AUV 协同设计平台中多任务流调度算法研究*

许真珍[1]，赵小薇[1]，徐秀娟[1]，胡志强[2]，陈 鑫[1†]
(1. 大连理工大学 软件学院，辽宁 大连 116620；2. 中国科学院沈阳自动化研究所 机器人学国家重点实验室，沈阳 110016)

摘 要：对 AUV 协同设计平台中多个任务流的调度问题进行建模，将其转换为分布式计算环境下的独立任务在线调度问题。针对系统异构和任务流具有优先级属性的特殊性，提出了一种基于预测的多任务流调度算法，采用统计和预测的方法评估各工作站执行任务的效用，并设计优先级策略和暂停调度策略，保证具有较高优先级的任务流较早分配和执行。实验结果表明，该算法在参数选取适当的情况下，性能优于传统的 MCT 和 MET 任务调度算法。

关键词：分布式计算；任务调度；异构系统；优先级；预测
中图分类号：TP311 **文献标志码**：A **文章编号**：1001-3695(2014)05-1345-04
doi:10.3969/j.issn.1001-3695.2014.05.014

Research on multiple task flows scheduling algorithm in collaborative design platform for AUV

XU Zhen-zhen[1], ZHAO Xiao-wei[1], XU Xiu-juan[1], HU Zhi-qiang[2], CHEN Xin[1†]
(1. *School of Software, Dalian University of Technology, Dalian Liaoning* 116620, *China*; 2. *State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang* 110016, *China*)

Abstract: This paper modeled the multiple task flows scheduling problem in the collaborative design platform for AUV and transformed it into the independent tasks online scheduling problem in the distributed computing environment. Aiming at the particularity including the system was heterogeneous and the task flow had priority attribute, and proposed a prediction-based multiple task flows scheduling algorithm. Evaluated the utility of executing the task on each workstation using statistics and prediction method to improve the performance of task scheduling and designed the priority strategy to ensure the task flow with a higher priority could be assigned and executed earlier. The experimental results show that this algorithm performs better than some classical algorithms such as MCT and MET task scheduling algorithms when the parameters are well selected.

Key words: distributed computing; task scheduling; heterogeneous system; priority; prediction

0 引言

自治水下机器人(autonomous underwater vehicle, AUV)协同设计平台是一个由一台服务器和多台工作站组成的异构分布式计算系统，每台工作站的软硬件结构均可不同。用户通过客户端浏览器访问设计平台，并定义和提交由多个计算任务组成的任务流来完成 AUV 的相关设计，如 AUV 水动力计算需要由 Gridgen、CFX-Pre、CFX-Solver 和 CFX-Post 软件分别完成网格划分、数据预处理、流体动力学计算和数据后处理四种计算任务。设计平台负责调度不同用户提交的多个任务流，将任务流中的计算任务分配到合适的工作站上执行，从而达到分布式并行计算的目的。

该问题属于分布式计算环境下的独立任务在线调度问题，现有的在线调度算法通常是基于最快执行时间或最快完成时间等指标将任务分配到各处理机上。MET 算法[1]把任务分配给具有最少执行时间的处理机，此算法会把大量的任务集中分配到某些性能较高的处理机上，导致其他处理机空闲，造成系统的负载不均衡。MCT 算法[2]按任务最早完成时间为指标进行分配，可以保证多处理机间的负载均衡，但任务的实际执行时间可能并非最小。SA 算法[3]结合了 MCT 和 MET 的分配思想，当系统处于负载均衡状态时，使用 MET 算法；当负载不均衡时，使用 MCT 算法进行分配，整个系统在负载均衡与不均衡间波动。KPB 算法[3]按 $k\%$ 的比例选取任务执行时间较小的若干处理机，再选取最早完成时间最小的处理器进行调度。当 $k=100/m$(m 为处理机数)时，KPB 算法退化为 MET 算法；当 $k=100$ 时，KPB 算法退化为 MCT 算法。

上述经典算法均假设处理机为平行机的同构环境，即每个任务可以在任一台处理机上执行，并未考虑每个处理机可以执行的任务种类不同的情况。文献[4]考虑了异构系统的任务分配，但研究背景是多核处理器，并不适用于 AUV 协同设计平台的分布式计算环境。现有研究异构系统任务分配的文献往

收稿日期：2013-08-08；**修回日期**：2013-09-28 **基金项目**：国家自然科学基金资助项目(51209036,61300016)

作者简介：许真珍(1980-)，女，江苏淮安人，讲师，博士，主要研究方向为分布式智能系统、任务调度算法；赵小薇(1978-)，女，辽宁大连人，讲师，博士研究生，主要研究方向为复杂系统；徐秀娟(1978-)，女，吉林舒兰人，讲师，博士，主要研究方向为智能算法；胡志强(1980-)，男，湖北咸宁人，副研究员，博士，主要研究方向为水下机器人总体设计；陈鑫(1983-)，男(通信作者)，辽宁锦州人，讲师，博士研究生，主要研究方向为调度算法(chenx_dlut@163.com).

往只考虑多个任务的分配问题,没有考虑到多个带优先级属性的任务流分配问题[5~7]。因此,本文提出一种基于预测的任务调度算法,针对异构系统和多任务流任务分配的特殊性,从统计和预测的角度出发,结合动态变化的分布式计算系统信息来评估任务调度的效用,从而提高任务分配的性能,并保证具有较高优先级的任务流较早完成。

1 问题描述

AUV 协同设计平台的总体结构如图 1 所示。负责执行计算任务的多台工作站具有不同的硬件配置(如 CPU、内存等)和软件环境(如操作系统、设计软件等),可以执行不同种类的计算任务。客户端将创建好的任务流提交给服务器,服务器根据工作站的配置和运行情况,对多个任务流中的计算任务进行在线调度。

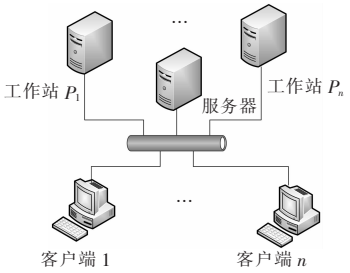

图 1 AUV 协同设计平台总体结构

考虑到 AUV 设计中任务的特殊性,AUV 协同设计平台中待分配的任务流具有如下特性:a)任务流中各计算任务是相互独立的;b)一台工作站在同一时间只能处理一个任务,不能并行处理多个任务;c)任务调度是非抢占式的,即如果一个任务已经开始执行,在它执行完之前,其他任务不能抢占其工作站上的资源;d)每个任务流中的任务必须按脚本定义的逻辑顺序逐个执行;e)单位时间(1 min)内任务到达概率较低($\leqslant 0.1$);f)任务的运算时间长,从几十分钟至几十小时不等。

设系统中工作站的集合为 $P=\{p_1,p_2,\cdots,p_m\}$,$m\in N$,一台工作站相当于一台处理机,所有任务种类的集合为 $R=\{r_1,r_2,\cdots,r_n\}$,$n\in N$。矩阵 $\boldsymbol{PR}=[pr_{ij}]_{m\times n}$ 表示处理机与它所能执行任务种类的映射关系,其中:

$$pr_{ij}=\begin{cases}1 & \text{如果处理机 } p_i \text{ 可以执行种类为 } r_j \text{ 的任务}\\ 0 & \text{否则}\end{cases}$$

假设用户陆续提交的多个任务流集合为 $F=\{\text{flow}_1,\text{flow}_2,\cdots,\text{flow}_w\}$,$w\in N$,其中 flow_k($1<k<w$)表示一个计算任务序列 $\text{flow}_k=\{n_1,n_2,\cdots,n_q\}$,$q\in N$。$\text{flow}_k$ 中全部任务执行完毕花费的时间记为 $CT(f_k)$,所有任务流全部执行完毕所耗费的时间记为 Makespan。假设任务 n_i 的种类为 r_j,则集合 $P(n_i)=\{p_k\mid pr_{kj}=1,k=1,2,\cdots,m\}$ 表示可以执行任务 n_i 的处理机集合。

由于 AUV 协同设计平台中只有一台服务器负责任务调度,由特性 a)和 d)可以看出,该问题可以抽象为一类独立任务调度问题。在处理机异构环境下,如果任务分配不当,将严重影响任务执行性能指标。如果在任务分配时能够预测到即将到达的任务类型,将有利于提高任务执行效率和保证各处理器的负载均衡。因此,本文提出一个基于预测的多任务流调度算法(prediction-based multiple task flows scheduling algorithm, PMTF)来解决异构处理机和多任务流条件下的任务分配问题。

2 算法描述

PMTF 算法的主要思想是设置任务流优先级队列,结合任务流的优先级属性和提交时间对任务流进行排序,决定当前服务器要分配的计算任务,并把按单一指标进行的任务调度转换成按工作站执行任务的效用(utility)进行分配。效用的计算将包括三个方面:a)处理机执行任务的直接回报(direct reward);b)处理机因为执行该任务而不能执行其他可能到来的任务导致的预计损失(predicted loss);c)处理机上因存在排队等待的任务,考虑到负载均衡而引入的延迟惩罚(delay penalty)。因此处理机 p_i 执行任务 n_j 的效用可以定义为

$$\text{Utility}(p_i,n_j)=DR(p_i,n_j)-PL(p_i,n_j)-DP(p_i,n_j) \tag{1}$$

2.1 直接回报

处理机 p_i 执行任务 n_j 所能获得的直接回报 $DR(p_i,n_j)$ 为

$$DR(p_i,n_j)=rwd_{ik}\times \text{PET}(p_i,n_j) \tag{2}$$

其中:$\text{PET}(p_i,n_j)$ 表示处理机 p_i 执行任务 n_j 的预计执行时间(predicted execution time)。若 n_j 的任务类型为 r_k,设直接回报矩阵 $\boldsymbol{RWD}$ 为一个 $|P|\times|R|$ 的矩阵,其中的元素 rwd_{ik} 代表处理机 p_i 运行种类为 r_k 的任务,单位时间所能得到的直接回报。

2.2 预计损失

如果处理机执行 n_j,那么在任务执行期间无法执行其他任务的预计损失 $PL(p_i,n_j)$ 需要通过预测来实现。预测是指系统根据以往各类型任务的到达频率、执行时间等信息进行统计,并对未来任务的出现作出合理的估计。由假设条件 e)和 f)以及概率论的知识可知,在时间段 T 内,类型为 r_k 的任务到达次数是一个随机变量(记做 X),X 近似服从参数 $\lambda=T\times Prb(r_k)$ 的泊松分布,其中 $Prb(r_k)$ 表示 r_k 类型任务在单位时间内出现的概率。任务平均执行时间(average execution time)$\text{AET}(r_k)$ 表示通过统计所得到的 r_k 类型任务的平均执行时间。设 n_j 的任务类型记为 $r(n_j)$,则 $PL(p_i,n_j)$ 是指在时间段 T 内处理机 p_i 遇到 n_j 之外其他类型任务所获得回报的期望,表示为

$$PL(p_i,n_j)=\sum_{k:pr_{ik}=1\wedge k\neq r(n_j)}\ \sum_{m=1}^{\left\lfloor\frac{\text{PET}(n_j)}{\text{AET}(r_k)}+1\right\rfloor} m\cdot q\cdot P(X=m)\cdot rwd_{ik}\cdot \text{AET}(r_k) \tag{3}$$

其中:$P(X=m)=\mathrm{e}^{-\lambda}\dfrac{\lambda^m}{m!}$,但此概率计算未考虑到条件假设 b)和 c),即处理机在同一时间只能执行一个任务,并且任务调度是非抢占的。因此需要引入一个参数 $q(0\leqslant q\leqslant 1)$,其值可由引理 1 获得。

引理 1 在时间段 T 内,类型为 r_k、执行时间为 t 的任务出现的次数 X 服从参数为 λ 的泊松分布,那么在任务执行期间此类任务没有重复出现的情况下,任务共出现 m 次的概率为 $q\cdot\mathrm{e}^{-\lambda}\dfrac{\lambda^m}{m!}$,其中 $q=\begin{pmatrix}T+t-m\cdot t+m\\ m\end{pmatrix}\Big/\begin{pmatrix}T\\ m\end{pmatrix}$。

证明 若时间段 T 内,任务出现了 m 次,则最后一个出现的任务执行完成时间可能会超出 T,并介于 T 与 $T+t$ 之间。为了方便讨论,延长 T 至 $T+t$,图 2 给出了 $m=3$ 时的任务执行情况示例。

时间段 $T+t$ 由 m 个 t 和 $m+1$ 个空闲时间片组成,各空闲

时间片长度分别记做 $x_1, x_2, \cdots, x_{m+1}$,可以得到如下方程:

$$x_1 + x_2 + \cdots + x_{m+1} = T + t - m \times t \tag{4}$$

其中:$x_k \geqslant 0, k = 1, 2, \cdots, m+1$。可以看到,每一个方程的解向量$(x_1, x_2, \cdots, x_{m+1})$都对应于 m 个任务在时间段 T 内的出现情况。因此,时间段 T 内出现 m 次任务的组合数就是方程所含有非负整数解向量的个数,于是可以得到此组合数为 $\binom{T+t-m\cdot t+m}{m}$。

因为 X 服从泊松分布,概率 $P\{X=m\}$ 包含了组合数为 $\binom{T}{m}$ 种任务出现情况概率的总和,所以可以得到 $q = \binom{T+t-m\cdot t+m}{m} \Big/ \binom{T}{m}$。证毕。

由引理 1 可以得到式(4)中参数 q 的计算方法。

$$q = \binom{T + \mathrm{AET}(r_k) - m \cdot \mathrm{AET}(r_k) + m}{m} \Big/ \binom{T}{m} \tag{5}$$

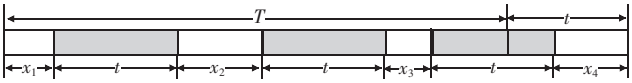

图 2　任务执行情况示意图

2.3　延迟惩罚

由于任务执行时间长、处理机数量有限等因素,系统还应该考虑到负载均衡的问题。处理机上已经分配的任务将导致待分配任务需要延迟执行。设延迟惩罚矩阵 PTY 为一个 $m \times n$ 的矩阵,其中元素 pty_{ij} 表示处理机 p_i 因已分配任务导致任务 n_j 等待执行单位时间的处罚量。设 n_r 表示处理机 p_i 正在执行的任务,函数 $ET(p_i, n_r)$ 表示处理机 p_i 上任务 n_r 已经执行的时间(elapsed time),已分配给 p_i 等待执行的任务数为 s,则有

$$DP(p_i, n_j) = [\mathrm{PET}(p_i, n_r) - ET(p_i, n_r) + \sum_{k=1}^{s} \mathrm{PET}(p_i, n_k)] \cdot pty_{ij} \tag{6}$$

2.4　优先级策略

每个任务流在用户提交时都设置了优先级属性,分为 1 ~ 5 共五个等级,默认为 1 级,最高为 5 级。优先级的设置受制于用户的权限,任务调度需要保证较高优先级任务流中的任务能够较早分配和执行。

假设 flow_k 中任务 n_i $(1 < i < q)$ 请求分配的时间记为 $AT(n_i)$,则对于 flow_k 中的任务 n_i 和 n_j $(i < j)$,有 $AT(n_i) < AT(n_j)$。任务流 flow_k $(k = 1, 2, \cdots, w)$ 的优先级记为 $\mathrm{Priority}(\mathrm{flow}_k)$。设 flow_k.next 表示任务流中即将需要分配的任务,若 flow_i 和 flow_j 的优先级满足 $\mathrm{Priority}(\mathrm{flow}_i) > \mathrm{Priority}(\mathrm{flow}_j)$,那么 flow_i.next 比 flow_j.next 优先进行分配;若 $\mathrm{Priority}(\mathrm{flow}_i) = \mathrm{Priority}(\mathrm{flow}_j)$,则按照任务流的提交时间次序进行排序,先提交的任务流中的计算任务优先分配。结合任务流集合 F 中各任务流的优先级属性和提交时间,服务器可以确定当前要分配的任务,只需考虑当前任务如何分配即可实现多任务流的调度。

2.5　暂停调度策略

此外,算法还增加了一个暂停调度策略,即在如下两种情况下暂停任务调度:

a)系统中没有空闲的工作站时,暂停任务调度。

b)当对于任意软件资源 $r_i \in R$,若安装有 r_i 的工作站全部处于运行状态,则暂停类型为 r_i 的任务调度。

系统暂停任务调度之后,随着系统的运行,当其中任何一个条件不再满足时,则解除相应的暂停调度约束。在不影响任务及时分配和执行的条件下,通过设置暂停调度策略,相当于起到一个缓冲池的作用,尽可能多地获取系统最新状态,获得更好的分配效果;另一方面也能够适当减少对某些运行中工作站 $DP(p_i, n_j)$ 的计算,提高调度算法的执行效率。

2.6　算法流程

1)更新任务流队列流程

```
任务流优先级队列 q←∅
Procedure update_flowquene begin
if 设计流程 f 为新提交流程 then
  按照优先级次序将 f 插入到 q 的相应位置
    if q 中存在优先级相同的任务流 then
        按照提交时间次序将 f 插入到 q 的相应位置
    end if
/* 当服务器获知某台工作站执行完一个任务之后,会将该任务所在的任务流重新插入到队列中 */
  else
      按照优先级和提交时间将 f 插入到 q 的相应位置
  end if
end procedure
```

2)对任务流中的任务进行分配的流程

```
Procedure scheduling_task begin
Loop do
  if q≠∅ then
    任务流 f←队列 q 的队首
    计算任务 t←f. next
    if t≠NULL and 不满足对 t 暂停调度的条件 then
      计算各工作站执行 t 的效用 utility
      找到 utility 最大的工作站 s
        将 t 分配到工作站 s 上执行
        将 f 从 q 中删除
    end if
  end if
end loop
end procedure
```

3)工作站执行任务的流程

```
Procedure execute_task begin
Loop do
  if 工作站的待执行任务队列不为空 then
    执行队首的计算任务 t
    t 执行完后将其所属任务流 f 重新加入到 q 中
  end if
end loop
end procedure
```

3　实验结果及分析

3.1　实验环境

本文采用的 AUV 协同设计平台实验系统由一台服务器和两台工作站组成,可供多台连接到服务器的客户端访问。该实验系统能够为用户提供基于网页的图形化的任务流创建和提交功能,不同客户端在不同时间提交的任务流能够被服务器接收。本文所提出的 PMTF 算法运行在服务器上,负责将多任务流中的任务分配到合适的工作站上去执行。

为了方便算法性能研究与分析,暂不考虑实际 AUV 设计软件的真实计算结果,仅根据本文中描述的任务特征模拟生成任务节点。工作站 P_1 可以执行两种类型的任务,分别为 r_1 和 r_2;工作站 P_2 仅可以执行种类为 r_1 的任务,即矩阵 $PR =$

$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$。设 r_1 和 r_2 类型任务在单位时间(1 min)内出现的概率为 $Prb(r_1)=Prb(r_2)=0.03$,任务平均执行时间 AET(r_1)和 AET(r_2)均为 1 h。同时考虑到工作站 P_2 的处理速度较快,对于 r_1 类型的任务,P_2 的执行时间是 P_1 的 $\frac{7}{10}$,因此经过测试,将直接回报矩阵和处罚矩阵配置为 $RWD=\begin{bmatrix} 10 & 10 \\ 20 & 0 \end{bmatrix}$, $PTY=\begin{bmatrix} 0.2 & 0.2 \\ 0.15 & 0 \end{bmatrix}$。

3.2 实验结果及分析

分别针对 PMTF、MCT 和 MET 三种任务调度算法进行比较研究。图 3 对比了不同数量任务流全部执行完毕所耗费的时间。基于第 1 章中指出的任务流特性 e)和 f),即单位时间内任务到达概率低以及任务运算时间长(从几十分钟至几十小时不等),仿真实验中的任务流数量设定在 50 条以内,一方面便于获取仿真结果,任务流数量在 50 条以内,仿真时间基本在一周以内,另一方面也符合 AUV 协同设计的实际需求,同时在平台中并行执行的任务流一般不会超过 50 条。纵轴显示了 50 次实验 makespan 的平均值。从图 3 中可以看出,PMTF 算法耗时最少,其次分别是 MCT 和 MET,与 MCT 相比,PMTF 算法耗时减少了 5%左右。图 4 显示了该实验中单个任务流的平均执行时间,同样是 PMTF 算法性能最好,比 MCT 算法减少约 7%。

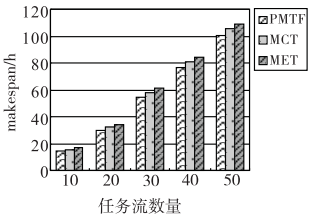

图3 多任务流运行时间比较

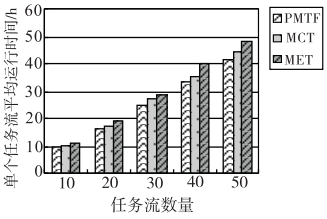

图4 单个任务流平均运行时间比较

从实验结果可以看出,在参数选取适当的情况下,与传统 MCP 和 MET 算法相比,PMTF 算法具有较好的性能。原因是 PMTF 算法采用效用作为任务分配的指标,考虑到了系统的异构性,通过预测到工作站 P_1 接受并执行 r_1 期间因不能接受 r_2 导致的损失,将任务 r_1 较多地分配给了工作站 P_2,从而使得任务 r_2 能够在工作站 P_1 上尽快得以执行,并且通过引入惩罚矩阵,平衡了各工作站的任务负载,进一步提高了工作站的执行效率。此外,引入任务流的优先级,保证了多任务流中的任务能够按照流程优先级属性以及提交的先后次序进行合理的分配。

4 结束语

从 AUV 协同设计平台中多任务流任务调度的实际需求出发,针对计算环境的异构性,以及任务到达率低、执行时间长和任务流具有优先级属性的特殊性,提出了一种基于预测的多任务流调度算法,通过评估各工作站执行任务的效用这一指标进行任务分配,效用综合考虑了执行任务的直接回报、预计损失和负载平衡三方面因素。实验结果表明,在参数选取适当的情况下,与传统 MCT 和 MET 算法相比,PMTF 算法具有更短的运行时间,从而提高了 AUV 协同设计平台的工作效率。

参考文献:

[1] ARMSTRONG R, HENSGEN D, KIDD T. The relative performance of various mapping algorithms is independent of sizable variances in run-time predications [C]//Proc of the 7th IEEE Heterogeneous Computing Workshop. Washington DC: IEEE Computer Society, 1998: 79-87.

[2] FREUND R F, GHERRITY M, AMBROSIUS S, *et al*. Scheduling resources in multi-user heterogeneous computing environments with smartNet[C]//Proc of the 7th IEEE Heterogeneous Computing Workshop. Washington DC: IEEE Computer Society, 1998: 184-199.

[3] MUTHUCUMARU M, SHOUKAT A, HOWARD J S, *et al*. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing system [C]//Proc of the 8th IEEE Heterogeneous Computing Workshop. [S. l.]: IEEE Computer Society, 1999: 30-44.

[4] AUGONNET C, THIBAULT S, NAMYST R, *et al*. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures [J]. Concurrency and Computation: Practice and Experience, 2011, 23(2): 187-198.

[5] SATHAPPAN O L, CHITRA P, VENKATESH P, *et al*. Modified genetic algorithm for multiobjective task scheduling on heterogeneous computing system[J]. International Journal of Information Technology, Communications and Convergence, 2011, 1(2): 146-158.

[6] WEN Yun, XU Hua, YANG Jia-dong. A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system [J]. Information Sciences, 2011, 181(3): 567-581.

[7] HSU C C, HUANG Kuo-chan, WANG Feng-jian. Online scheduling of workflow applications in grid environments[J]. Future Generation Computer Systems, 2011, 27(6): 860-870.

(上接第 1319 页)

[10] QU Yun-yao, WANG Chang-zhou, WANG X S. Supporting fast search in time series for movement patterns in multiples scales[C]//Proc of the 7th International Conference on Information and Knowledge Management. New York: ACM Press, 1998: 251-258.

[11] ABAM M A, BERG M D, HACHENBERGER P, *et al*. Streaming algorithms for line simplification[J]. Discrete & Computational Geometry, 2010, 43(3): 497-515.

[12] SOROUSH E, WU Kui, PEI Jian. Fast and quality-guaranteed data streaming in resource-constrained sensor networks[C]//Proc of the 9th ACM International Symposium on Mobile Ad hoc Networking and Computing. New York: ACM Press, 2008: 391-400.

[13] CULL D E, HILL J, BOUNADONNA P, *et al*. A network-centric approach to embedded software for tiny devices[C]//Proc of the 1st International Workshop on Embedded Software. 2001: 114-130.

[14] JENSEN C S, LAHRMANN H, PAKALNIS S, *et al*. The INFATI data, TR-79[R]. Aalborg: Database and Programming Technologies Institute at Aalborg University, 2004.

[15] HÖNLE N, GROSSMANN M, REIMANN S. Usability analysis of compression algorithms for position data streams[C]//Proc of the 18th International Conference on Advances in Geographic Information Systems. New York: ACM Press, 2010: 240-249.

[16] LANGE R, DÜRR F, ROTHERMEL K. Online trajectory data reduction using connection-preserving dead reckoning[C]//Proc of the 5th Annual International Conference on Mobile and Ubiquitous Systems. 2008: 1-10.