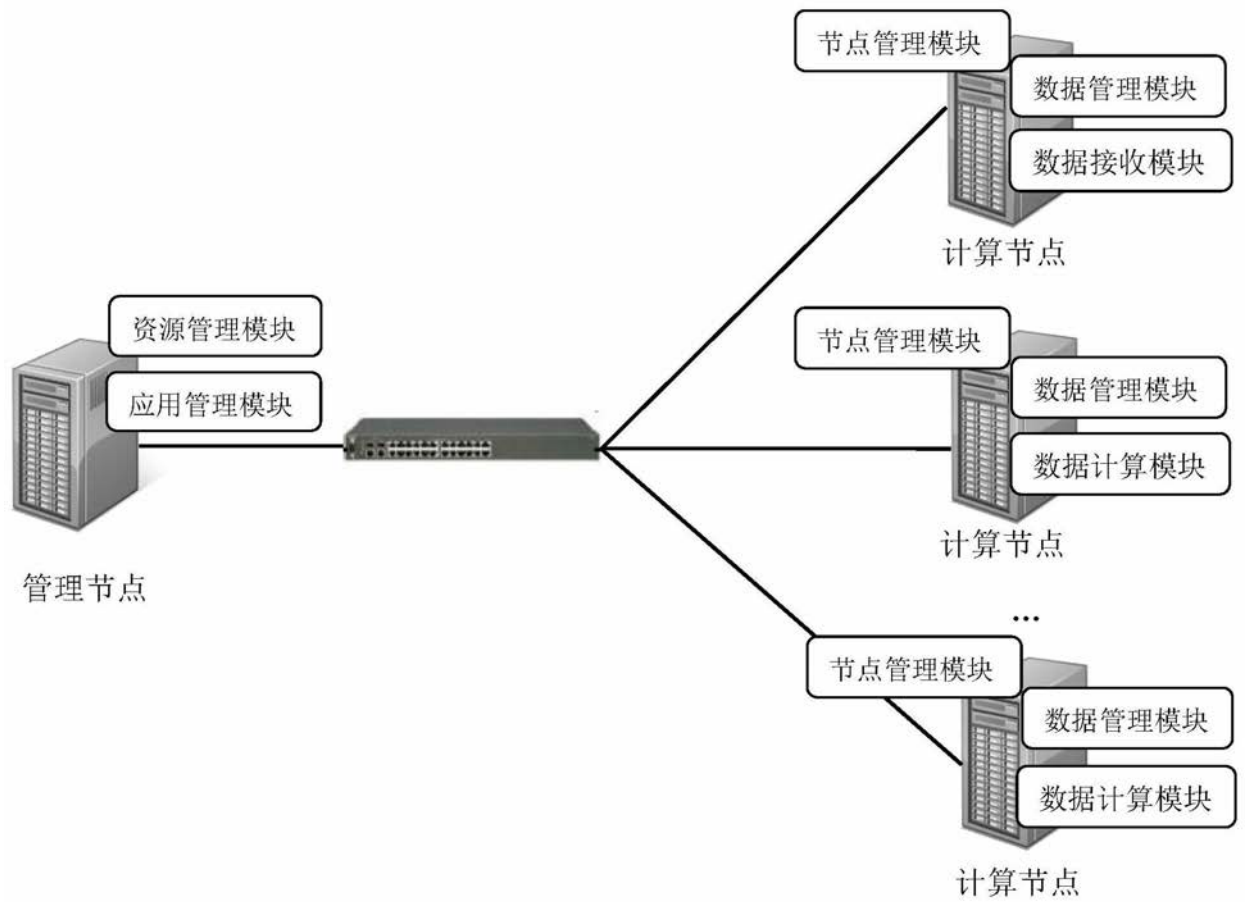

节点管理模块
数据管理模块
数据接收模块
计算节点
资源管理模块
应用管理模块
管理节点
节点管理模块
数据管理模块
数据计算模块
计算节点
...
节点管理模块
数据管理模块
数据计算模块
计算节点

图1

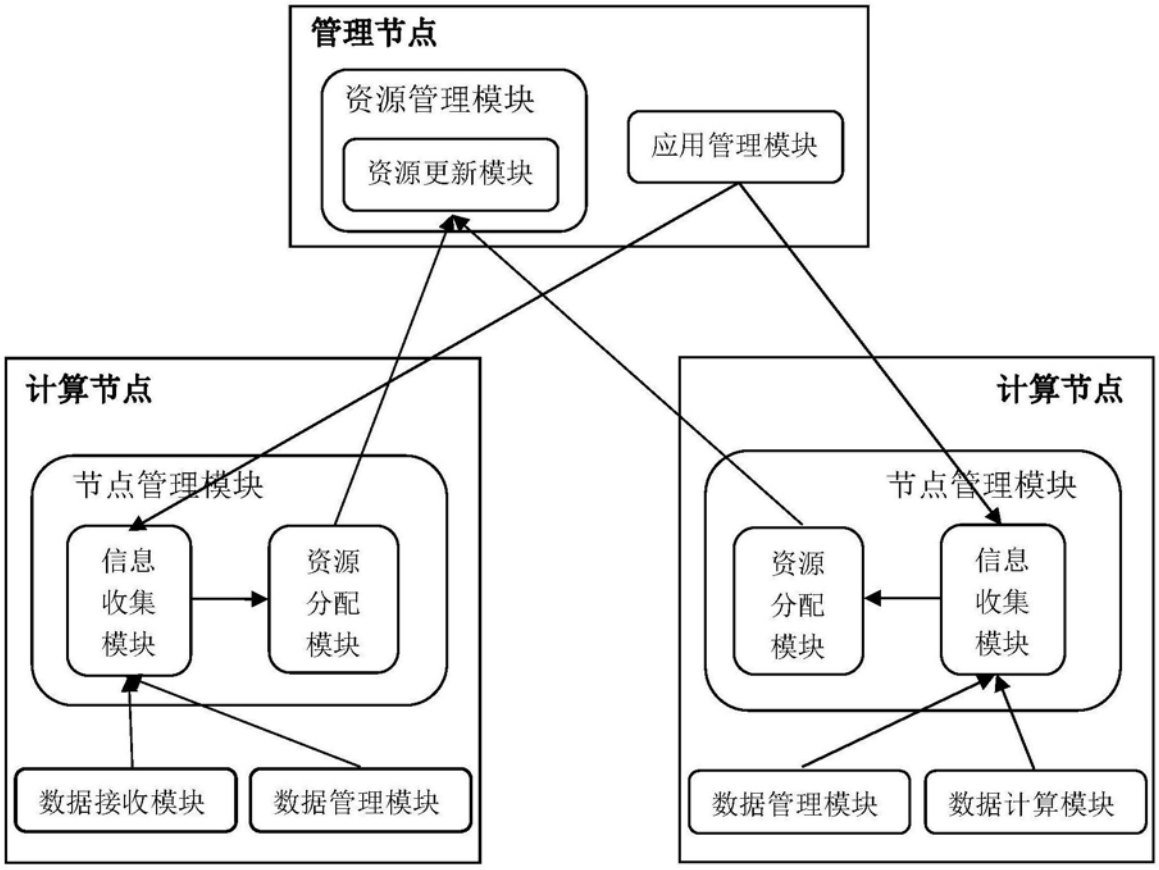

管理节点
资源管理模块
资源更新模块
应用管理模块
计算节点
节点管理模块
信息
收集
模块
资源
分配
模块
数据接收模块
数据管理模块
计算节点
节点管理模块
资源
分配
模块
信息
收集
模块
数据管理模块
数据计算模块

图2

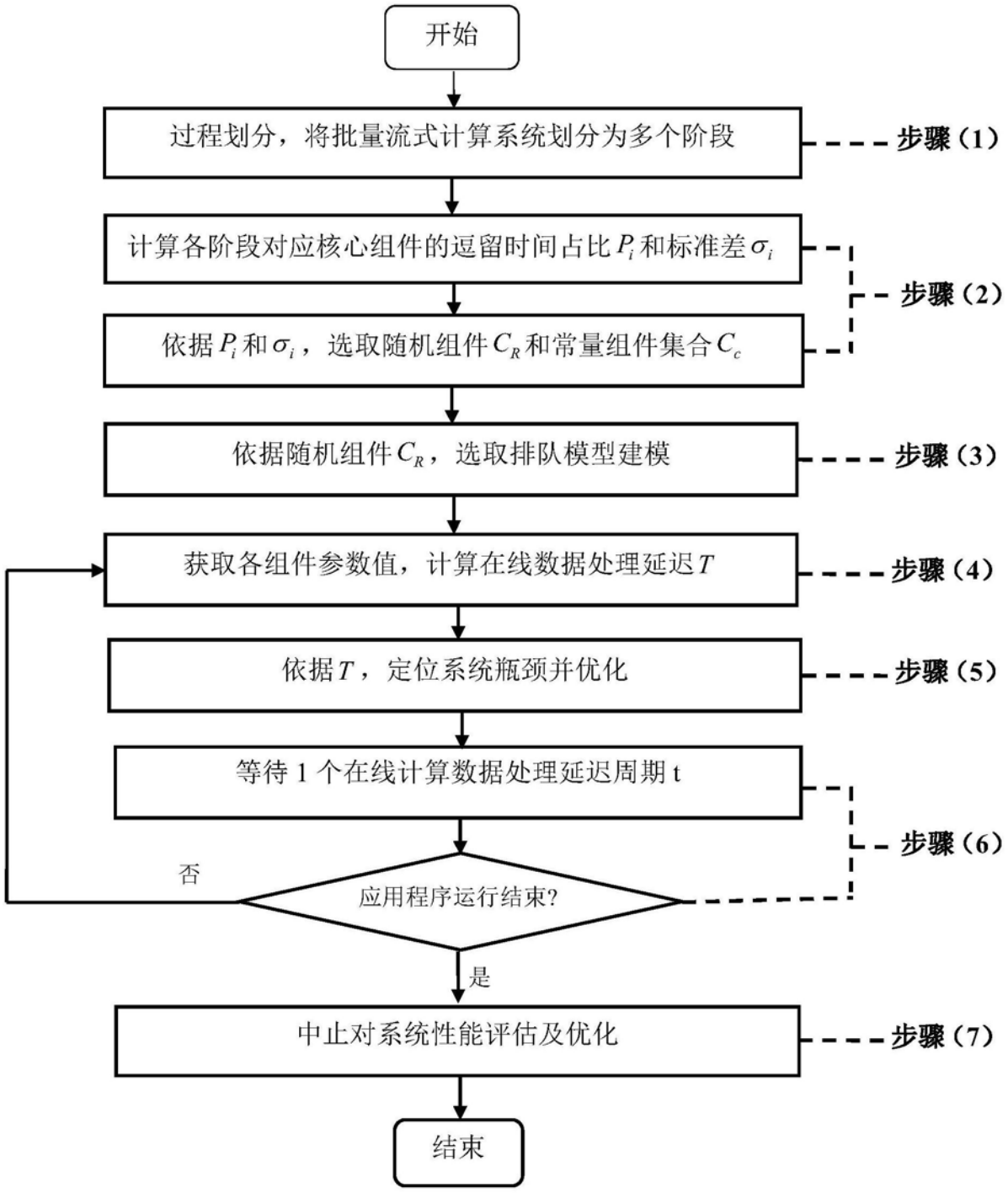

开始
过程划分，将批量流式计算系统划分为多个阶段
步骤（1）
计算各阶段对应核心组件的逗留时间占比P_i和标准差σ_i
步骤（2）
依据P_i和σ_i，选取随机组件C_R和常量组件集合C_c
依据随机组件C_R，选取排队模型建模
步骤（3）
获取各组件参数值，计算在线数据处理延迟T
步骤（4）
依据T，定位系统瓶颈并优化
步骤（5）
等待 1 个在线计算数据处理延迟周期 t
步骤（6）
否
应用程序运行结束?
是
中止对系统性能评估及优化
步骤（7）
结束

图3

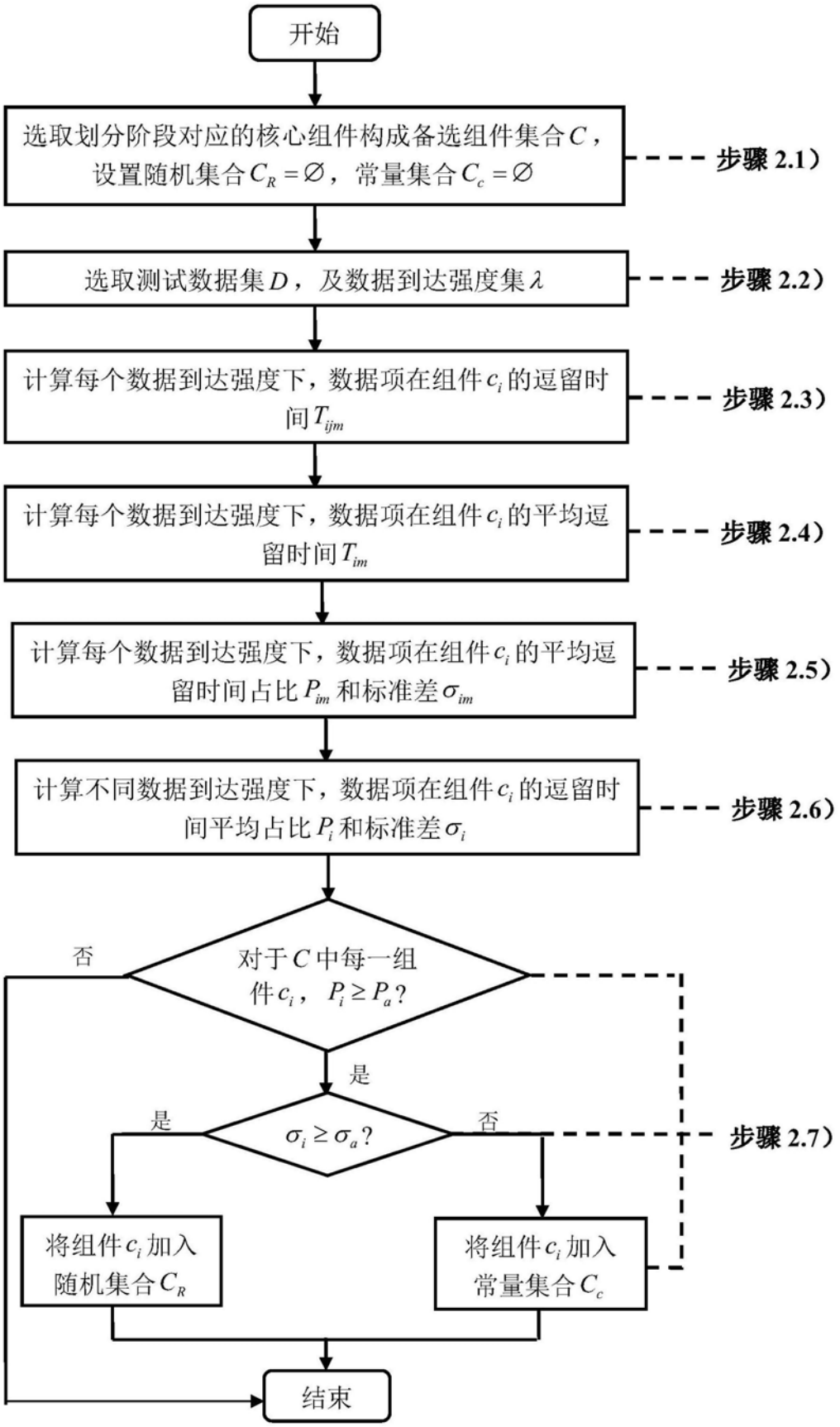

开始
选取划分阶段对应的核心组件构成备选组件集合C，设置随机集合$C_R=\varnothing$，常量集合$C_c=\varnothing$
步骤 2.1）
选取测试数据集D，及数据到达强度集λ
步骤 2.2）
计算每个数据到达强度下，数据项在组件c_i的逗留时间T_{ijm}
步骤 2.3）
计算每个数据到达强度下，数据项在组件c_i的平均逗留时间T_{im}
步骤 2.4）
计算每个数据到达强度下，数据项在组件c_i的平均逗留时间占比P_{im}和标准差σ_{im}
步骤 2.5）
计算不同数据到达强度下，数据项在组件c_i的逗留时间平均占比P_i和标准差σ_i
步骤 2.6）
对于C中每一组件c_i，$P_i \geq P_a$?
否
是
$\sigma_i \geq \sigma_a$?
是
否
步骤 2.7）
将组件c_i加入随机集合C_R
将组件c_i加入常量集合C_c
结束

图4

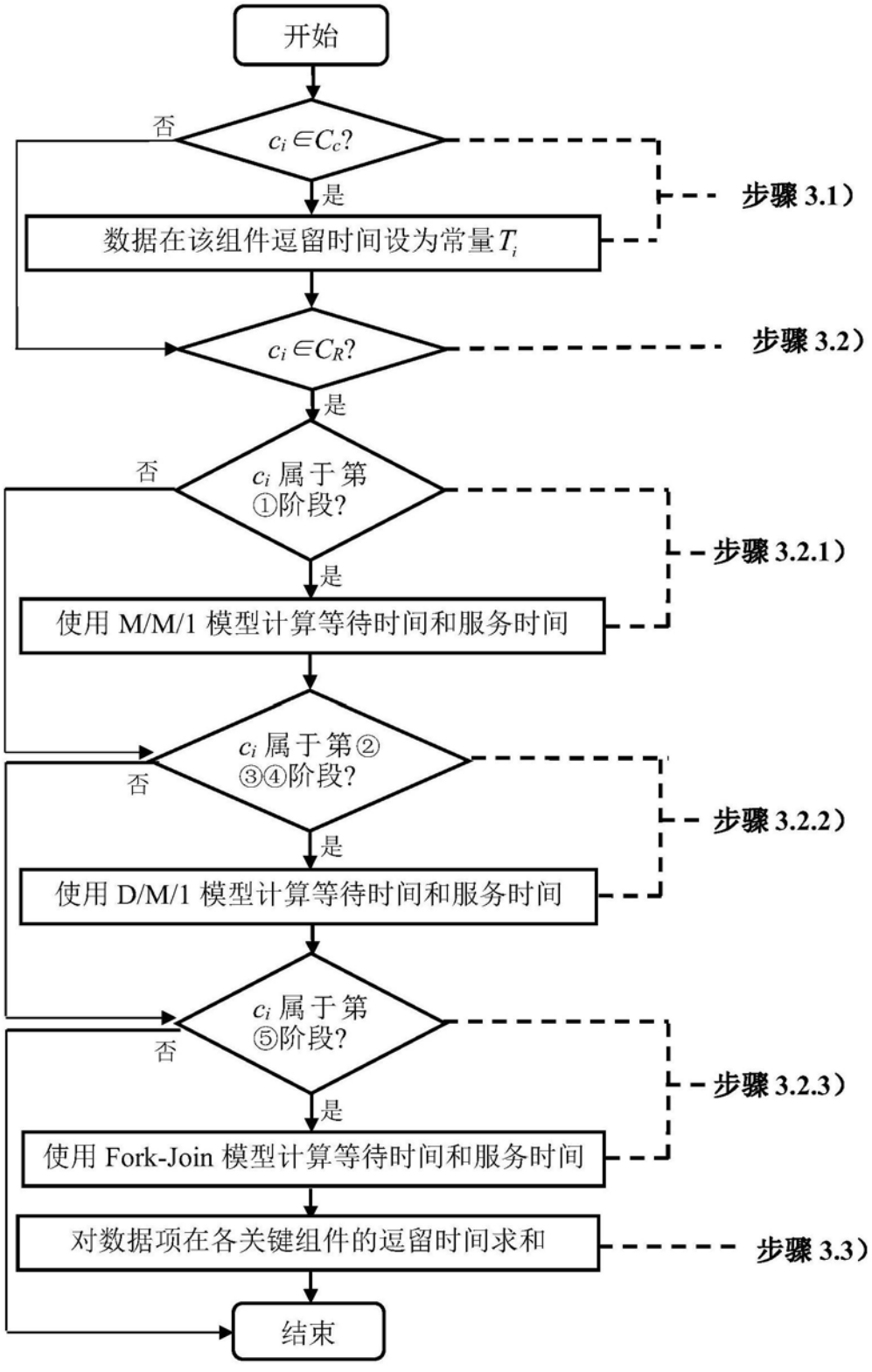

开始
$c_i \in C_c$?
否
是
步骤 3.1）
数据在该组件逗留时间设为常量 T_i
$c_i \in C_R$?
步骤 3.2）
是
c_i 属于第①阶段?
否
步骤 3.2.1）
是
使用 M/M/1 模型计算等待时间和服务时间
c_i 属于第②③④阶段?
否
步骤 3.2.2）
是
使用 D/M/1 模型计算等待时间和服务时间
c_i 属于第⑤阶段?
否
步骤 3.2.3）
是
使用 Fork-Join 模型计算等待时间和服务时间
对数据项在各关键组件的逗留时间求和
步骤 3.3）
结束

图5

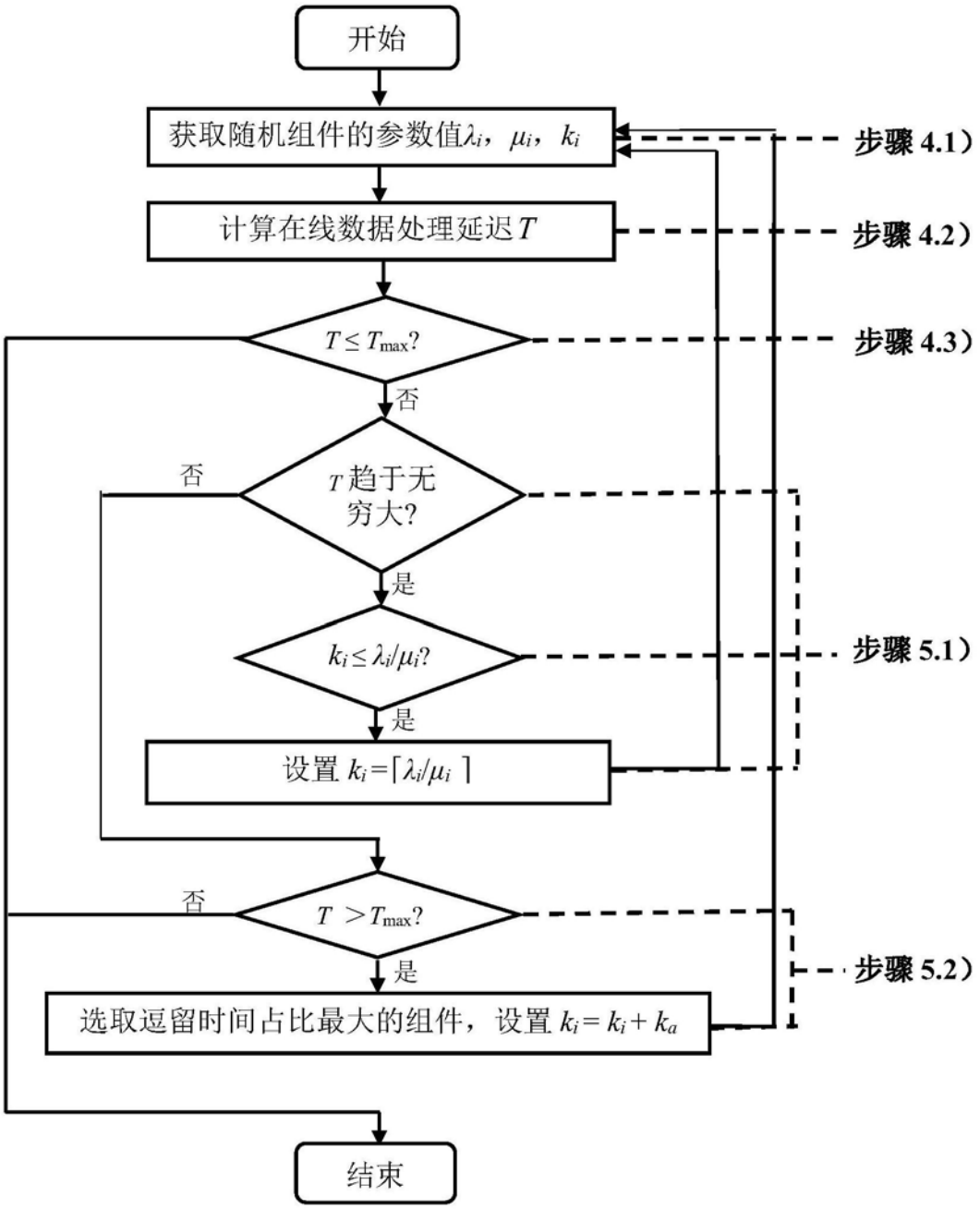

开始
获取随机组件的参数值λ_i，μ_i，k_i
计算在线数据处理延迟T
$T \le T_{max}$?
否
T趋于无穷大?
否
是
$k_i \le \lambda_i/\mu_i$?
是
设置 $k_i=\lceil \lambda_i/\mu_i \rceil$
$T > T_{max}$?
否
是
选取逗留时间占比最大的组件，设置 $k_i = k_i + k_a$
结束
步骤 4.1）
步骤 4.2）
步骤 4.3）
步骤 5.1）
步骤 5.2）

图6

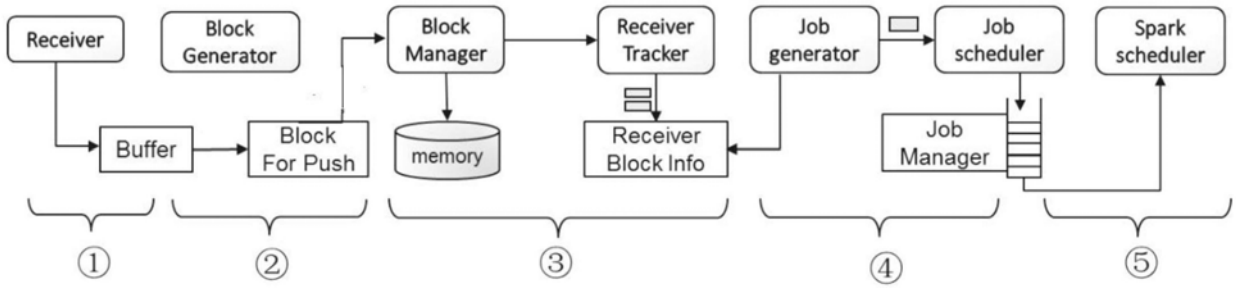

Receiver
Block Generator
Block Manager
Receiver Tracker
Job generator
Job scheduler
Spark scheduler
Buffer
Block For Push
memory
Receiver Block Info
Job Manager
①
②
③
④
⑤

图7

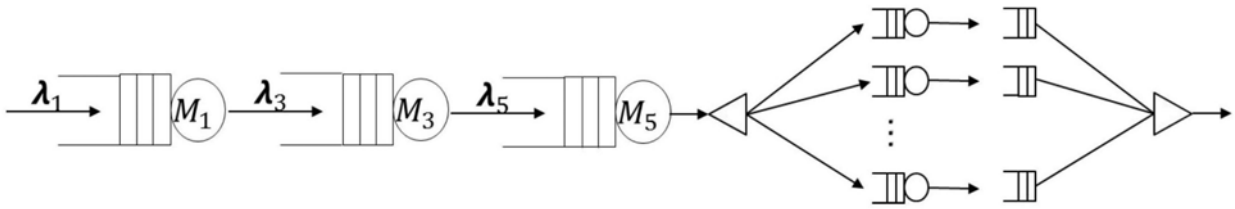

λ_1
M_1
λ_3
M_3
λ_5
M_5

图8