一个网格服务工作流的动态调度算法*

李 超[1,2], 朱巧明[1], 李培峰[1,2], 马峰明[1,2]
(1. 苏州大学 计算机学院, 江苏 苏州 215006; 2. 江苏省计算机信息处理技术重点实验室, 江苏 苏州 215006)

摘 要: 针对服务网格环境中资源的动态性, 提出了一种并行调度算法 PGSWA(parallel grid service workflow scheduling), 该算法引入了性能预测模型和并行就绪队列来预测下一段时间资源的性能并使得成员服务能够并行执行。实验证明, 该算法能较好地缩短工作流的执行时间, 提高工作流的执行性能。

关键词: 网格服务工作流; 动态调度; 性能预测模型; 并行队列
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2008)11-3285-03

Dynamic scheduling algorithm for grid service workflow

LI Chao[1,2], ZHU Qiao-ming[1], LI Pei-feng[1,2], MA Feng-ming[1,2]
(1. *School of Computer Science & Technology, Soochow University, Suzhou Jiangsu* 215006, *China*; 2. *Key Laboratory of Computer Information Processing Technology of Jiangsu Province, Suzhou Jiangsu* 215006, *China*)

Abstract: Considering the dynamics of resources in the service-oriented grid environment, this paper proposed a novel scheduling algorithm PGSWA(parallel grid service workflow scheduling) to solve such problem. The algorithm introduced the performance prediction model and parallel ready queue into the scheduling to predict the resource performance and run services in parallel. The experiment results show that it can reduce the executing time of workflow, and improve its performance.
Key words: grid service workflow; dynamic scheduling; performance prediction model; parallel queue

在服务网格环境中, 所有的资源包括计算、通信和存储等都是以服务的形式存在。同时, 工作流作为实现网格中资源协同工作的一项重要技术手段, 将网格中的多个服务包装成一个更大粒度的服务部署在网格中, 供其他服务或上层的应用访问, 来解决复杂的科学和商业应用。

网格服务工作流可以看成是一组成员服务的集合, 服务之间存在着时序或因果的约束条件, 并最终完成一个特定的目标。它与传统的工作流之间具有很大的差别, 如基于虚拟组织的工作方式; 网格资源的动态性导致网格的计算和处理能力随时间变化, 应用性能估计困难; 资源异构以及跨管理域等。

调度是网格工作流中重要的课题, 并且网格工作流调度不同于一般的任务调度, 在调度时不仅要考虑为任务选择一个最佳的资源, 还要考虑各个任务之间的时序或因果的约束条件, 以及协调各个任务的执行来获取最终的执行结果[1]。

本文提出了一种新的网格服务工作流调度算法。算法引入了性能预测模型能够较好地预测下一段时间资源的性能; 引入了就绪队列和并行就绪队列来提高任务执行之间的并行性; 考虑资源可靠性, 一定程度上保证了工作流的顺利执行。

1 相关工作

在已有的网格资源调度算法中, 主要侧重于解决网格作业与网格资源间的调度问题。文献[2]提出了一种基于经济学模型的优化调度模型, 其目的是在资源提供者和使用者间建立交易, 以尽可能低的费用满足资源使用者进行计算任务的最低要求; 文献[3]采用了一种基于 Class Ad 匹配的集中式方案, 以形成资源调度器; 文献[4]提供了一个高度结构化的可扩展的调度方案。但是这些调度方案未能从网格服务的角度考虑网格服务工作流的调度。

文献[5,6]提出了一种基于遗传算法的网格服务工作流调度算法, 从不同侧面改进了遗传算法, 克服了传统遗传算法收敛性差和早熟收敛的缺陷, 提高了资源选择的效率。但是却没有考虑服务网格环境中资源的动态性和可靠性, 而且没有考虑任务之间的并行性。文献[7]通过对网格工作流中应用程序优先级进行定义, 可以使用优先级调度算法来保证优先级较高的网格工作流应用程序能够被优先调度和运行, 但它也同样没有考虑到任务之间的并行执行, 导致执行效率比较低。

通过具体的实验和模拟实验验证, 本文提出的调度算法能够有效地预测资源节点的性能, 协助调度器获得合理的调度方案; 通过引入关键网格服务和并行就绪队列提高了成员服务之间执行的并行性, 缩短了工作流的执行时间, 提高了工作流的执行性能。

2 网格服务工作流调度模型

2.1 工作流执行模型

工作流系统的核心部件是工作流引擎。它首先解析工作流描述文件, 根据抽象的工作流定义, 获取包含的成员服务及成员服务之间的依赖关系, 这些依赖关系不仅包括数据传输依

收稿日期: 2008-01-13; 修回日期: 2008-03-27 基金项目: 国家自然科学基金资助项目(60673041); 国家“863”高技术研究发展计划资助项目(2006AA01Z147); 江苏省高技术研究资助项目(BG2005020)

作者简介: 李超(1982-), 男, 山东济宁人, 硕士研究生, 主要研究方向为网格工作流调度(lakerlc@163.com); 朱巧明(1963-), 男, 教授, 硕导, 主要研究方向为网格计算、并行计算等; 李培峰(1971-), 男, 副教授, 硕导, 主要研究方向为网格计算、并行计算等; 马峰明(1981-), 男, 硕士研究生, 主要研究方向为网格工作流调度.

赖,还包括控制依赖。

获取了所需的成员服务和依赖关系之后,系统就会将满足执行约束的任务放到就绪队列中,等待系统调度器为其选择一个服务资源并执行。工作流执行模型如图 1 所示。

从图 1 可以看出,工作流执行模型由有向无环图(directed acyclic graphs, DAG) 管理器和调度器组成。DAG 管理器主要负责监视任务的执行,基于任务之间的事件及依赖关系来动态更新就绪队列。调度器主要为就绪队列中的任务选择一个最佳的服务资源,并且传送控制信息和数据信息到服务资源所在的节点,并驱动服务的执行。

2.2 网格服务工作流动态调度模型

网格调度作为网格资源与网格用户之间的桥梁,是网格资源管理系统的重要组成部分。网格调度系统一方面收集大量的可用的资源信息;另一方面根据用户的性能需求将合适的资源分配给应用。根据 GGF 网格调度研究小组,网格调度器负责: a) 发现应用的可用资源; b) 选择合适的资源; c) 执行作业。简而言之,网格调度是由排序和映射组成,排序是根据网格应用的优先级决定网格作业执行的顺序,映射是一个选择合适资源并将其分配给应用的过程,为获得最佳的应用执行性能,每次映射之前都应该进行性能评估。

2.2.1 服务节点性能的预测模型

从 GIS(grid information services) 上获取每个节点的服务资源列表,并采用每隔一段时间重新获取的方式来保证每次都能够较为准确地获取资源信息。

节点的性能是动态变化的,通过综合考虑 CPU、CPU 负载、内存、网络负载以及节点上等待队列的长度这些因素来考查一个节点的性能。通过大量的实验,并对实验结果进行回归统计处理,得出了用来预测某个节点性能的预测模型。

$$M_p(n_i) = \alpha \times n_{i\_cload} - \beta \times n_{i\_nload} - \delta \times n_{i\_c} - \gamma \times n_{i\_m} + \eta \times n_{i\_tnum} + \varepsilon \quad (1)$$

其中: α、β、δ、γ、η是性能因子,分别代表了 CPU 负载、网络负载、CPU、内存及节点上等待队列长度影响节点性能所占的比重; ε为随机误差,均值为零的正态分布随机变量; n_i 表示第 i 个资源节点,且 $1 \le i \le n$; n 为所有节点的数目。

采用的数学模型是岭回归统计,岭统计是一种理论上有较大影响并得以广泛应用的估计方法。对于线性回归模型,回归系数 b 的岭估计定义为

$$\hat{b}(k) = (X^{\mathrm{T}}X + kI)^{-1}X^{\mathrm{T}}y \quad (2)$$

用 b 的岭统计建立的回归方程称为岭回归方程。其中 $k>0$ 称为岭系数。式(1) 中的矩阵 X 和 y 都经过中心化和标准化交换。岭估计随 k(峰系数) 值改变而改变,若记 $\hat{b}_i(k)$ 为 $\hat{b}(k)$ 的第 i 个分量,当 k 在[0, +∞) 上变化时, $\hat{b}(k)$ 的图像称为岭迹。

选择 k 值的岭迹法是: 将 $\hat{b}_1(k)$, $\hat{b}_2(k)$, …, $\hat{b}_m(k)$ 的岭迹画在同一个图上,根据岭迹的变化趋势选择 k 值,使得各个回归系数的岭估计大体上稳定,并且各个回归岭系数估计的符号比较合理。

通过在 MATLAB 中进行回归,发现在 k 取 0.6 时,各个回归系数的岭估计比较稳定,因此将 k 设为 0.6。

为了验证一个预测模型是否合理,是否可以进行较为准确的预测。要对其进行检验,最常用的方法是显著性检验,也称 F 检验。其中统计量计算公式如下:

$$F = [\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2] / [m\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 / (n - m - 1)] \quad (3)$$

在给定的显著性水平 α下,把计算出的 F 分布表中查得的自由度为(m, $n - m - 1$) 的 F_α 值相比,若 $F \ge F_\alpha$,则可以认为因变量 y 与自变量 $x_1, x_2, \cdots, x_m$ 间线性相关关系是显著的,所建回归预测模型有效可用。

根据实验数据和式(1) (2) ,计算得出 F 的值为 5.553 6。在 α取 0.95 时, $F_{0.95}(6, 425 - 6 - 1) = 2.120\ 3$,满足 $F \ge F_\alpha$,证明预测模型是有效的。

获取了节点的性能值,就可根据各个节点的性能值生成一个按性能值非递减排列的资源列表。

2.2.2 资源的可靠性定义

利用 Re(n_i) 代表资源节点 n_i 的可靠性,只有当资源的可靠性达到一定标准时才会有可能成为候选资源,表示为在该资源上能成功运行某服务的概率。因此对于用户而言,体现了资源安全可靠执行服务的概率。采用式(4) 计算:

$$\mathrm{Re}(n_i) = \alpha \times \mathrm{succ}_{n-1}^{s_i} + \beta \times 1/(n-1)\sum_{k=1}^{n-1}\mathrm{succ}_k^{s_i} \quad (4)$$

其中: $\mathrm{succ}_k^{s_i} = \begin{cases} 1 & n_i \text{ 上 } s_i \text{ 运行成功} \\ 0 & n_i \text{ 上 } s_i \text{ 运行失败} \end{cases}$ 表示了在资源 n_i 上运行服务 s_i 是否成功, α、β是权重因子且 α + β = 1,分别代表了最近一次执行情况和历史执行情况所占的比重。

同样, 利用 Re (s_i) 来代表服务资源 s_i 的可靠性, 其计算式为

$$\mathrm{Re}(s_i) = \partial \times \mathrm{succ}_{n-1}^{s_i} + \beta \times 1/(n-1)\sum_{k=1}^{n-1}\mathrm{succ}_k^{s_i} \quad (5)$$

其中: $\mathrm{succ}_k^{s_i} = \begin{cases} 1 & s_i \text{ 执行成功} \\ 0 & s_i \text{ 执行失败} \end{cases}$ 代表了 s_i 是否执行成功。有了每个 s_i 的可靠性约束,调度器在选择资源时就可以根据 Re (s_i) 的大小来筛选服务资源。

2.2.3 服务估计执行时间的定义

$T_{\mathrm{est}}(s_i, n_i)$ 用来表示服务 s_i 在节点 n_i 上的估计执行时间,计算式为

$$T_{\mathrm{est}}^{n}(s_i, n_i) = \lambda T_{\mathrm{est}}^{n-1}(s_i, n_i) + (1 - \lambda)[T_{\mathrm{exc}}^{n-1}(s_i, n_i) + \beta_{n-1}] \quad (6)$$

$$\beta_n = \sqrt{1/(n-1)\left(\sum_{k=1}^{n}T_{\mathrm{exc}}^{k}(s_i, n_i) - \left(\sum_{k=1}^{n}T_{\mathrm{exc}}^{k}(s_i, n_i)\right)^2/n\right)} \quad (7)$$

$$\lambda = \begin{cases} T_{\mathrm{exc}}/T_{\mathrm{est}} & (T_{\mathrm{est}} > 2T_{\mathrm{exc}}) \\ 1 - T_{\mathrm{exc}}/T_{\mathrm{est}} & (T_{\mathrm{exc}} < T_{\mathrm{est}} \le T2\,T_{\mathrm{exc}}) \\ T_{\mathrm{est}}/T_{\mathrm{exc}} & (T_{\mathrm{est}} \le T_{\mathrm{exc}}) \end{cases} \quad (8)$$

其中: $T_{\mathrm{est}}^{n}(s_i, n_i)$ 表示 s_i 在资源 n_i 上第 n 次执行时的估计执行时间, β_n 是标准方差,反映了在该资源上执行该服务实例的历史变动情况。 $\lambda T_{\mathrm{est}}^{n-1}(s_i, n_i)$ 代表了前一次的估计执行时间对当前执行时间的影响, $(1 - \lambda)[T_{\mathrm{exc}}^{n-1}(s_i, n_i) + \beta_{n-1}]$ 则代表了历史执行情况对当前执行时间估计的影响。λ是根据最近一次的估计时间和执行时间动态确定的,用于控制这两部分对目前时间估计的影响力,并且 λ的取值是动态确定的,更加适应了网格环境的动态性。

2.3 就绪队列和并行就绪队列

就绪队列中的任务是可以执行的活动。在工作流执行时,工作流中的活动单元(即处于就绪状态的成员服务) 可能被分成 n 个就绪队列,笔者用{ RL_1, RL_2, …, RL_n} 代表执行时的 n 个就绪队列。 n 的大小和每个就绪队列中的活动单元是在运行时确定的。每个就绪队列中活动单元之间是独立的,并可以并行执行,不同就绪队列中的活动单元根据约束条件可能并行

或串行执行, 如图 2 所示。

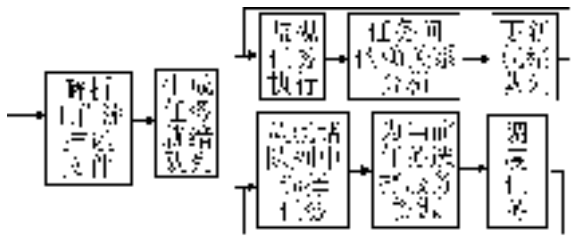

图 1　工作流执行示意图

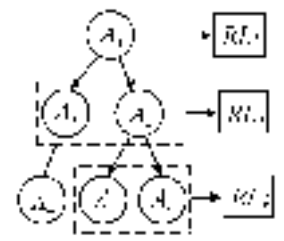

图 2　算法队列调度图

如图 2 所示, RL_i 和 RL_j 只能串行的执行, 因为只有 RL_i 和 RL_j 存在着序列约束。如果 A_3 能够在 A_2 之前执行完, 则 RL_j 和 RL_k 可以部分并行执行。

工作流开始执行的时间用 T_s 表示, 结束时间用 T_e 表示。分析可知, 工作流的执行时间取决于就绪队列的执行和就绪队列之间的并行度。并行度越高, 则执行效率越高。为了定义并行就绪队列, 首先引入关键成员服务 CGS(critical grid service) 。

2.3.1　关键服务成员的定义

假设 L 代表 RL_k 的长度, 假设 $S_{i,k}$ 表示就绪队列 RL_k 中 S_i 服务, 若满足如下性质, 则将其定义为关键网格服务 CGS。

$T_{\mathrm{CGS}} = \max\{ T_{\mathrm{est}}^{s_i,k} \mid 1 \leq k \leq L\}$ 。其中 $T_{\mathrm{est}}^{s_i,k}$ 是 RL_k 中 s_i 的估计执行时间。

由上可知, CGS 的执行时间的快慢决定了所在就绪队列 RL_k 的执行时间的快慢, 在一定意义上也就决定了整个工作流的执行时间快慢。

2.3.2　并行就绪队列的定义

通过定义并行就绪队列, 就可以确定哪些就绪队列可以并行执行, 从而提高工作流任务的执行效率, 缩短工作流的执行时间。

引入队列的开始执行时间 $T_{\mathrm{sta}}^{RL_k}$ 和执行结束时间 $T_{\mathrm{end}}^{RL_k}$, 定义如下:

$$T_{\mathrm{end}}^{RL_k} = T_{\mathrm{sta}}^{RL_k} + T_{\mathrm{CGS}} \tag{9}$$

对开始时间和结束时间进行了粗略的计算。其中 T_{CGS} 表示关键服务的估计执行时间, 利用式(6) 计算获得。

如果 RL_i 和 RL_j 满足下式, 则表明两个就绪队列是可以并行执行的。

$$\begin{aligned}&(T_{\mathrm{sta}}^{RL_j} \leq T_{\mathrm{sta}}^{RL_i} < T_{\mathrm{end}}^{RL_j}) \vee (T_{\mathrm{sta}}^{RL_j} < T_{\mathrm{end}}^{RL_i} \leq T_{\mathrm{end}}^{RL_j}) \vee \\ &(T_{\mathrm{sta}}^{RL_i} \leq T_{\mathrm{sta}}^{RL_j} < T_{\mathrm{end}}^{RL_i}) \vee (T_{\mathrm{sta}}^{RL_i} < T_{\mathrm{end}}^{RL_j} \leq T_{\mathrm{end}}^{RL_j})\end{aligned} \tag{10}$$

2.4　算法实现

根据上述的分析和定义, 针对服务工作流调度问题, 提出并行调度算法——PGSWA(parallel grid service workflow scheduling) 算法。该算法描述如下:

输入: 工作流应用$\{A_1, A_2, \cdots, A_n\}$, 资源节点可靠性阈值 NC, 服务资源可靠性阈值 SC;

输出: 成员服务与服务资源间的映射关系;

a) 获取就绪队列$\{\mathrm{RL}_1, RL_2, \cdots, RL_n\}$, 并获取每个队列的关键任务;

b) 通过式(11) 计算来获取并行队列;

c) 根据系统设定的 NC 来筛选筛选资源节点;

d) 通过预测模型获取可用资源节点的性能值;

e) 根据需求考虑就绪队列中成员服务的时间和成本约束, 根据设定的 SC, 为处于就绪状态的成员服务分配最佳的可用的服务资源。

3　实验验证

在已有的网格调度算法中, 最短作业最快资源 (SJFR)[8] 和最长作业最快资源 (LJFR)[8] 是两种常用算法, 前者可以获得较短的总体作业执行时间, 但会导致长作业执行时间很长; 而后者虽然避免了长作业饿死, 但作业执行的总体时间也比较长。

为了验证 PGSWA 算法的性能, 提交的工作流由 1 ～20 个成员服务组成, 每次实验时提交 5 ～10 个工作流, 记录每种情况下工作流的平均执行时间。实验环境为: 一个主机作为调度节点, 八个性能不同的主机作为资源节点, 并且每个节点上都部署了分词服务、人名识别服务、地名识别服务、机构名识别服务和词性标注服务。实验结果如图 3 所示。

实验表明, 基于 PGSWA 算法的工作流的执行时间低于 SJFR 算法和 LJFR 算法产生的执行时间。当工作流包含的成员服务较少时, 三种算法的执行时间相差不大, 这主要是因为处于就绪状态的任务较少, 任务区分度较小。随着成员服务数量的增加, PGSWA 由于考虑了任务之间的并行执行, 能获得较好的执行性能。

此外, 为了验证调度算法的性能, 通过在 GridSim 中进行模拟实验与 FCFS、LJFR、SJFR 调度算法进行了比较, 并设置了五种不同的工作流结构, 每种工作流由 5 ～50 个成员服务组成。在实验中设定了 10 个资源节点, 每个节点包含一样的服务资源列表。两种调度算法的实验结果如图 4 所示。

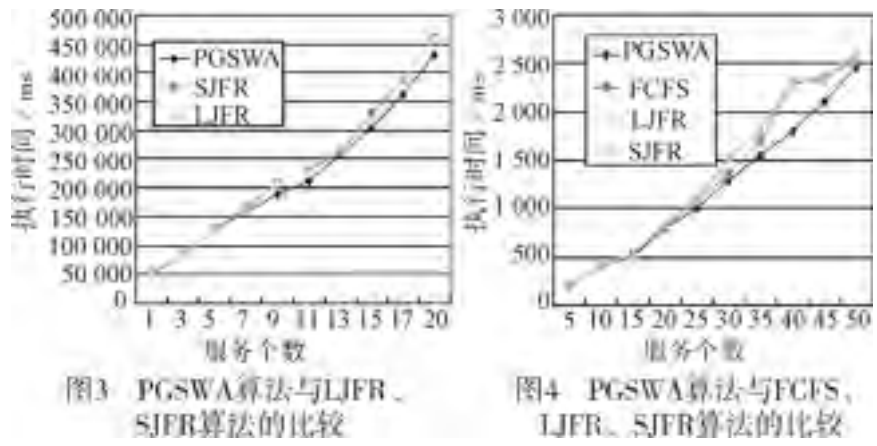

图 3　PGSWA 算法与 LJFR、SJFR 算法的比较　　图 4　PGSWA 算法与 FCFS、LJFR、SJFR 算法的比较

由图 4 可知, 在模拟环境中, PGSWA 算法表现出较好的性能, 执行时间均小于 FCFS、LJFR 和 SJFR 算法下的平均执行时间。工作流的成员服务数量比较少的情况下, 几种算法下工作流的执行时间相差不大; 当成员服务的数量超过 30 时, 执行时间出现了较大的区别, 主要是由于 PGSWA 算法考虑了服务就绪队列中服务之间执行的并行性, 使得满足条件的服务能够并行的执行, 从而提高了执行效率。

4　结束语

针对服务网格环境下的网格工作流调度, 本文提出了一种新型的工作流调度模型, 通过引入性能预测模型, 较好地预测资源的性能; 通过引入并行就绪队列, 提高了任务之间的并行执行。通过实验验证, 该调度模型能够较好地缩短工作流的执行时间, 保证工作流的顺利执行。但该调度方法仅仅考虑了资源的性能和可靠性, 没有考虑用户的 QoS[9] 以及系统多层次的 QoS; 而且调度方法中还没有提出一种较好的容错机制来保障工作流的安全执行。这些将是下一步研究的重点和难点问题。

(下转第 3295 页)

S_t = Resample(S_t) ; // 重采样
返回 S_t

3　实验结果

在开放系统[8] 的基础上笔者开发了一个 RBPF-SLAM 算法实验平台。其运行界面如图 1 所示。为了验证本文提出的算法,分别对开放数据库 Radish[9] 中提供的两组不同数据记录利用算法 1 和 2 进行了处理,然后对各算法的结果进行了对比。在实验过程中,算法 1 和 2 的所有参数设置相同。其中粒子数目均为 20,算法 2 的固定滞后长度为 5。实验结果分别如图 2 和 3 所示。

图1　RBPF SLAM算法实验平台

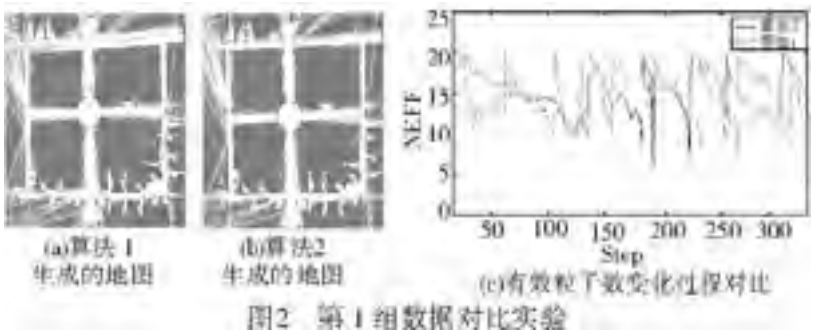

图2　第 1 组数据对比实验

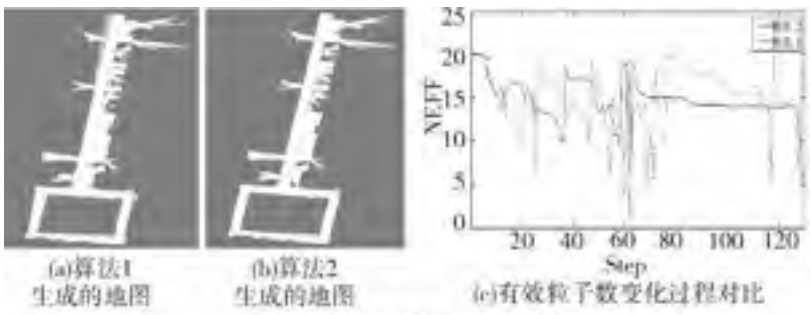

图3　第 2 组数据对比实验

从实验结果来看,在相同粒子数目的条件下,进一步改进后的算法 2 比原算法 1 有明显的改进: a) 环路闭合更加准确; b) 在运行过程中有效粒子数目有所改善。但是, 算法 2 也存在着一定的不足,虽然在空间复杂度方面没有什么增加,但是时间复杂度增加了不少,这也是进一步研究的方向之一。

4　结束语

本文对采用栅格地图表示法的 Rao-Blackwellized 粒子滤波 SLAM 算法进行了改进,分析了采样和重采样操作对 RBPF-SLAM 算法估计一致性的影响,设计了一种基于滞后采样策略——固定滞后 Gibbs 采样的改进 RBPF-SLAM 算法。对比实验表明,这种滞后采样策略都能在一定程度上改善 RBPF-SLAM 算法的综合性能: 在采用相同粒子数目的情况下,重采样次数显著减少,地图估计一致性也显著提高。但是该算法存在着时间复杂度偏高的缺点,当 L 的取值较大时不一定能满足实时的要求,需进一步对算法优化。

参考文献:

[1] MITH R, SELF M, CHEESEMAN P. A stochastic map for uncertain spatial relationships[C]//Proc of the 4th International Symposium on Robotic Research. Cambridge: MIT Press, 1987: 467-474.

[2] MONTEMERLO M, THRUN S, KOLLER D. FastSLAM: a factored solution to the simultaneous localization and mapping problem[C]// Proc of National Conference on Artificial Intelligence. Edmonton, Canada: AAAI Press, 2002: 593-598.

[3] BAILEY T, NIETO J, NEBOT E. Consistency of the FastSLAM algorithm[C]//Proc of IEEE International Conference on Robotics and Automation. San Francisco: IEEE Press, 2006: 424-429.

[4] GRISETTI G, STACHNISS C, BURGARD W. Improved techniques for grid mapping with Rao-Blackwellized particle filters[J]. IEEE Trans on Robotics, 2007, 23(1): 34-46.

[5] MURPHY K P. Bayesian map learning in dynamic environments[C]// Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2000: 1015-1021.

[6] DOUCET A, GODSILL S, ANDRIEU C. On sequential Monte Carlo sampling methods for Bayesian filtering[J]. Statistics and Computing, 2000, 10(3): 197-208.

[7] LIU J. Metropolized independent sampling with comparison to rejection sampling and importance sampling[J]. Statistics and Computing, 1996, 6(2): 113-119.

[8] GRISETTI G, STACHNISS C, BURGARD W. GMapping[EB/OL]. (2007). https://svn.openslam.org/data/svn/gmapping.

[9] HOWARD A, ROY N. The robotics data set repository[EB/OL]. (2003). http://radish.sourceforge.net.

(上接第 3287 页)

参考文献:

[1] HU Qiao-ming, LI Pei-feng, GONG Zheng-xian, *et al*. ADJSA: An adaptable dynamic job scheduling approach based on historical information[C]//Proc of the 2nd International Conference on Scalable Information Systems. 2007: 201-207.

[2] BUYYA R, ABRAMSON D, GIDDY J. An economy driven resource management architecture for a global computational power grid[C]// Proc of Int'l Conf on Parallel and Distributed Processing Techniques and Applications. 2000: 259-264.

[3] FREY J, TANNENBAUM T, FOSTER I, *et al*. Condor-G: a computation management agent for multi-institutional grids[J]. Cluster Computing, 2002, 5(3): 237-246.

[4] CHAPIN S, KARPOVICH J, GRMSHAW A. The Legion resource management system[C]//Proc of the 5th Workshop on Job Scheduling Strategies for Parallel Processing. 2000: 123-128.

[5] DONG Fang-peng, SELIM G. An adaptive double-layer workflow scheduling approach for grid computing[C]//Proc of the 21st International Symposium on High Performance Computing Systems and Applications. Washington DC: IEEE Computer Society, 2007: 156-163.

[6] TRETOLA G, ZIMEO E. Activity pre-scheduling in grid workflows[C]//Proc of the 15th Euromicro International Conference on Parallel, Distributed and Network-based Processing. Washington DC: IEEE Computer Society, 2007: 245-253.

[7] 刘洋,桂小林,徐玉文. 网格工作流中基于优先级的调度方法研究[J]. 西安交通大学学报, 2006, 40(4): 411-414.

[8] ABRAHAM A, BUYYA R. Nature's heuristics for scheduling jobs on computational grids[C]//Proc of the 8th Int'l Conf on Advanced Computing and Communication. 2000: 301-308.

[9] YUAN Ying-chun, LI Xiao-ping, WANG Qian. Dynamic heuristics for time and cost reduction in grid workflows[M]. Berlin: Springer-Verlag, 2007: 499-508.

[10] 郭文彩,杨扬. 基于遗传算法的网格服务工作流调度的研究[J]. 计算机应用, 2006, 26(1): 54-56.

[11] 王勇,胡春明,杜宗霞. 服务质量感知的网格工作流调度[J]. 软件学报, 2006, 17(11): 2341-2351.