异构分布式环境中 Alhusaini 资源映射算法的改进

廖剑伟　余建桥

（西南大学计算机与信息科学学院　重庆 400715）

摘　要　为异构分布式系统中提供良好的资源映射算法，可以有效、加速部署在分布式系统中的应用程序的执行，其中 Alhusaini 算法是该研究领域中成功和具有影响力的算法之一。本文针对该算法的不足提出了一个两阶段动态资源映射的改进算法：第一个阶段仅仅收集数据以提供给第二阶段使用；在第二个阶段选择一组独立的任务并根据这些任务的权值将它们映射到相关资源中去。改进的算法有效地解决了 Alhusaini 算法存在的算法复杂度过高和在通信密集型应用中算法效率低等问题。

关键词　异构分布式系统，Alhusaini 算法，两阶段动态资源映射

Improved Algorithm for Alhusaini's Method in Heterogeneous Distributed Systems

LIAO Jian-Wei　YU Jian-Qiao

(College of Computer and Information Science, Southwest University, Chongqing 400715)

Abstract　Resource mapping algorithm for an application on heterogeneous distributed system (HDC) will promote the execution capability of it. In most of mapping Algorithms for application in HDC, the Alhusaini's method is one of the most important Algorithms. We propose a two-phase algorithm called 2-phases dynamic resource co-allocation algorithm (2PDRCA) based on Alhusaini's method. The first phase only generates the data that will be used in the second phase. The second phase will selected a set of independent tasks and allocate according to the weight of each task in our method. The simulation results show that the method is effective, and solves the problem such as Low efficiency of Alhusaini's method in communication intension application.

Keywords　Heterogeneous distributed systems, Alhusaini's algorithm, 2-phases dynamic resource co-allocation algorithm (2PDRCA)

1　引言

各种不同资源通过高速网络连接在一起并向用户或者应用程序提供了一个统一应用平台的系统，称为异构分布式系统(Heterogeneous Distributed Computing, HDC)。资源映射的任务是为 HDC 中分布式应用程序中的任务分配合适的资源，并调度这些任务按照一定的顺序完成，使得资源利用率和应用程序的执行时间都达到最佳的效果。

Alhusaini 是该研究领域中最有影响的学者之一[7]，他提出的资源映射算法[7,8]较有代表性，其算法核心是在信息汇集阶段进行资源映射，但是由于在信息汇集阶段无法确定任务在实际的执行过程中释放资源供其它任务使用的确切时间，从而导致整个应用无法在最优调度时间长度内完成[9]。本文在 Alhusaini 算法[8]的基础上提出两阶段动态资源映射解决思路，在第一阶段产生用于第二阶段的数据，而真正的映射操作在第二阶段完成。

2　Alhusaini 算法

假设有 m 台计算设备，$M=\{m_1, m_2, \cdots, m_m\}$ 和 r 种非计算资源，$R=\{r_1, r_2, \cdots, r_r\}$，一起构成 HDC 系统，在该 HDC 中运行的分布式应用程序 T 分解为 n 个子任务 $T=\{T_1, T_2, \cdots, T_n\}$。

定义 1　用 $Exec(T_i, m_j)$ 表示机器 m_j 上完成 T_i 的计算代价，包括通信时间和计算时间。

定义 2　当任务 T_i 所需要资源与 T_j 所需的资源的交集不为空时，表示任务 T_i 和 T_j 是不相容。

定义 3　给定的一组任务，$T=\{T_1, T_2, \cdots, T_n\}$ 及其资源需求，相容图表示为 $g=(V, E)$，其中：

V 是任务 T 的集合；只有当工作 T_i 和工作 T_j 是不相容时，才存在 $e(i,j)$。

定义 4　加权相容图(Weighted Compatible Graph, *WCG*) $W=(T,E,C)$，其中：

T 表示任务集；E 是一组边的集合，每条边连接了在 T 中的每两个任务，如 $e(i, j)$ 表现了工作 T_i 和工作 T_j 之间的数据依赖性；C 是一个与 E 相关的函数，c_{ij} 表示从任务 T_i 和任务 T_j 共同所需的资源数目。

Alhusaini 算法的基本思想是当一个任务在运行阶段不需要在使用资源 r_k 时，则释放该资源。每个映射事件都会以下面的过程完成：一个多个任务的集合 S 被选择执行时，首先选择处于该集合中调度顺序最优先的就绪任务开始执行，直到 S 中的所有任务都按照调度次序逐一完成。对于每个任务 T_i，它在映射事件发生时，首先完成自身所需时间最短的机器 m_b(称之为最优机器)，寻找一个最优机器是通过式(1)进行机器的比较，只要条件式满足，就说明任务 T_i 可以在机器 m_b 上面执行。

$$Exec(T_i, m_b) \leqslant Exec(T_i, m_j) + \Delta Exec(T_i, m_j) \quad (1)$$

廖剑伟　硕士，助教，研究方向：网格计算、分布式计算；**余建桥**　博士，教授，研究方向：数据库技术、人工智能。

其中 Δ 是从 0%到 100%的值，表示任务 T_i 在机器 m_b 机器上执行的可能性。

Alhusaini 算法通过尽早释放资源，在信息汇集阶段为有向无环图 DAG 中的任务 T_i 随机分配到系统中的某台机器中，然后算法根据迁移条件决定是否将任务 T_i 从机器 m_j 迁移到机器 m_b 中。如果迁移条件不满足，可能造成任务 T_i 不会被分派到机器 m_b 中。通常在分布式异构环境中迁移条件是很难满足的。如果任务 T_i 没有被分派到机器 m_b 中，那么 T_i 完成时间就不会是最优时间。

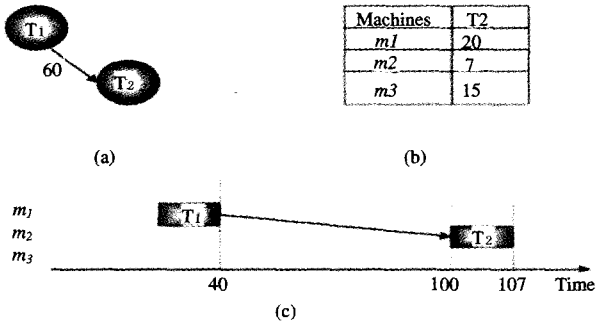


图 1 (a)部分任务的 DAG；(b)任务 T2 在各台机器上完成的估算时间；(c)调度顺序

同时，在迁移条件中并没有考虑 T_i 和它的前驱之间的通信代价，这将会造成任务调度完成所需的时间变长，在通信较为密集的应用中更为突出。例如在图 1 中假设有 3 台机器，任务 T1 在运行阶段分配到了机器 m1 中。先假设 T2 在信息汇集阶段被分派到了机器 m2 中，Alhusaini 算法将会找到机器 m1，然而当验证迁移条件发现即使当式(1)中 Δ 取值为 100%，迁移条件依然不成立。因此任务 T2 将会被分派到机器 m2 并在单位时间 107 后完成。如果任务 T2 被分派到机器 m1 中，完成它的时间将会是单位时间 60。

而且在通信密集型应用中任务需要进行迁移，那么为了测试迁移条件是否满足，就必须与其它机器进行通信，这样势必会使得本来通信量非常大的应用的通信变得更为紧张。而且 Alhusaini 算法的测试迁移条件关心的也仅仅是计算资源，也就是说即使通过测试满足迁移条件后，直到任务迁移到机器上面以后才发现其并不能够获得足够的非计算资源而得以执行。

3 二阶段动态资源映射算法(2PDRCA)

为了解决 Alhusaini 算法存在的不足，提出了基于 Alhusaini 算法的改进两阶段动态资源映射算法 2PDRCA，该算法分为信息汇集阶段和运行分派阶段，资源映射是在运行分派阶段才完成。

3.1 信息汇集阶段

在信息汇集阶段只是收集信息并经过标记处理，然后提供数据，供运行分派阶段使用。主要工作只有两步：第一步是构建加权相容图 WCG，加权相容图的定义为定义 4 所示。第二步是标明所有的任务在其执行过程是否需要非计算资源。图 2 是该阶段的部分伪代码。

```
Compiler time phase
Input: DAGs and their resource requirement
Output: WCG
Begin
1. Construct the weighted compatible graph for the DAGs
2. Mark all tasks that do not require any non-compute resource
End
```

图 2 信息汇集阶段的部分伪代码

3.2 运行分派阶段

该阶段主要是根据信息汇集阶段提供的数据完成资源映射和任务执行，执行 G 表示属于一个或者多个应用程序的任务之间的资源共享关系，图 3 给出算法该阶段实现的部分伪代码。

```
Run time allocation phase
Begin
1. Let WCG be the weighted compatible graph generated in the compiler time phase
2. While (counter < total number of tasks) do:
3.     At each mapping event do:
4.         Put all ready state tasks into the set READY
5.         Extract all tasks that are marked at compiler time from READY and put them
into set M
6.         Construct set C from READY
7          …
8.         Find a machine mj that has earliest finish time to t
9.         Allocate R(t) to t
10.        Execute t on machine mj
```

图 3 运行分派阶段的部分伪代码

每个任务就绪状态的必备条件是它所有的前驱任务都已经完成，而且它所需要的所有资源都能满足。在运行分派阶

段最为重要的操作就是选择最大的独立集合 S，而完成该操作的时间复杂度是 $O(2C^{C}R)$，其中 C 是不包括关键任务 Vc 的独立集合中的任务的数目，R 表示非计算资源的数目。下面简单描述在 2PDRCA 中构建最大独立集合的基本步骤：

a)当任务 Ti 不需要任何非计算资源且当它处于就绪状态时，将该任务加入到最大独立集 S 中。

b)选择一个关键任务集，该集合中的任务数是一个小于 k(k 满足 2^k k 小于任务总数)常数 C。如果选择较多的关键任务，就必须为这些关键任务预分配更多的非计算资源。因此，与关键任务共享非计算资源的其它任务就会受到资源限制，不能进入就绪状态，则把这些任务从最大独立集合中删除。

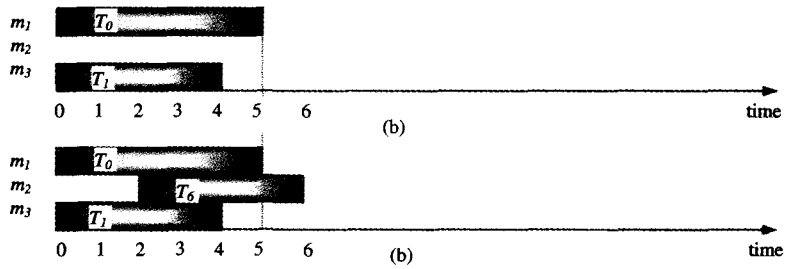
c)在从 READY 集合中选择最大独立集合的时候，必须首先取出在信息汇集阶段进行了标注的任务，这样可以有效地减少从 READY 集合中构建最大独立集合的时间复杂度，因此可以将这些任务直接加入到最大独立集合中去。

d)当某个任务在 READY 集合中具有最大权值，那么将它选择加入到最大独立集合中。之所以这样对待最大权值的任务 T_i，因为通常在就绪状态中权值最大的任务的后继任务比较多，而这些后继任务也有可能要使用 T_i 所需要的资源，那么 T_i 越早完成，就可以从就绪集合中选择更多的任务去执行，这样可以减少构建最大独立子集的时间复杂度。

e)当任务 T_i 的映射事件发生以后，根据该原则选择其它的关键任务，直至关键任务数达到 C。当选择了关键任务以后，还需要从 READY 集合中选择其它任务加入到最大独立集合 S。

2PDRCA 采用最大权值的关键任务优先，而其它任务采用随机选取的原则相结合的方法，一方面尽可能达到资源映射的要求，另一方面也大大降低了时间复杂度和减少资源映射所需的时间。图 4 给出了一个例子，任务 T0 和任务 T1 都处于就绪状态，因为 T0 不需要任何的非计算资源，那么它会被加入到最大独立集合，如图 4(b)所示。当任务 T1 在时间单位 2 的时候释放了资源 1，图 4(c)就是映射事件发生后的情形。

	T_0	T_1	T_2	T_3	T_4	T_5	T_6
M_1	5	5	4	7	8	3	3
M_2	7	6	4	6	7	8	4
M_3	4	4	5	5	6	5	3



(a) 任务的估算计算代价；(b) 最大权值任务映射事件发生后；(c) 在时间单位 2 的映射事件发生后

图 4

4 分析与比较

4.1 算法时间复杂度比较

2PDRCA 通过把任务 T_i 分派到能在最短时间内完成该任务的机器，避免了在通信密集型应用时调度时间长度的增加；Alhusaini 算法中，当 HDC 系统的异构性比较大时，迁移条件变为假的可能性就越大，因此也就很难将任务 T_i 分配给系统中最优的机器 mb。2PDRCA 中，在运行阶段直接将任务 T_i 分配给机器 mj，有效避免了 Alhusaini 算法的迁移机制可能带来的调度时间变长的问题。

在 Alhusaini 算法的信息汇集映射阶段，选择最大独立集合的时间复杂度是 $O(2N^NR)$[7]，其中 N 任务总数，R 是资源总数。在运行自适应阶段，Alhusaini 算法的时间复杂度是 $O(N^2R)$。在 2PDRCA 的信息汇集阶段，主要工作就是构建任务加权相容图 WCG，构建 WCG 的时间复杂度是 $O(N^2R)$[12]，在 2PDRCA 的运行分派阶段的时间复杂度是 $O(N^2R)$。比较结果如图 5 所示。

	信息汇集阶段	运行分派阶段
Alhusaini 算法	$O(2N^NR)$	$O(N^2R)$
2PDRCA	$O(N^2R)$	$O(N^2R)$

图 5 Alhusaini 算法和 2PDRCA 的时间复杂度

4.2 实验与仿真

为了描述方便，下面给出仿真实验中主要参数及其定义。

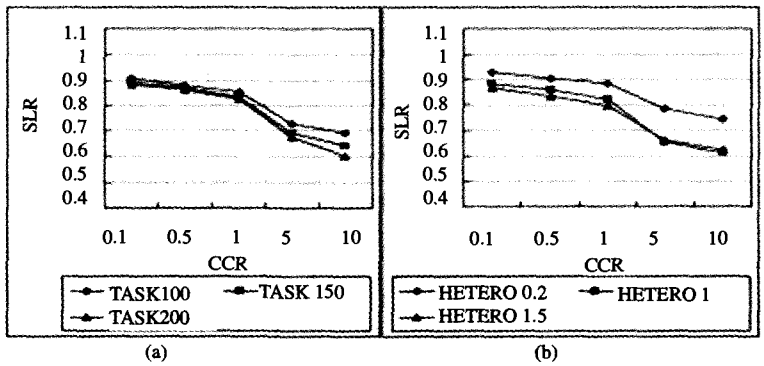
DAG 图中通信与计算比值(Communication to computation ratio, CCR)：当一个 DAG 的 CCR 比较低时，可以认为是计算密集型应用，反之则是通信密集型应用；

各台机器中计算代价的比例范围(HETERGENEITY)；

定义调度时间长度率(Schedule Length Ration, SLR)：2PDRCA 的平均调度时间与 Alhusaini 算法的平均调度时间比值。当 SLR 大于 1，意味着 Alhusaini 算法调度时间更少；如果 SLR 是小于 1.0，那么就说明 2PDRCA 平均调度时间更小。

为了更好地说明 2PDRCA 算法的效能，利用 Simgrid

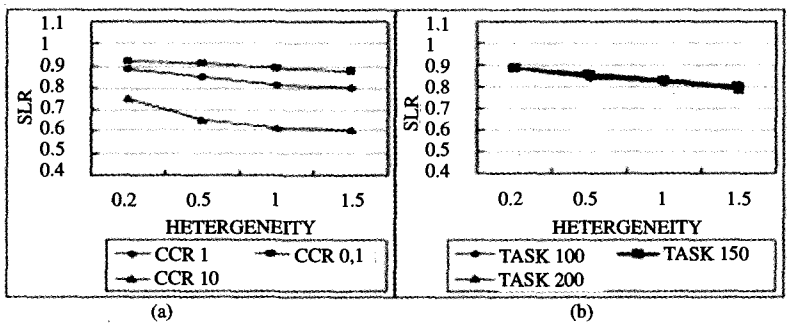
Toolkit[10]开发了调度模拟器。Simgrid Toolkit 提供了一系列核心函数，用以建立特定应用领域的模拟器。这些函数包括资源描述(如处理机和存储资源)以及任务描述(如数据传输、计算和依赖关系等)，同时提供了诸如调度、顶测、时间和模拟的功能函数。通过随机生成的主机处理能力，存储资源和通信延迟可以描述实际的异构式分布式环境[10]；测试在一台 CPU 为酷睿双核 T2300(1.66GHz)、内存 512M 的主机上完成，每个测试结果均取 3 次模拟的平均值。当任务数 TASK{100,150,200}和 HETEROGENEITY {0.2,1,1.5}变化时观察到 CCR 结果{0.1,0.5,1,5,10}，仿真结果如图 6 所示。



(a) TASK 变化时 CCR 的仿真结果；(b) 当 HETEROGENEITY 变化时 CCR 的仿真结果

图 6

在通信密集型的应用中，减少任务之间通信代价而进一步缩短调度时间长度。但是在 Alhusaini 算法的迁移情况下，只有把计算代价作为运行阶段是否迁移的依据。因此，调度时间长度由于通信代价的剧增反而会增加。在 2PDRCA 中没有采用迁移机制，即使 2PDRCA 没有采用特殊的机制来应对通信密集型应用，SLR 值仍然小于 1，即可以有效地减少调度时间长度。



(a) CCR 变化时 HETEROGENEITY 的仿真结果；(b) TASK 变化时 HETEROGENEITY 的仿真结果

图 7

在图 7 中可以看到 HETEROGENEITY 对 SLR 的影响。通过上述两个实验可以发现，当 HETEROGENEITY 增加时，SLR 会减少。通过实验观察了参数 TASK，当在对应于一个应用的 DAG 中有更多的任务时，那么测试迁移条件为假的可能性就越大。因此，当任务数增加时，SLR 就会减少。

结论 与 Alhusaini 算法相比较，2PDRCA 不需要迁移机制，直接在运行阶段利用加权函数进行任务的分派。通过仿真可以发现，与 Alhusaini 算法相比，它可以更有效地减少调度时间，尤其在通信量密集的应用中；从效率来说，Alhusaini 的方法和 2PDRCA 在运行阶段复杂度都是 $O(N^2R)$，其中 N 是任务数，R 是非计算资源数。但是在信息汇集阶段，相比 Alhusaini 算法复杂度是 $O(2^NNR)$，2PDRCA 算法的复杂度是 $O(N^2R)$。

进一步主要研究具有不同拓扑结构和延迟的局域网的全连接网格[6]的资源映射算法，同时将考虑算法支持多计算资源的使用和资源预定。

参 考 文 献

1 Wang L Z, Cai W T, Lee B S, et al. Resource co-allocation for parallel task s in computational grids [A]. In: Proceedings of the International Workshop on Challenges of Large Application s in Distributed Environments [C]. Seattle: IEEE Computer Society, 2003. 88～95

2 Yang Juan, Bai Yun, Qiu Yuhui. A decentralized resource allocation policy in minigrid [J]. Future Generation Computer Systems, 2006

3 Rehn C. Dynamic mapping of cooperating tasks to nodes in a distributed system [J]. Future Generation Computer Systems, 2006. 35～45

4 Sanyal S, Jain A, Das S K, et al. A hierarchical and distributed approach for mapping large applications to heterogeneous grids using genetic algorithms [A]. In: Proceedings of the IEEE International Conference on Cluster Computing [C]. Hong Kong: IEEE Computer Society, 2003. 496～499

5 Azzedin F, Maheswaran M, Arnason N. A synchronous co-allocation mechanism for grid computing system s. Cluster Computing [J], 2004, 7(1): 39～49

6 Liao Jianwei, Cai Hongbin, et al. Design and Implementation of Grid Monitoring System based on GMA [C]. In: The sixth Inter-

national Conference on Parallel and Distributed Computing, Applications and Technologies, December 2005. 94～96
7 Alhusaini A H, Prasanna V K, Raghavendra C S. A framework for mapping with resource co-allocation in heterogeneous computing system [C]. In: 9th Heterogeneous Computing Workshop, May 2000. 273～286
8 Alhusaini A H, Prasanna V K, Raghavendra C S. Run-Time Adaptation for Grid environments [C]. In: Proceedings 15th International Parallel and Distributed Processing Symposium, April 2001. 864～874
9 Topcuoglu H, Hariri S, Wu Min-You. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Trans on Parallel and Distributed System, 2002, 13: 260～274
10 Simgrid C H. A toolkit for the simulation of application scheduling [C]. In: Proceedings of the 1st IEEE international Symposium on Cluster Computing and the Grid (CCGrid'01), 2001
11 丁箐，陈国良，顾钧. 计算网格环境下的一个统一的资源映射策略[J]. 软件学报，2002, 13(7): 1303～1308
12 李慧贤，程春田. 一种并行的网格资源协同分配方法[J]. 大连理工大学学报，2005, 45(2): 272～277

(上接第 63 页)

3.2 软件设计

软件设计分为两个部分：手持设备端和主机端。手持设备端完成数据采集和传送工作，主机端完成数据处理工作。手持端的软件设计要求是能够以足够的精度对传感器、按键电路进行检测，然后将信息打包发送。所以手持设备端的软件以轮询的方式对按键输入，传感器输入采样，并将信息按照协议打包，通过串口发送至主机。软件流程图如图 5。

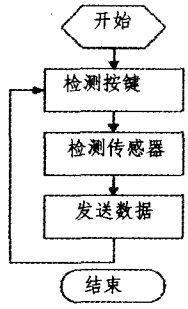


图 5 发送端流程图

该循环每执行一次的时间为 10ms，也就是每 10ms 对传感器和按钮输入采一次样。我们采用的是机械式按钮，需要软件消除抖动。每隔 10ms 访问一次按钮，可以满足软件消抖的需要。角速度传感器反映的是用户手臂的角速度，以 10ms 采样，每秒钟可以采样 100 次。我们实地测量，用户完成一次指向滑动动作，时间在几百毫秒到几秒之间，所以这样的采样精度已可以重现动作信息。手持设备端的软件将用户在每个 10ms 内的手臂运动角速度采集并发送给主机端，主机端则根据第 2 节中我们讨论过的算法，完成光标定位和按键响应的任务。

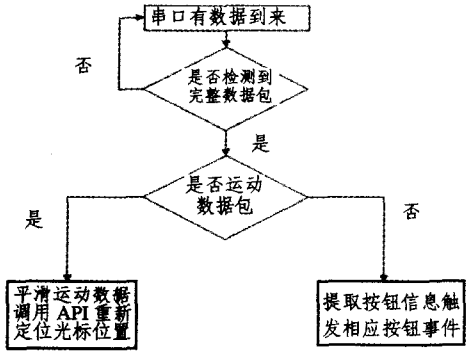


图 6 接收端流程图

主机端的监视串口输入的数据，分解数据，提取出用户的动作信息和按钮输入信息，依据算法完成光标定位和按钮响应。主机端为 windows 平台，通过进程检测串口输入数据，每检测到完整的数据包就对其进行分解，抽取出用户的动作角速度和按钮输入。对于按钮输入的响应可以调用 windows API 中的 SendInput 函数，根据按钮的输入状态向 window 的消息队列发送标准鼠标的左键或右键。按下或弹起或双击消息，系统便会在光标当前位置上响应这一消息。对于动作角速度，由于用户的手臂悬空操作，因此会有手臂的无意识抖动信息包含在其中，需要做平滑处理。即在每次收到数据后，将数据和前几次收到的数据做均值，用均值来代替当前用户的手臂角速度，然后使用我们在第 2 节讨论过的算法完成光标定位。

主机端程序的处理流程见图 6。

4 使用前景和存在的问题

基于动作检测的交互模式允许用户直接和屏幕交互，交互的模式是符合用户直觉的，学习和掌握的过程并不复杂，适合应用于大屏幕交互场合，如商务展示或是家用大屏幕显示设备的交互。

这种交互模式也存在一定的局限性，检测动作的传感器存在性能上的误差，所以用户的过于细微的动作并不能被全部探测到，这在使用时会给用户带来一定的限制。经实际测试后，我们认为可以用多次调整拖动的方式完成细微定位。

结束语 本文介绍了基于动作检测的交互模式，并阐述了这种交互模式的实现原理和实验模型的实现方法。基于动作检测的交互模式是为大尺寸屏幕交互环境设计，用户可以直接用手臂动作把光标拖动到希望到达的位置，不需采用传统的指点设备进行间接定位，从而提高了操作效率。

参 考 文 献

1 Cheng K, Polo K. Direct Interaction with Large-Scale Display Systems using Infrared Laser Tracking Devices. School of Information Technologies, The University of Sydney NSW, Australia, 2006
2 Oh Ji-Young, Stuerzlinger W. Laser Pointers as Collaborative Pointing Devices. Department of Computer Science, York University. http://www.cs.yorku.ca/～wolfgang
3 MacKenzie I S, Jusoh S. An Evaluation of Two Input Devices for Remote Pointing. Department of Mathematics & Statistics, York University, Toronto, Ontario, Canada M3J1P3; Department of Computer Science University of Victoria, Victoria, British Columbia, Canada V8W 3P6
4 Fuhrman T, Klein M, Dahl M O. Blue wand. A versatile remote control and pointing device. Institute furs Telemetric Universidad Karlsruhe, Germany
5 Soukoreff R W, MacKenzie I S. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. Department of Computer Science and Engineering, York University, 4700 Keele Street, Toronto, Ont, Canada M3J 1P3