多核处理器缓存结构技术综述

王力生，周成君，陈丽果
同济大学电子与信息工程学院，上海　200092

摘　要　随着半导体制造工艺的发展，晶体管集成度迅速提高。多核心，片上内存管理，动态功耗等技术已经大量的出现在各种处理器上。强大的计算能力背后，更需要强大的缓存系统支持。因此，为适应多核心高速数据共享，高效的数据预取和写回，片上存储系统的规模也越来越大，结构越来越多样。近几年，从缓存结构和数据存储策略的角度思考缓存效率逐渐成为热点。

关 键 词　多核；缓存效率；缓存结构；算法优化
中图分类号 TP39　　**文献标识码** A　　**文章编号** 1674-6708（2011）54-0174-04

0 引言

随着半导体工艺的进步，单片处理器上的晶体管集成度迅速提高。这为制造具有更大容量的片上缓存提供了技术基础。同时，由于多核处理器的普及，随之而来的是对缓存效率、数据一致性以及功耗方面的全新考虑。就目前的处理器设计来看，一味的追求缓存容量，并不能得到很好的性能提升，反而对芯片面积和功耗提出了考验。

1 对称缓存结构

目前，多核处理器使用的对称缓存结构主要有两种基本的片上存储系统模型，即 Streaming Memory 和 Coherent Cache。

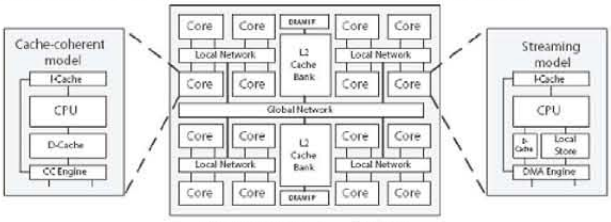

图 1 CMP Cache 分类 [1]

1.1 Coherent Cache

所有片上存储资源都用于构成处理器内核的私有 Cache 和内核间的共享 Cache，并通过硬件保证数据的一致性。处理器内核隐式地从 Cache 中读写数据，Cache 控制器根据内核的访存要求以“块”（block）为单位从片下存储器（内存）中读取和替换 Cache 中的数据。Cache 的数据存取、访问同步、数据一致都由硬件实现，程序员无法干预，程序可以直接寻址的存储资源是片下存储器。

1.2 Streaming Memory

部分片上存储资源被组织成独立的可被程序寻址的存储结构（local storage），与片下存储结构的数据交换需要程序控制和 DMA 操作。在这个模型下，处理器内核从 local storage 中读写数据，local storage 通过 DMA 操作与片下存储器（或更低层次的共享 Cache）交换数据，local Storage 中的数据存放位置、存取粒度和替换策略都可以由应用程序控制。

1.3 性能比较

Jacob Leverich 等对两种模型进行了性能比较 [1]，结论如下：

1）对于存在大量复用数据的并行应用程序，两种模型的性能接近。对于某些 benchmark，Streaming Memory 比采用写分配策略的 Coherent Cache 性能高 10%~25%，但采用非写分配策略的 Coherent Cache 会削弱这个优势；

2）对于不存在复用数据的应用程序，随着处理器内核数目和计算能力的提高，Streaming Memory 的优势越来越明显。采用硬件预取技术可缩小 Coherent Cache 与 Streaming Memory 的性能差距。在某些需要数据冗余复制的应用中，Coherent Cache 甚至比 Streaming Memory 的性能要好。

纯粹的 Streaming Memory（即没有经过软件优化）不存在显著的性能优势；但 Streaming Memory 对于软件设计影响很大，特别是当内核数目很大时，对软件进行合理优化会显著提高性能。

2 非一致的缓存结构

非一致缓存结构最初是 2002 年由 Changkyu Kim 等人在参考文献 [2] 中提出的，起初用于解决由于线延迟提高带来的 Cache 访问周期过长的问题。随着 CMP 的普及，人们越来越重视 NUCA 在 CMP 上的应用，即构建非一致的 Coherent Cache 结构。

2.1 NUCA（Non-Uniform Cache Architecture）结构

随着 Cache 容量的扩大，Cache 的访问延迟越来越受到数据的存放位置的影响，离处理器近的数据的访问延迟要远远低于离处理器远的数据。例如，在 50nm 的制造工艺下，对于一个 16M 的片上 L2 Cache，处理器访问距离它最近的数据单元需要 4 个时钟周期，而访问最远的数据单元需要 47 个时钟周期 [2]，在传统的 UCA（Uniform Cache Architecture）中，Cache 的访问延迟由距离处理器最远的数据单元决定，即需要 47 个时钟周期，这势必会严重影响处理器速度，成为处理器速度提升的瓶颈。NUCA 结构正是针对这种情况提出的一种片上 Cache 的组织方式。其基本思想就是将 L2 Cache 组织为多个 Cache bank，每个 bank 可以单独访问，这样处理器访问距离它近的 bank 的速度就快于距离它远的 bank，从而可以降低访问 Cache 的平均延迟。

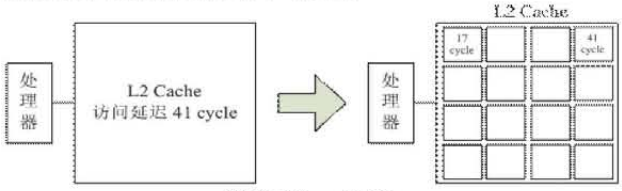

图 2 UCA vs. NUCA

2.2 静态 NUCA

在静态 NUCA 中，每个 Cache bank 可以单独进行寻址，数据根据其在下层存贮器中的地址静态地映射到某一个 bank 中。相对于 UCA，Static NUCA 有两个优势：1）距离 Cache controller 近的 bank 可以获得较低的访问延迟；2）对不同 bank 中数据的访问可以并行进行。Static NUCA 根据访问线路的设计不同可以分为 Private Channel based 和 Switched Channel based 两种。

在 Private Channel 中，每个 Cache bank 有两个专用的数据通道（channel）分别用于数据的读写，从而可以保证每个 bank 的读写都相互独立，这样可以提高数据访问的并行性。但它的问题在于，当 Cache 的划分粒度很小时，会有很多 bank，从而需要很多数据通道，这将增加芯片的面积。

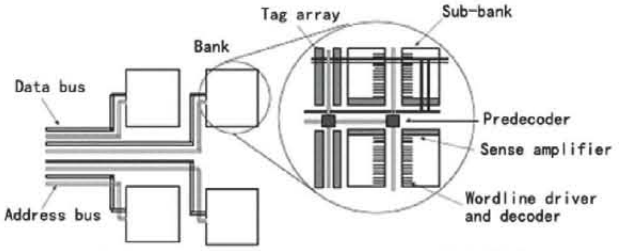

图 3 Private Channel based Static NUCA 缓存设计

在 Switched Channel 中，不为每一个 bank 设计专用通道，而是用一个 2-D mesh 网络将所有的 bank 连接起来，采用分组交换的方式在 Cache controller 与 bank 之间传输数据。这样可以节省芯片面积，但对于数据密集型的应用程序可能造成互连网络繁忙和拥塞。

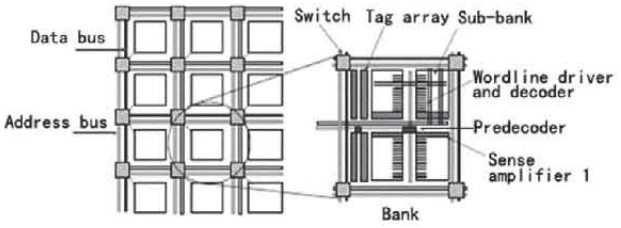

图 4 Switched Channel based Static NUCA 缓存设计

2.3 动态 NUCA

在动态 NUCA 中，并不是将数据完全静态的存放在一个固定的 Cache bank 中，而是可以根据一定的策略动态的调整数据的位置，在不同的 Cache bank 中进行移动，从而保证常用的数据存放在距离 Cache controller 较近的 bank 内。为此，需要考虑数据映射（mapping）、数据定位（locating）和数据迁移（movement）3 个方面的策略。

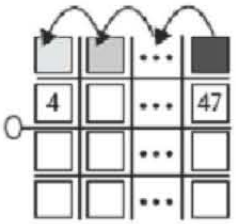

图 5 Dynamic NUCA 中的数据迁移

2.3.1 数据映射

数据映射 (mapping) 是指从下一级存储设备中导入的数据可以被存放在哪些 Cache bank 中，一种极端的做法是，根据一个数据在下一级存储设备中的地址将它映射到一个固定的 bank 中，数据不能移动，这和 static NUCA 没有差别；另一个极端是，数据可以在 bank 之间任意存放和移动，这种策略会使硬件和寻址开销达到最大。

在文献 [2] 中，Kim 等提出了一种折中的策略，将 NUCA 与传统的组相联映射相结合，即将所有的 bank 分为多个组，传统的组相联映射方式中的一组数据只能被映射到一个 bank 组中，且每个字放在不同的 bank 中，之后它也只能在本 bank 组内移动，这要求每个 bank 组中 bank 个数与组相联的映射方式相对应。根据 bank 分组方式的不同可以分为 simple mapping，fair mapping，shared mapping。

在 simple mapping 中，把 Cache bank 排列为二维阵列，其中每一列作为一个 bank 组。搜索数据的时候可以依 bank 组 ->bank-> 字的顺序进行搜索。这个策略有两个缺陷：1）bank 阵列中的行数不一定与组相联映射的策略相匹配；2）每个 bank 组的访问延迟仍不一致。

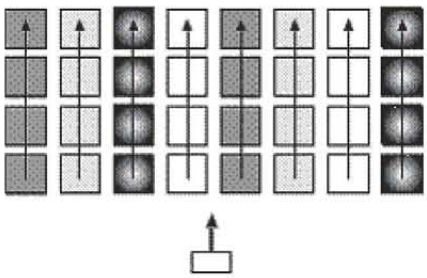

图 6 Simple mapping

为弥补 simple mapping 的缺陷，fair mapping 被提出，其思想是尽量使每个 bank 组的平均访问延迟相等（图 7）。其优点是对每个 bank 组的平均访问延迟近似相等。缺点是读写数据的循径逻辑复杂。

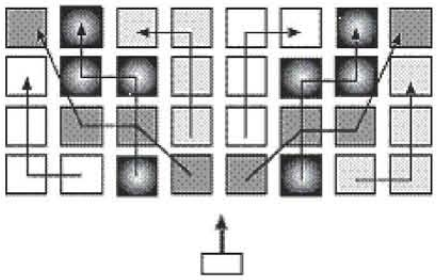

图 7 Fair mapping

shared mapping 通过允许数据在最靠近控制器的一排 bank 中移动，力图使每个 cache bank 组都有最快的访问延迟。但这样做 bank 阵列的行列数和组相联映射的策略有严格的匹配要求。

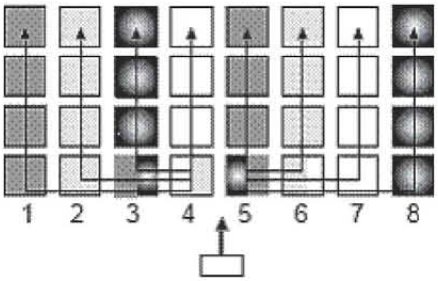

图 8 Shared mapping

2.3.2 数据定位

数据定位（locating）是指如何确定处理器访问的数据在 cache 中的位置，NUCA 提供两种：incremental search，从距离 cache controller 最近的 bank 开始查找，直到在一个 bank 中找到，或到最后一个 bank 仍未找到，返回 miss 信号；multicast search，地址信号被广播到所在 bank 组的多个或所有 bank 中，并行查找。

另外，还可以将这两种策略混合，即先在 n 个 bank 的前 m 个并行查找，之后在下面 m 个中查找……直到最后一个 bank。

2.3.3 数据迁移

Dynamic NUCA 的目标是尽量使处理器对 cache 的访问都发生在距离 cache controller 最近的 cache bank 里，因此 cache 内数据的移动应尽量使最近访问的数据出现在距离 cache controller 近的 bank 中。因此人们很自然的想到使用 LRU 算法，使距离 cache controller 最近的 bank 中保存着最近使用的数据，但传统的 LRU 算法对导致过多的数据移动。

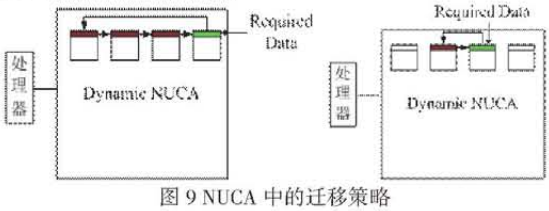

图 9 NUCA 中的迁移策略

如果处理器要访问距离它最远的 bank 中的数据，根据传统的 LRU 算法，该相联组内所有的数据都要依此移动，这会给片上网络带来很大压力（图 9 左）。因此，Kim 等人改进了 LRU 算法，使命中的数据只向处理器近的方向移动一个 bank 的距离（图 9 右），这样既能减轻片上网络的压力，又可以是经常访问的数据距离处理器越来越近。

3 Dance Hall 结构

Dance Hall 结构（图 10）[3] 是一种完全共享的二级缓存结构，这种结构实现起来比较简单，可采用“写直达”或“延迟写”来保持一、二级缓存的数据一致性，为保证一级缓存间的数据一致性，可采用监听协议或者目录协议 [3]。由于所有处理器共享整个二级缓存，所以可以最大程度的利用二级缓存空间，以获得非常高的命中率。

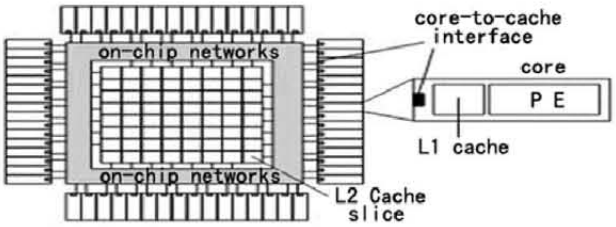

图 10 具有 Dance Hall 结构的 64 核处理器

但从图 10 中可以看到，当某个处理器从二级缓存中读取数据时，会因为数据块距离处理器的远近而导致读取延迟有所不同 [2]，这种延迟称为线延迟，在 dance hall 结构中，由于线延迟的问题使得平均命中延迟非常高，这大大的降低了缓存结构的效率。在文献 [4] 中介绍了一种降低线延迟的策略，这种策略通过控制数据块的迁移，使某个处理器频繁使用的数据块能够存放在离该处理器较近的地方，但是这种策略不能从根本上解决命中延迟高的缺点。因为处理器没办法预先知道哪些数据是它要频繁使用的，它只能根据运行时的状况进行预测，这种亡羊补牢的方法只能在一定程度上缓解问题的严重性。

4 Cache in Middle 结构

在文献 [5] 中，Jaehyuk Huh 等对 NUCA 用于 CMP 做了深入的探讨，其思想是在将 L2 cache 划分为多个 cache bank 之后，通过互连网络与处理器内核连通，cache 阵列分布在芯片中心，处理器内核均匀排列在 cache 周围（Cache In Middle，CIM 结构）。

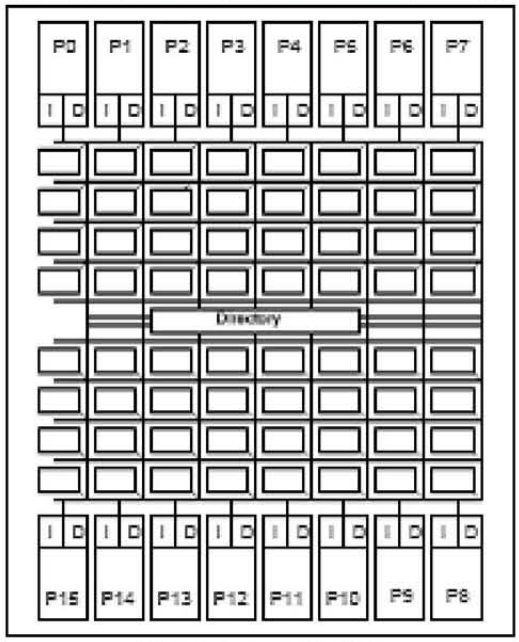

图 11 Cache in Middle 结构

作为多核 L2 cache 的 NUCA 结构与单核的 NUCA 有所不同，主要研究重点在以下几个方面：

1）共享度（sharing degree）

共享度，即将 cache 划分为不同的 bank 后，每个处理器内核可以访问所有的 bank，还是只能访问部分 bank。

当 sharing degree 为 1 时，每个内核只能访问其自己的 L2 cache，这相当于每个内核有一个私有 cache，当 sharing degree 为 2 时，两个内核共享以部分 cache bank，以此类推，当 sharing degree 等于内核数时，表示每个内核都可以访问所有的 cache bank。

共享度对 cache 性能的影响主要集中在命中率（hit rate）、访问延迟（hit latency）、核间数据传输和保持一致性的复杂度等方面。较小的共享度可以降低访问延迟；较大的共享度可以提高命中率，减少核间传输；共享度对 cache 一致性的影响要从两方面看，如果共享度较小，保持 L2 cache 的一致性比较复杂，如果共享度较大，保持 L1 cache 的一致性比较复杂。

Jaehyuk Huh 等人通过实验证明，当共享度为 4 的时候，系统的性能最高，各方面参数都能达到一个平衡。

2）Mapping Algorithm

单核的 NUCA 中采用 static mapping 和 Dynamic Mapping 两种策略，其中 Dynamic Mapping 可以弥补 static mapping 的某些不足。但多核的 NUCA 给 Dynamic mapping 也带来了一些问题，比如当多个核心都访问相同数据时可能导致数据迁移冲突，即数据可能在两个核心间来回不断移动；另外，传统的 central tag 也不适用于多核处理器，因为 tag 阵列无法距离每个内核都很近。

3）移动策略

单核的 NUCA 中数据只允许沿一个坐标方向移动，因为数据只要沿一个方向移动就可以控制其与处理器的距离。但在多核环境下，数据要根据处理器的请求在多个坐标方向上移动，因此多核的 NUCA 应支持这种移动。

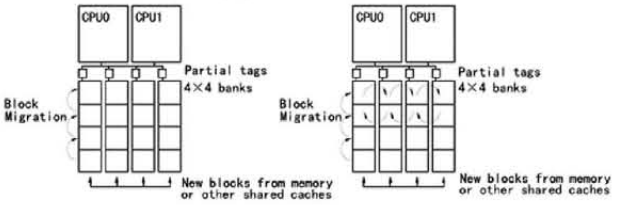

图 12 移动策略

5 Cooperative Cache 结构

Jichuan Chang 等在文献 [6] 中提出 Cooperative Cache（CC）结构。CC 基于私有的二级缓存结构，但是允许每个处理器访问片上的任意处理器的私有二级缓存，这样把常用的数据放在本地的私有缓存中，当本地私有缓存的大小不能满足任务对于命中率的要求时，也可以把自己的数据放在其他处理器的二级 Cache 中，这样既可以保证常用数据具有较低的命中延迟，也可以通过使用其他处理器的本地 Cache 来达到相对满意的命中率，从而避免了从主存中取数据的开销，因为毕竟主存延迟相对于缓存间的线延迟来说还是非常可观的。Michael Zhang 等在 [7] 中提出了一种类似于 CC 的方法，它的出发点与 CC 基本上是相同的，只是在实现细节上有所不同，我们把它们归结为同类算法。

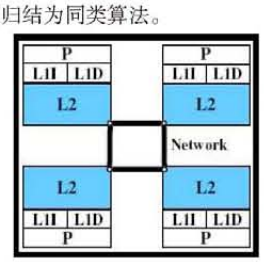

图 13 Cooperative Cache 结构

但是，这类方法没有解决动态的缓存资源共享的平衡问题，而且当处理器 a 直接使用处理器 b 的本地私有二级缓存，但此时

处理器 b 的二级缓存资源也相对稀缺的时候，会对处理器 b 的效率产生很大的影响，极端情况下，作用在 Cache 系统的调度策略可能会导致系统颠簸。即大部分的系统时间都用在了 Cache 替换上。产生这种问题的主要原因是因为系统没有对 Cache 空间以及存储带宽的分配给予软件或者硬件上的支持，Ravi Iyer，Fei Guo 等在文献 [8-9] 中专门探讨了针对这种问题的解决方案。

Bradford M 等在文献 [10] 中提出了一种对 CC 的改进方法。由于 CC 在向局部 L2 中迁移数据的算法是静态的，迁移数据能降低命中延迟，但是由于数据副本增加不但降低了 L2 的命中率，同时也提高了保持数据一致性的复杂度，所以迁移数据的静态算法在某些时候可能对性能产生相反的效果。文中提出了一种动态的 ASR 算法，仅当进行数据迁移后，低的线延迟带来的性能提高大于低 L2 命中率所带来的性能损失时，该算法才进行数据迁移。这种方法靠着复杂的协议来提高性能，当将来 CMP 规模越来越大时，这种方法显然就不太合适了。

6 结论

本文就目前存在的处理器缓存结果进行了简单阐述。对称缓存结构中，Coherent Cache 结构的寻址和数据一致性操作均由硬件实现，减轻了程序员的负担；但是硬件设计相对复杂。Streaming Memory 结构为程序员提供了更多的自由，程序员可以控制数据的存取位置和内核之间的通信，消除了内核通信和数据一致性的硬件开销；但是提高了程序设计的复杂性，程序员必须显式的控制数据的存取和内核间的通信。相比之下非一致的缓存结构显示出一定的优势。目前，各处理器厂商都在自己的产品上综合采用各种缓存技术。这些技术都需要为处理器增加或多或少的额外逻辑线路，有的技术甚至需要操作系统的支持。所以，随着处理器制造工艺的发展，越来越多地从结构和算法角度考虑缓存效率，而不是单纯增大片上缓存容量，必然是将来处理器功耗领域的发展趋势。

参考文献

[1] Jacob Leverich, Hideho Arakida, Alex Solomatnikov, Amin Firoozshahian, Mark Horowitz, Christos Kozyrakis. “Comparing Memory Systems for Chip Multiprocessors”. ISCA’ 07, June 9-13, 20.

[2] Changkyu Kim, Doug Burger, Stephen W. Keckler; An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches; October. 2002, ACM SIGPLAN Notices, ACM SIGARCH Computer Architecture News, ACM SIGOPS Operating Systems Review, Proceedings of the 10th international conference on Architectural support for programming languages and operating systems ASPLOS-X, Volume 37, 30, 36 Issue 10, 5, 5.

[3] L. Barroso et al. Piranha: a scalable architecture based on single-chip multiprocessing. In ISCA-27, May 2000.

[4] Beckmann, B. M.; Wood, D. A.; Managing Wire Delay in Large Chip-Multiprocessor Caches; Microarchitecture, 2004. MICRO-37 2004. 37th International Symposium on 04-08 Dec. 2004 Page(s): 319-330.

[5] Jaehyuk Huh, Changkyu Kim, Hazim Shafi, Lixin Zhang, Doug Burger, Stephen W. Keckler. “A NUCA Substrate for Flexible CMP Cache Sharing”. Appears in the 19th International Conference on Supercomputing. ICS’ 05, June 20-22, 2005.

[6] Jichuan Chang; Sohi G. S; Cooperative Caching for Chip Multiprocessors; Computer Architecture; 2006. ISCA ’06. 33rd International Symposium on, 2006 Page(s): 264-276.

[7] Michael Zhang, Krste Asanovic; Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors; May 2005, ACM SIGARCH Computer Architecture News, Proceedings of the 32nd annual international symposium on Computer Architecture ISCA ’05, Volume 33 Issue 2.

[8] Ravi Iyer, Li Zhao, Fei Guo, Ramesh Illikkal, Srihari Makineni, Don Newell, Yan Solihin, Lisa Hsu, Steve Reinhardt; QoS Policies and Architecture for Cache/Memory in CMP Platforms; June 2007, Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS ’07.

[9] Fei Guo, Hari Kannan, Li Zhao, Ramesh Illikkal, Ravi Iyer, Don Newell, Yan Solihin, Christos Kozyrakis; From Chaos to QoS: Case Studies in CMP Resource Management; March 2007, ACM SIGARCH Computer Architecture News, Volume 35 Issue 1.

[10] Bradford M. Beckmann, Michael R. Marty, David A. Wood; ASR: Adaptive Selective Replication for CMP Caches; December 2006, Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture MICRO 39.

(上接第200页)

状态预测，观测更新，状态增广 3 个阶段，获得一个描述较为准确、完整的环境地图。

3 实验分析与结论

实验环境为四周是垂直墙壁的不规则线形走廊，P3-DX 机器人沿走廊运动，用里程计和超声波传感器构建环境地图。

从实验过程可以看出，在移动机器人运动环境中有墙壁、开或关闭的门以及角落等不同环境特征，随着机器人的不断运动，走廊环境地图被逐渐创建出来，且准确度较高。实验结果显示，x 和 y 校正量保持在 ±50*mm* 以内，θ 的校正量在 ±5 度以内。此外，机器人局部位姿有些时刻校正量较大，分析其原因，主要是基于以下几点：启动阶段 Δx，Δy 较大是因为机器人由静止到运动对控制器有较大冲击造成的；当机器人运动中转变方向时，受移动机器人平台点镇定控制器的影响，机器人位姿 x，y，θ 的校正量同时增大；环境地面状况的局部不平整也会造成位姿误差的突增。

实验结果表明基于里程计与超声波传感器的同时定位与地图构建算法，基本消除了里程计的累计误差，纠正了机器人的位姿，提高了定位的精度，可以快速的完成机器人的环境地图构建任务。

参考文献

[1] Thrun S, Bayesian Landmark Learning for Mobile Robot Localization [J]. Machine Learning, 1998, 33(1): 41-76.

[2] Kenneth D. Harries, Michael Recce. Absolute localization for a mobile robot using place cells [J]. Robotics and Autonomous Systems, 1997, 22(3): 393-406.

[3] Greg Welch, Gary Bishop. An Introduction to the Kalman Filter. [EB/OL]. http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf., 2007-1-16.

[4] 梁靓，黄玉清，张玲霞，李想. 机器人的差分方向控制与实现[J]. 信息与电子工程，2004，9.

[5] 梁靓，黄玉清. 基于嵌入式PC的机器人光电寻线系统[J]. 西南科技大学学报，2004，1.