·云计算专题· 文章编号:1000-3428(2017)12-0030-08 文献标志码:A 中图分类号:TP391

考虑服务质量的并行 MapReduce 启发式车载云资源调度

罗小波,王 超

(重庆邮电大学 计算机科学与技术学院,重庆 400065)

摘 要: 为提高车载云计算资源调度的可靠性,减少数据处理时间,提出一种服务质量感知的并行 MapReduce 启发式车载云资源调度算法。在 MapReduce 并行计算模型的基础上,设计云计算环境中以车载单元为基础的车辆并行检测服务框架,利用相对优先级因子构建车载云计算调度模型,并通过启发式并行优化算法对模型进行优化,降低算法复杂度。在 NS-3 中的仿真结果表明,该算法可有效缩短作业执行时间,并具有较高的可靠性。

关键词: 服务质量;并行云计算;MapReduce 模型;车载云资源;启发式调度算法

中文引用格式: 罗小波,王 超. 考虑服务质量的并行 MapReduce 启发式车载云资源调度[J]. 计算机工程,2017,43(12):30-37.

英文引用格式: LUO Xiaobo, WANG Chao. Parallel MapReduce Heuristics On-board Cloud Resource Scheduling Considering Quality of Service[J]. Computer Engineering,2017,43(12):30-37.

Parallel MapReduce Heuristics On-board Cloud Resource Scheduling Considering Quality of Service

LUO Xiaobo, WANG Chao

(School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

【Abstract】 In order to improve the reliability of on-board cloud computing resource scheduling and reduce the computation time of data processing, a parallel MapReduce heuristics on-board cloud resource scheduling algorithm with Quality of Service (QoS) perception is proposed. Based on the MapReduce parallel computing model, the On-Board Unit (OBU)-based vehicle parallel detection service framework in cloud computing environment is designed, and the relative priority factor is used to construct the on-board cloud computing scheduling model. Then the cloud resource scheduling model is optimized by using heuristic parallel optimization algorithm to reduce the computational complexity of the proposed algorithm. The simulation results in NS-3 show that the proposed algorithm can shorten the job execution time effectively and has higher reliability.

【Key words】 Quality of Service (QoS); parallel cloud computing; MapReduce model; on-board cloud resource; heuristic scheduling algorithm

DOI: 10.3969/j.issn.1000-3428.2017.12.006

0 概述

目前,道路上行驶的车辆一般携带板载计算机设备,其具有小规模的数据处理能力[1-2]。很多研究致力于通过使用车载 Ad Hoc 网络获得更安全的驾驶体验[3]。随着车辆越来越多地配备先进的传感器,从医疗保健到公路动态拥塞管理等应用程序均得以部署,并依赖于从传感器获得的数据。此外,车载云计算有利于在路面上行驶的车辆以及街道和停车场中的车辆充分利用车载计算资源,提供广泛的、一致可靠的实时服务[4]。

为提高经典交通智能系统通信效率,利用车载调度方式可实现更为高效的数据采集和分发,使数据传输能力得到提升。车载调度方式存在的主要问题是车载物理层的有效利用,包括车辆的无线信道、移动性、相对位置等。文献[5]提出利用 RIU 作为中继的车载数据传输协作调度方式,提高了调度效率。文献[6]提出基于路旁基站的下行车辆调度策略,提高了信道下行资源管理的有效性。文献[7]对调度过程的下行和上行请求同时进行考虑,提高了数据执行效率。文献[8]研究 WAVE 网和 WiMAX 网中调度策略的差异,设计了基于反馈机制的车载

基金项目: 国家自然科学基金(61272195);重庆市教委科学技术研究项目(KJ12057,KJ1402801)。

作者简介: 罗小波(1976—),男,副教授、博士,主研方向为云计算、遥感信息反演;王 超,硕士研究生。

收稿日期: 2016-08-19 **修回日期:** 2016-10-29 **E-mail:** wangchaorrn@qq.com

云资源调度策略。文献[9]则提出基于转发放大化的 Ad Hoc 自组织车载云资源网络调度方法。上述文献采取的是集中式模型研究方法，在处理时间敏感数据时均未考虑工作执行的可靠性。同时，当中央通信和管理基础设施出现故障时，过于集中的系统运作模式无法提供服务，而对于传感器收集的数据，在较短时间内实现分发处理至关重要。

针对上述问题，本文构建一个独立的车载云计算系统架构，设计服务质量（Quality of Service，QoS）感知的任务调度系统。假设道路上具有较低相对速度的一组车辆为一个车辆集群，每个集群含有一个簇头（Cluster Head，CH）车辆，首先利用 MapReduce 计算模型[10-11]进行数据的快速和分布式处理，使 CH 节点车辆获得集群中每个车辆节点完成任务的时间和可靠性；然后通过混合整数线性规划（Mixed Integer Linear Programming，MILP）实现应用程序任务与工作节点车辆的最优映射；最后设计一种启发式任务调度算法，以减少时间复杂度，便于实际应用程序开发。

1　车载云计算模型

本文考虑由附近车辆车载单元（On-board Unit，OBU）构建的车载云计算环境下的车辆检测和计算服务。在此项研究中，允许一组车辆在同一个方向上以非常低的相对速度移动，从而形成一个车辆集群。簇头（CH）车辆负责管理执行 MapReduce 分布式计算模型成员节点的任务调度。车辆在道路上高速移动，将会导致拓扑结构快速变化，进而造成通信的不稳定以及较高的拓扑信息交换开销。为优化车辆之间的数据路由，并增加车载云计算的 QoS，将同一移动方向上的车辆进行聚集。CH 的责任是对集群成员进行管理，对成员和媒体的访问进行控制，并对任务进行分解，最后将结果返回给请求者。

在每个集群中，使用 MapReduce 大数据处理模型进行任务处理。MapReduce 是一种对大量原始数据进行分布式处理的开源计算框架。MapReduce 模型可实现计算的并行简化，在工作分配和机器可靠性方面可帮助车载云计算应用开发者专注于特定的业务目标实现。MapReduce 是去除相关性无共享并行执行的批量处理模型，其包含了 Map 和 Reduce 函数。Map 函数对输入任务执行预处理，并生成（*key*，*value*）对中间结果[12-13]：

$$Input\_task(x_i)\xrightarrow{\text{Map}}ilist(key,value) \tag{1}$$

Map 函数将输入数据进行分类，使数据可通过 Reduce 函数进行处理。Reduce 函数对中间列表（*key*，*value*）对进行操作，并产生最终的输出：

$$ilist(key,value)\xrightarrow{\text{Reduce}}Output(y_i) \tag{2}$$

基于 MapReduce 模型的车载云计算框架如图 1 所示。其中，CH 执行 MapReduce 中管理者的角色，成员节点将他们的作业执行请求发送到该 CH，将作业划分为多个 Map 任务，将 Map 任务输出传递给另一组资源，并执行 Reduce 任务。模型在得到 Reduce 任务的处理结果后将其传递给 CH，最终发送到请求车辆。

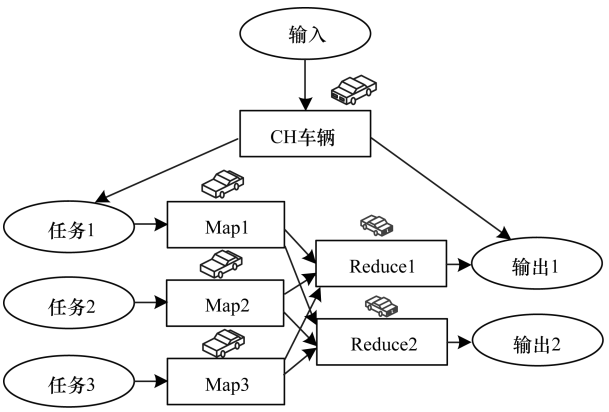


图 1　基于 MapReduce 模型的车载云计算框架

在 CH 中，根据每个成员节点的能力进行任务调度分配，该容量由其包含的插槽数量所决定，插槽是车辆上可用物理资源的简单抽象。每个车辆的 Map 和 Reduce 插槽数量是固定的，插槽/核心比是由系统设计人员指定的。每个工作节点定期监测可用插槽，并通过以下二元组信息形式通知 CH：

$$<ID,loc(x,y),speed,acceleration,freeslot> \tag{3}$$

本文所用到的符号和参数声明如下：$\mathcal{J}$ 表示调度请求；$\mathcal{M}$ 表示集群中所有成员节点集合；S_m 表示节点可用的插槽集，$m\in\mathcal{M}$；$\mathcal{B}$ 表示用于无线信道的比特率（单位为 bit/s）；D_{φ}^{jms} 表示在节点 $m\in\mathcal{M}$ 插槽 $s\in S_m$ 位置进行 Map 任务 $j\in J$ 的传输延迟；$D_{\mathrm{P_1}}^{jms}$ 表示节点 $m\in\mathcal{M}$ 插槽 $s\in S_m$ 位置进行 Map 任务 $j\in\mathcal{J}$ 的处理延迟；D_{φ}^{jms} 表示在节点 $m\in\mathcal{M}$ 插槽 $s\in S_m$ 位置进行 Reduce 任务 $j\in\mathcal{J}$ 的传输延迟；$D_{\mathrm{P_2}}^{jms}$ 表示在节点 $m\in\mathcal{M}$ 插槽 $s\in S_m$ 位置进行 Reduce 任务 $j\in\mathcal{J}$ 的处理延迟；D_{R}^{jms} 表示在节点 $m\in\mathcal{M}$ 插槽 $s\in S_m$ 位置进行 Reduce 任务 $j\in\mathcal{J}$ 的结果收集延迟；v_m 表示节点 $m\in\mathcal{M}$ 的指令执行速率；δ_m 表示节点 $m\in\mathcal{M}$ 与 CH 节点的接触率；t_m^j 表示节点 $m\in\mathcal{M}$ 上一个插槽执行任务 $j\in\mathcal{J}$ 的延迟；σ_m 表示节点 $m\in\mathcal{M}$ 与 CH 节点间的接触时间；$p_m(t)$ 表示节点 $m\in\mathcal{M}$ 与 CH 节

点间的接触概率；p_m^j 表示节点 $m \in \mathcal{M}$ 上任务 $j \in \mathcal{J}$ 的执行成功概率；α 为响应时间与成功概率的相对优先级因子。

2 任务调度优化

2.1 成功概率计算

节点服务器在 $\mathcal{T}$ 之前具有更高的完成任务的概率，本文利用节点车辆同 CH 节点的接触率来计算概率值，其中节点 $m \in \mathcal{M}$ 的接触率为：

$$\delta_m = 1/E[\sigma_m] \tag{4}$$

节点 m 和 CH 之间的接触率 δ_m 满足泊松分布，节点的成功率的期望最大化可通过最大化每个单独任务的成功概率实现。节点 $m \in \mathcal{M}$ 执行任务 x_j 所需时间为：

$$t_m^j = f(x_j)/v_m \tag{5}$$

其中，$f(\ )$ 函数是 $f'(\ )$ 或者 $f''(\ )$。

节点 $m \in \mathcal{M}$ 在时间间隔 0 ~ t 的任意值 t 在区间 $0 \leqslant t \leqslant \mathcal{T} - t_m^j$ 第一次遇见 CH 的连接概率 $p_m(t)$ 为：

$$p_m(t) = \int_0^{T-t-t_m^j} \delta_m \cdot \mathrm{e}^{-\delta_m y} \mathrm{d}y = 1 - \mathrm{e}^{-\delta_m(T-t-t_m^j)} \tag{6}$$

节点 $m \in \mathcal{M}$ 上的任务 $j \in \mathcal{J}$ 的计算成功概率 p_m^j 是连接概率 $p_m(t)$ 在区间 0 ~ t_m^j 上的累积分布函数：

$$\begin{aligned} P_m^j &= \zeta_m \times \int_0^{T-t_m^j} p_m(t) \cdot \delta_m \cdot \mathrm{e}^{-\delta_m t} \mathrm{d}t \\ &= \zeta_m \times \{1 - (\delta_m(T - t_m^j) + 1)\mathrm{e}^{-\delta_m(T-t_m^j)}\} \end{aligned} \tag{7}$$

其中，二进制变量 ζ_m 表示节点 $m \in \mathcal{M}$ 上的插槽的可用性，如果节点 $m \in \mathcal{M}$ 上至少有一个可用插槽，则 $\zeta_m = 1$，否则 $\zeta_m = 0$。

因此，作业 $\mathcal{J}$ 中的所有任务的综合成功概率为：

$$S(\mathcal{J}) = \prod_{\forall j \in \mathcal{J}} p_m^j \mid p_m^j > 0 \tag{8}$$

2.2 响应时间计算

执行作业的响应时间由 Map 任务传播时间、Map 任务执行时间、Map 任务发送至 Reduce 任务时间、Reduce 任务执行时间和结果汇总时间组成。令二进制变量 A_{jms} 在节点 $m \in \mathcal{M}$ 插槽 $s \in S_m$ 位置执行任务 $j \in \mathcal{J}$ 时，取值为 1，否则取值为 0。CH 节点将作业 $\mathcal{J}$ 分解成 j' 个子任务，在 Map 插槽中执行，并将 Map 结果传递至 $(j - j')$ 个 Reduce 插槽中执行。函数 $f'(\ )$ 和 $f''(\ )$ 分别提供在 Map 和 Reduce 上进行任务操作的指令。因此，Map 和 Reduce 任务集分别为：

$$\mathcal{J} = \{x_1, x_2, \cdots, x_j \mid (\forall j > j', \exists j'' \leqslant j') x_j\} = f'(x_j'') \tag{9}$$

选取适当的插槽数量 j，然后对 Map 和 Reduce 任务进行调度。令 Y 为 Map 和 Reduce 的输出元素集为：

$$Y = \{y_1, y_2, \cdots, y_j \mid (\forall j \in J) y_j = f'(x_j \mid j \leqslant j') \vee f''(x_j \mid j > j')\} \tag{10}$$

根据不同的应用类型，$f'(x_j)$ 和 $f''(x_j)$ 的执行时间存在差异。每个节点将其包含邻域节点 $m \in \mathcal{M}$ 的连接列表发送至 CH 节点，据此 CH 节点可构建二维连接矩阵 $\boldsymbol{C}$，其元素 C_{mn} 取值如下：如果 $m = n$，则 $C_{mn} = 0$；如果 m 与 n 为邻域节点，则 $C_{mn} = 1$；如果 m 与 n 通过 CH 节点连接，则 $C_{mn} = 2$。考虑到 $\mathcal{B}$ 为无线信道比特率（单位为 bit/s），节点 $m \in \mathcal{M}$ 接收 Map 任务的延迟为：

$$D_{\varphi}^{jms} = \max_{\forall j \in \mathcal{J} \mid j \leqslant j'} \left\{ A_{jms} \cdot \frac{x_j}{\mathcal{B}} \right\} \tag{11}$$

不同的车载单元可能有不同的计算能力，因此，不同节点任务 x_j 的执行延迟不同，则延迟 $D_{\mathrm{P}_1}^{jms}$ 为：

$$D_{\mathrm{P}_1}^{jms} = \max_{\forall j \in \mathcal{J} \mid j \leqslant j'} \left\{ A_{jms} \cdot \frac{f'(x_j)}{v_m} \right\} \tag{12}$$

任何节点 $m \in \mathcal{M}$ 的 Map 任务结束后，节点 $m \in \mathcal{M}$ 的 $|\mathcal{J}| - j'$ 的 Reduce 任务联合传输延迟为：

$$D_{\varphi}^{jms} = \max_{\forall j \in \mathcal{J} \mid j \leqslant j'} \left\{ A_{jms} \cdot (|\mathcal{J}| - j') \cdot C_{mn} \cdot \frac{y_j}{\mathcal{B}} \right\} \tag{13}$$

因此，在节点 $m \in \mathcal{M}$ 插槽 $s \in S_m$ 位置进行 Reduce 任务 $j \in \mathcal{J}$ 的处理延迟 $D_{\mathrm{P}_2}^{jms}$ 为：

$$D_{\mathrm{P}_2}^{jms} = \max_{\forall j \in \mathcal{J} \mid j \leqslant j'} \left\{ A_{jms} \cdot \frac{f''(x_j)}{v_m} \right\} \tag{14}$$

延迟 D_{R}^{jms} 计算公式为：

$$D_{\mathrm{R}}^{jms} = \max_{\forall j \in \mathcal{J} \mid j \leqslant j'} \left\{ A_{jms} \cdot \frac{y_j}{\mathcal{B}} \right\} \tag{15}$$

因此，作业调度的总体响应时间为：

$$D_T^{\mathcal{J}} = D_{\varphi}^{jms} + D_{\mathrm{P}_1}^{jms} + D_{\mathrm{P}_2}^{jms} + D_{\mathrm{R}}^{jms} \tag{16}$$

相反地，如果作业的任务是在本地计算，则所需的时间为：

$$T_{\mathrm{o}} = \frac{\sum_{\forall j \in \mathcal{J} \mid j \leqslant j'} f'(x_j) + \sum_{\forall j \in \mathcal{J} \mid j \leqslant j'} f''(x_j)}{v_{\mathrm{o}} \times S_{\mathrm{o}}} \tag{17}$$

其中，v_{o} 是工作请求的单插槽指令执行率，S_{o} 是可用的插槽数量。

2.3 问题优化模型

上述资源调度优化问题可表示为混合整数线性规划的优化问题，目标是最大化成功概率 $S(\mathcal{J})$ 并最小化总服务时间 $D_{\tau}^{\mathcal{J}}$，则可得目标函数为：

$$\max Z = \alpha S(\mathcal{J}) - (1 - \alpha) \frac{D_{\tau}^{\mathcal{J}}}{T_{\mathrm{o}}} \tag{18}$$

参数 α 给出了作业完成时间与成功概率之间的相对优先权。一些应用程序可能是延迟容忍的，但

需要更高的成功处理概率,而其他的实时应用程序可能会强调工作完成时间,而不是成功概率,则根据响应时间 $\mathcal{I}_s$ 和成功概率 $\mathcal{I}_r$,可定义参数 α 的计算公式为:

$$\alpha = \frac{\mathcal{I}_s}{\mathcal{I}_r + \mathcal{I}_s} \tag{19}$$

1)插槽可用约束,所有节点的可用插槽总数应高于或等于任务总数;否则,作业 $\mathcal{J}$ 的执行是不可行的:

$$|\mathcal{J}| \leqslant \sum_{\forall m \in \mathcal{M}} |S_m| \tag{20}$$

2)作业完整性约束,作业 $\mathcal{J}$ 的任务 x_j 必须分配给处理节点,没有任务可消失或保持未分配状态:

$$\sum_{\forall m \in \mathcal{M}} \sum_{\forall s \in S_m} \sum_{\forall j \in \mathcal{J}} A_{jms} = |\mathcal{J}| \tag{21}$$

3)分配约束条件如下:

$$A_{jms} \in \{0,1,\}, \forall j \in \mathcal{J}, \forall m \in \mathcal{M}, \forall s \in S_m \tag{22}$$

$$\sum_{\forall m \in \mathcal{M}} \sum_{\forall s \in S_m} A_{jms} = 1, \forall j \in \mathcal{J} \tag{23}$$

在一个任务中,在给定的时间可以分配给特定的插槽,其他的插槽可能处于空闲状态。

4)QoS 约束:

$$D_T^{\mathcal{J}} < T_o, D_T^{\mathcal{J}} \leqslant \mathcal{T} \tag{24}$$

5)能力约束,分配给处理节点的任务数不应超出其容量:

$$\sum_{\forall j \in \mathcal{J}} \sum_{\forall s \in S_m} A_{jms} \leqslant |S_m|, \forall m \in \mathcal{M} \tag{25}$$

3　车载云计算模型的启发式并行优化

3.1　算法步骤

对于给定的任务 $j \in J$,执行节点必须满足最小执行成功阈值 p^j,以及最小任务执行时间 Z,即必须满足如下不等式:

$$\forall j \in \mathcal{J} | p_m^j < p^j \{p_m^j \cdot \frac{1}{t_m^j}\} \leqslant p^j \{p^j \cdot \frac{1}{t_m^j}\} \leqslant \forall j \in \mathcal{J} | p_m^j \tag{26}$$

对于每个任务 $j \in \mathcal{J}$ 的最小执行成功阈值 p^j,通过所有节点的成功任务完成概率均值,并利用相对优先级因子缩放:

$$p^j = \begin{cases} a\,\overline{p^j} \times 2\alpha, \overline{p^j} \times 2\alpha \leqslant \max\limits_{\forall m \in \mathcal{M}} (p_m^j) \\ \max\limits_{\forall m \in \mathcal{M}} (p_m^j), \text{otherwise} \end{cases} \tag{27}$$

其中,$\bar{p}^j = \sum_{\forall m \in \mathcal{M}} p_m^j / |\mathcal{M}|$。在最坏情况下,如果所有的节点满足阈值,则 p^j 值上界由最大 p_m^j 值确定,其中 $\forall m \in \mathcal{M}$。

算法 1 给出了使用 MapReduce 的启发式任务调度算法的具体步骤。首先,对于每个任务 $j \in \mathcal{J}$,根据成功概率 p_m^j 选取执行节点集 M_p(第 9 行)。在此情形下,节点集 M_p 产生一个空集,即没有一个节点能够满足任务成功执行概率的最小阈值,然后插槽搜索停止。否则,在车辆节点候选插槽中对任务处理的延迟进行计算。同样地,计算候选插槽到其他选取插槽的数据传输延迟(第 17 行)。然后选取最小数据处理和传输时间的插槽,添加到所选插槽集中(第 20 行 ~ 第 22 行)。最后输出在节点 $m \in \mathcal{M}$ 插槽 $s \in S_m$ 位置执行任务 $j \in \mathcal{J}$ 情况的二进制变量标识 A_{jms}。

算法 1　启发式调度算法

输入　任务集 $\mathcal{J}$,车辆节点集 $\mathcal{M}$,节点 $m \in \mathcal{M}$ 上的插槽集 S_m,相应时间指标与成功概率指标间的优先级因子 α

输出　在节点 $m \in \mathcal{M}$ 插槽 $s \in S_m$ 位置执行任务 $j \in \mathcal{J}$ 情况的二进制变量标识 A_{jms}

1. $S_\Omega \leftarrow \Phi$;
2. $A_{jrms} = 0 | \forall j \in J, \forall m \in M, \forall s \in S_m$;
3. for $\forall j \in J$ do
4. $\alpha = I_s / (I_r + I_s)$;
5. for $\forall m \in M$ do
6. $P_m^j = \zeta_m \times \int_0^{T - t_m^j} p_m(t) \cdot \delta_m \cdot e^{-\delta_m t} dt = \zeta_m \times \{1 - (\delta_m (T - t_m^j) + 1) e^{-\delta_m (T - t_m^j)}\}$;
7. endfor
8. $p^j = \begin{cases} a\bar{p}^j \times 2\alpha, \text{ if } \bar{p}^j \times 2\alpha \leqslant \max\limits_{\forall m \in M} (p_m^j) \\ \max\limits_{\forall m \in M} (p_m^j), \text{ otherwise} \end{cases}$;
9. $M_p \leftarrow \{m_1, \cdots, m_k | \forall m \in M, p_m^j \geqslant p^j\}$;
10. if $M_p = \varnothing$ then
11. Exit;
12. endif
13. for $\forall s \in S_m | m \in M_p$ do
14. $t_m^j \leftarrow \frac{f(x_j)}{v_m}$;
15. $d_t^j \leftarrow 0$;
16. for $\forall \omega \in S_\Omega$ do
17. $d_t^j \leftarrow d_t^j + \frac{x_j}{B} \times C_{mn} | \omega \in S_n$;
18. endfor
19. endfor
20. $s_j \leftarrow \text{argmin}\{t_m^j + d_t^j\}$;
21. $S_m \leftarrow S_m \setminus s_j$;
22. $S_\Omega \leftarrow S_\Omega \cup s_j$;
23. $A_{jrms} = 1 | s = s_j \& s_j \in S_m$;
24. endfor

3.2　计算复杂度分析

对于算法 1 的计算复杂度,可通过逐步分析算法伪代码获得:第 3 行 ~ 第 24 行代码在一个总的循环中,共迭代执行 $|\mathcal{J}|$ 次;第 5 行 ~ 第 9 行具有相同的计算复杂度为 $O(|\mathcal{M}|)$;第 13 行 ~ 第 19 行最坏情况下执行 $|S_m|$ 次迭代;第 16 行 ~ 第 18 行执行

$|S_\Omega|$次迭代,其中 $S_\Omega \in S_m$;代码的其余部分有固定的单位时间复杂度。因此,车载云计算模型的启发式并行优化算法的计算复杂度构成如下:

$$O(|\mathcal{J}|\cdot|\mathcal{M}|+|\mathcal{J}|\cdot|S_m|^2)\approx O(|\mathcal{J}|\cdot|S_m|^2) \tag{28}$$

4 实验与结果分析

4.1 实验环境与评价指标

首先介绍实验环境设置,硬件设置:处理器为 i3-2440k 1.8 GHz,内存为 16 GB ddr2-800 GHz,系统为 Win7 旗舰版,仿真平台选取网络模拟器版本 3(NS-3)。利用每台计算机模拟一个车辆进行实验对比。考虑测试道路环境下,每公里道路区域内的车辆数量为20 ~50,行驶速度为 36 km/h ~90 km/h。假定每辆车的信号传输距离为 100 m,建立一个边长为 500 m 的城市道路环境,如图 2 所示,在方形区域的每边250 m处都有交叉路口。

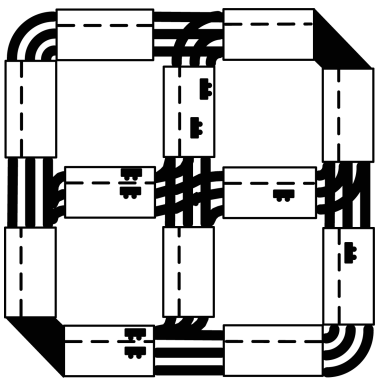


图 2 模拟场景

对于车辆之间的数据传输,采用车载环境的无线接入方式(IEEE 802.11p 标准),具有 7 个通道各有 6 Mb/s 的数据速率,每个数据包大小为 1 024 Byte,每个车辆具有 2 个 ~6 个 Map 插槽或者 Reduce 插槽,或者混合类型插槽。每个节点(计算机)都包含 1 GB RAM 以及 40 GB ~100 GB 的二级存储。在此仿真环境下,每分钟安排了 2 个 ~8 个应用,每组 10 个 ~20 个任务。根据不同的指令和输入,每个任务需不同执行周期。分别设定 $\mathcal{I}_s=6$ 和 $\mathcal{I}_r=4$,目的是在临界执行时间下,获得更高任务执行成功率。仿真运行 1 000 s,资源调度周期为 1 s,频率设定为 5.9 GHz,带宽为 50 MHz,发射功率为20 dBm。选取 20 个单独的模拟运行结果进行均值求取,以减少输入随机变化的影响。对比算法选取文献[14-16]算法。

根据以下指标对所研究系统的性能进行比较评估:作业平均执行时间,成功执行概率,系统吞吐量指标和任务的执行开销。

任务的执行开销定义如下:

$$O_{\text{overhead}}=\left(1+\frac{D_{P_1}^{jms}+D_{P_2}^{jms}}{D_T^{\mathcal{J}}}\right)\times 100 \tag{29}$$

4.2 车辆密度影响

一般情况下,增加处理节点数量,能否增加可用的数据处理资源,并降低作业执行时间。但同时,它还会增加节点之间数据传输时的干扰,进而增加数据传输时间。图 3 显示了 20 辆/km ~50 辆/km 不同车辆密度对道路系统性能影响。作业到达率固定在 6 个/min,用于测量系统的效率。

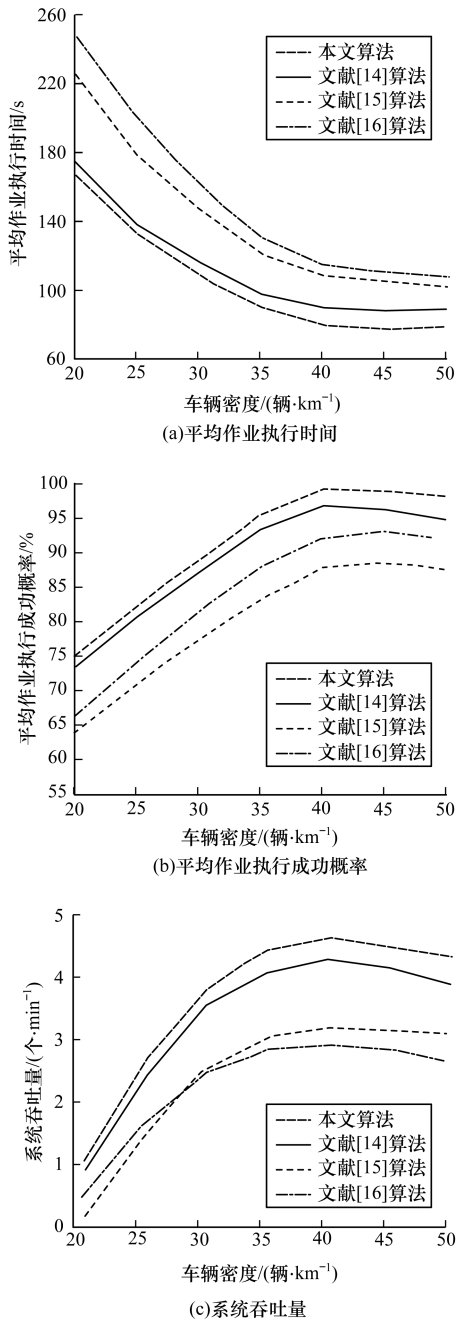


图 3 车辆密度对任务调度模型性能的影响

根据图 3(a)可知，随着车辆密度的增加，作业的执行时间均值均有所下降，这是由于随着车辆数量的增加，可用计算资源增多，因此作业执行时间降低，当车辆密度增加到一定程度后，计算时间达到饱和。但总体上本文算法计算时间要优于选取对比算法，而文献[14]要优于文献[15]和文献[16]算法，文献[15]优于文献[16]算法。

根据图 3(b)可知，随着车辆密度的增加，平均作业执行成功概率指标先增加后降低，这是由于车辆增加会降低车辆之间的信息传输距离，以提高信息传输的可靠性，因此，在到达一定车辆密度前，平均作业成功概率指标随车辆密度增加而增加。但因为车辆增加会增加车辆间信息传递干扰，所以当车辆密度到达一定程度后，继续增加车辆密度反而会降低作业执行成功概率。总体上本文算法平均作业成功概率要优于选取的对比算法，而文献[14]要优于文献[15]和文献[16]算法，文献[16]优于文献[15]算法。

根据图 3(c)可知，随着车辆密度的增加，吞吐量指标同样先增加后有所降低，这是由于车辆增加了处理资源，但是同样因为车辆的增加会增加车辆间信息传递的频次，因此当车辆密度到达一定程度后，继续增加车辆密度反而会降低系统吞吐量。

总体上本文算法计算时间要优于选取的对比算法，而文献[14]要优于文献[15]和文献[16]算法，文献[16]和文献[15]算法相差不大。

4.3　作业到达率影响

在 VCC 工作环境下增加到达率，系统性能会得到逐步提升，并逐步达到饱和。图 4 显示了在 VCC 节点不同的工作到达率的系统性能表现。在此情形下，考虑道路车辆密度为 30 辆/km。根据图 4(a)可知，随作业到达率增加，作业执行时间均值呈现先缓慢增加后快速升高的特点，主要原因在于作业到达率小于一定数值时，车载计算资源未达到满负荷运转，而当大于该数值时，因为车载计算资源满负荷运行导致计算时间增加。总体上本文算法计算时间要优于对比算法。根据图 4(b)可知，随作业到达率增加，作业执行成功概率呈现先缓慢降低后快速降低的特点，主要原因与图 4(a)作业的执行时间变化原因相同。总体上本文算法平均作业成功概率要优于对比算法。根据图 4(c)可知，随作业到达率增加，作业执行成功概率呈现先快速增加后逐渐降低的特点，原因在于车载计算资源满负荷运行后，继续增加作业到达率，会增加计算机的运行负荷，反而降低了系统的吞吐量指标。因此，本文算法系统吞吐量要优于选取的对比算法。

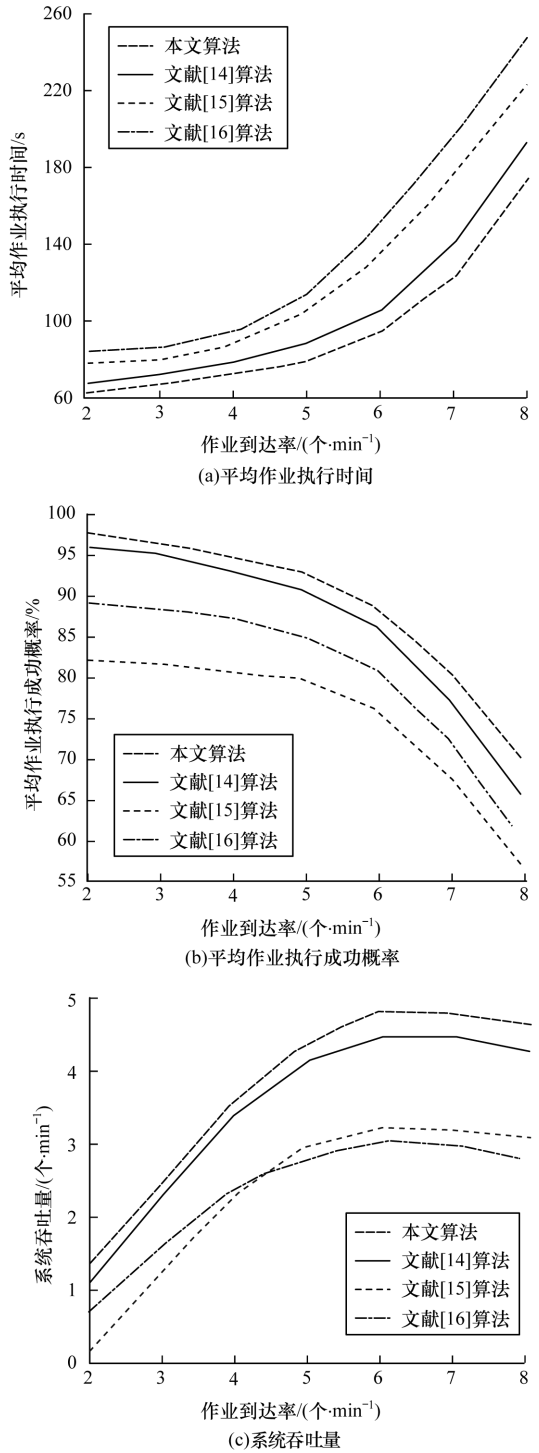
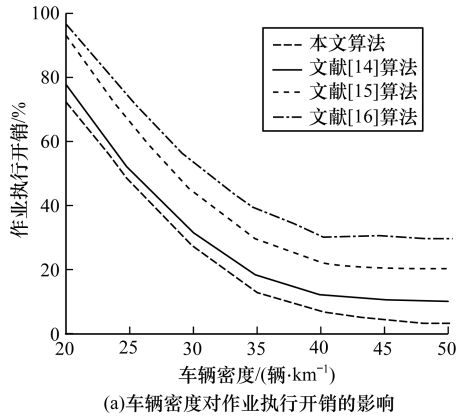


图 4　作业到达率对任务调度模型性能的影响

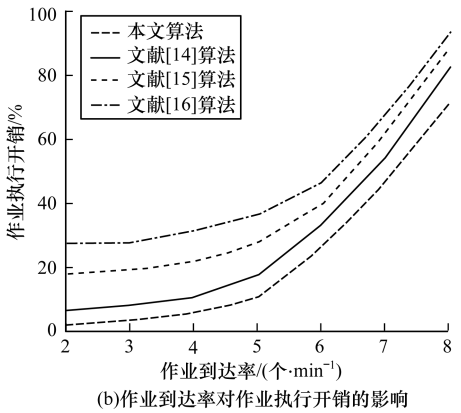
4.4　任务执行开销

图 5 给出了本文研究的任务调度系统在不同车

辆密度和作业到达率下的任务执行开销。对于不同车辆密度的任务执行开销,工作到达率固定为6 个/min,对于不同的工作到达率任务执行开销,车辆密度固定在 30 辆/km。



(a)车辆密度对作业执行开销的影响



(b)作业到达率对作业执行开销的影响

图 5 系统任务执行开销

根据图 5(a)可知,随着车辆密度增加,作业执行开销先增加后逐渐达到饱和,其中本文算法所需的作业执行开销始终小于选取的 3 种对比算法。

根据图 5(b)可知,随着作业到达率的增加,作业执行开销先缓慢增加,在达到计算饱和后作业开销迅速增加,其中本文算法所需的作业执行开销始终小于选取的 3 种对比算法。

上述 2 组实验结果表明,本文算法具有较高的资源调度利用率。

4.5 并行效率

为验证本文算法的并行执行效率优势,选取每个车辆具有 2 个 ~6 个不同数量 Map 插槽或者 Reduce 插槽,对其进行实验分析。对比算法仍然选取文献[14-16]算法,实验参数设置同上。对比指标选取平均作业执行时间、平均作业成功概率和系统吞吐量。作业到达率参数设置为 5 个/min。对比实验结果如表 1 所示。可以看出,随着 Map 插槽或者 Reduce 插槽数量的增加,本文算法的平均作业执行时间降低,成功概率和系统吞吐量增加,算法的整体性能得到提升,这体现了所提 Mapreduce 算法的性能优势。同时,可以看出随着插槽数量的增加算法性能的提升幅度逐渐降低,这表明插槽数量的增加不能无限制地提高算法性能,由于受到插槽之间数据交换的影响,会对算法执行效率和精度产生影响。在本文实验中,当选取插槽数量为 5 时,算法的性能较好,而文献[14-16]算法因为未采用并行 MapReduce 算法,所以计算效率较低,且成功概率和系统吞吐量也受到一定影响。

表 1 并行效率实验对比数据

算法	插槽数	平均作业执行时间/s	平均作业成功概率/%	系统吞吐量/(个·min⁻¹)
本文算法	2	98.4	93.2	3.9
	3	91.6	95.1	4.1
	4	85.2	96.7	4.3
	5	83.7	97.1	4.5
	6	83.2	97.3	4.6
文献[14]算法	–	99.3	93.8	3.9
文献[15]算法	–	110.6	83.7	3.5
文献[16]算法	–	123.5	87.1	3.8

5 结束语

本文提出一种考虑服务质量的并行 MapReduce 启发式车载云资源调度算法。建立基于 MapReduce 并行计算模型的车载云计算车辆并行检测服务框架,利用启发式并行优化算法对车载云资源调度模型进行优化,并通过仿真对比结果验证所提调度算法的有效性。由于受到实验条件限制,对本文算法的验证只在实验室环境下进行,而其对于真实道路环境的适应能力,有待进一步的实验验证。

参考文献

[1] 唐 伦,王晨梦,陈前斌. 车载自组织网络中基于时分复用的异步多信道 MAC 协议[J]. 计算机学报,2015,38(3):673-684.

[2] GUERRERO-IBANEZ J A, CONTRERAS-CASTILLO J, ZEADALLY S. Integration Challenges of Intelligent Transportation Systems with Connected Vehicle, Cloud Computing, and Internet of Things Technologies[J]. IEEE Wireless Communications, 2015, 22(6): 122-128.

[3] 李元振,廖建新,李彤红,等. 城市场景车载 Ad Hoc 网络竞争转发关键参数分析[J]. 电子学报,2011,39(5):1154-1158.
[4] ZHANG Hongli,ZHANG Qiang,DU Xiaojiang. Toward Vehicle-assisted Cloud Computing for Smartphones[J]. IEEE Transactions on Vehicular Technology,2015,64(12):5610-5618.
[5] HE Wu,YAN Gongjun,XU Lida. Developing Vehicular Data Cloud Services in the IoT Environment[J]. IEEE Transactions on Industrial Informatics,2014,10(2):1587-1595.
[6] ZHAO Jiaqi,TAO Jie,STREIT A. Enabling Collaborative MapReduce on the Cloud with a Single-sign-on Mechanism[J]. Computing,2016,98(1):55-72.
[7] 贾建斌,陈颖文,徐　明. 基于预测的机会车载网络中继选择策略研究[J]. 软件学报,2015,26(7):1730-1741.
[8] JIAU M,HUANG S,HWANG J,et al. Multimedia Services in Cloud-based Vehicular Networks[J]. IEEE Intelligent Transportation Systems Magazine,2015,7(3):62-79.
[9] YU Rong,ZHANG Yan,GJESSING S,et al. Toward Cloud-based Vehicular Networks with Efficient Resource Management[J]. IEEE Network,2013,27(5):48-55.
[10] HAN Liangxiu,ONG H Y. Parallel Data Intensive Applications Using MapReduce:A Data Mining Case Study in Biomedical Sciences[J]. Cluster Computing,2015,18(1):403-418.
[11] 宋彩霞,谭国真,丁　男,等. 面向应用的车载自组织网络跨层多信道 MAC 协议[J]. 通信学报,2016,37(5):95-105.
[12] LI Ren,HU Haibo,LI Heng,et al. MapReduce Parallel Programming Model:A State-of-the-Art Survey[J]. International Journal of Parallel Programming,2016,44(4):832-866.
[13] TRAN H M,HA S V U,HUYNH T K,et al. A Feasible MapReduce Peer-to-Peer Framework for Distributed Computing Applications[J]. Vietnam Journal of Computer Science,2015,2(1):57-66.
[14] 肖锐锋,傅　明,唐雨龙. 车载自组织网络中基于链路可靠性的路由优化方案[J]. 计算机工程,2016,42(3):13-17.
[15] AMINIZADEH L,YOUSEFI S. Cost Minimization Scheduling for Deadline Constrained Applications on Vehicular Cloud Infrastructure[C]//Proceedings of the 4th International Conference on Computer and Knowledge Engineering. Washington D. C.,USA:IEEE Press,2014:358-363.
[16] BITAM S,MELLOUK A. ITS-cloud:Cloud Computing for Intelligent Transportation System[C]//Proceedings of 2012 IEEE Global Communications Conference. Washington D. C.,USA:IEEE Press,2012:2054-2059.

编辑　金胡考

(上接第 29 页)

[8] RAHIMI M R,VENKATASUBRAMANIAN N,MEHROTRA S,et al. MAPCloud:Mobile Applications on an Elastic and Scalable 2-tier Cloud Architecture[C]//Proceedings of the 5th International Conference on Utility and Cloud Computing. Washington D. C.,USA:IEEE Computer Society,2012:83-90.
[9] EL-DERINI M N,ALY H H,EL-BARBARY E H G,et al. Droid Cloudlet:Towards Cloudlet-based Computing Using Mobile Devices[C]//Proceedings of the 5th International Conference on Information and Com-munication Systems. Washington D. C.,USA:IEEE Press,2014:1-6.
[10] VERBELEN T,SIMOENS P,de TURCK F,et al. Adaptive Deployment and Configuration for Mobile Augmented Reality in the Cloudlet[J]. Journal of Network and Computer Applications,2014,41(1):206-216.
[11] MURALEEDHARAN R. Cloud-Vision:Real-time Face Reco-gnition Using a Mobile-Cloudlet-Cloud Acceleration Architecture[C]//Proceedings of International Symposium on Computers and Communications. Washington D. C.,USA:IEEE Press,2012:59-66.
[12] RAVI A,PEDDOJU S K. Mobility Managed Energy Efficient Android Mobile Devices Using Cloudlet[C]//Proceedings of IEEE Students' Technology Symposium. Washington D. C.,USA:IEEE Press,2014:402-407.
[13] 许　丞,刘　洪,谭　良. Hadoop 云平台的一种新的任务调度和监控机制[J]. 计算机科学,2013,40(1):112-117.
[14] KHAZAEI H,MISIC J V B. Performance Analysis of Cloud Computing Centers Using M/G/m/m + r Queuing Systems[J]. IEEE Transactions on Parallel and Distributed Systems,2012,23(5):936-943.
[15] 徐　昕. 基于博弈论的云计算资源调度方法研究[D]. 上海:华东理工大学,2015.
[16] 陈丽萍,郭力争,赵曙光. 云计算环境下基于排队论的数字图书馆性能优化[J]. 东华大学学报(自然科学版),2014,40(3):318-321.
[17] 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.
[18] 刘　赛,李绪蓉,万麟瑞. 基于排队论的云计算资源池模型研究[J]. 计算机技术与发展,2012,22(12):87-89.
[19] 韩玉群,梁希泉. 一类服务率可变的 M/M/s/K 排队模型研究[J]. 青岛科技大学学报(自然科学版),2014,35(1):95-99.

编辑　陆燕菲