

一种支持多种访存技术的 CBEA 片上 多核 MPI 并行编程模型

冯国富¹⁾ 董小社¹⁾ 胡 冰¹⁾ 王旭昊¹⁾ 王恩东^{2),3)}

¹⁾(西安交通大学计算机科学与技术系 西安 710049)

²⁾(高效能服务器和存储技术国家重点实验室 济南 250013)

³⁾(浪潮电子信息产业股份有限公司 北京 100085)

摘 要 现有的 CBEA(Cell Broadband Engine Architecture)编程模型多侧重于支持类似于流处理的“批量访存”(Bulk Data Transfer)应用,传统非规则访存应用性能较低.文中基于 Cell 架构提出了一种同时支持“批量访存”与非规则访存应用的 MPI 并行编程模型,将通信分解在 PPE(PowerPC Processing Element)上,拓宽模型的适用范围;在统一访存接口下,通过运行时访存剖分信息指导选择和优化访存以提高计算效率.实验结果表明,文中提出的编程模型支持多种访存模式并具有很好的并行加速比,可获得较同类相关技术 30%~50%左右的性能提升.

关键词 异构多核;CBE 架构;并行编程模型;MPI;访存技术;剖分优化

中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2008.01965

A MPI Parallel Programming Model for CBEA Based on Hybrid Memory Access Technology

FENG Guo-Fu¹⁾ DONG Xiao-She¹⁾ HU Bing¹⁾ WANG Xu-Hao¹⁾ WANG En-Dong^{2),3)}

¹⁾(Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049)

²⁾(State Key Laboratory of High-end Server & Storage Technology, Jinan 250013)

³⁾(Langchao Electronic Information Industry Co. Ltd., Beijing 100085)

Abstract Most programming models for CBEA (Cell Broadband Engine Architecture) well support bulk data transfer application which is suitable for stream processing, but the applications whose memory access patterns are irregular or unpredictable can not be supported or suffer performance degradation. This paper proposes a MPI programming model and corresponding runtime library to support both streaming and irregular applications. The MPI communication was assigned on PPE (PowerPC Processing Element) side to broaden the applicable field of the model. Moreover, a runtime memory access profiling infrastructure under uniform access interface was adopted in the model to help programmer to select proper memory access method and optimize data transfer between different memory hierarchies. The experimental results show that, besides getting high speedup ratio and supporting various memory methods, the application based on pro-

posed parallel programming model performs about 30%~50% better than that based on related technologies.

Keywords heterogeneous multi-core; CBE architecture; parallel programming model; MPI; memory access technology; profile optimization

1 引言

功耗和散热问题限制了处理器主频的提高,多核正成为流行架构,国际上诸多研究机构与处理器厂商正在积极研究多核技术.中国科学院在成功推出龙芯 1 号^[1]和龙芯 2 号^[2]处理器后,也开始积极研究并准备推出多核处理器.

异构多核技术通过在片上简化单个核设计,增加专用协处理核个数,以较小的功耗和成本获得更高的计算性能.基于 CBEA(Cell Broadband Engine Architecture)的 Cell^[3-4]是由 STI(SONY、Toshiba、IBM)设计的一种典型异构多核处理器,Cell 不仅可用于娱乐用途同时也可用于科学计算^[5].在 2008 年 6 月公布的 TOP 500 中,名列榜首的千万亿次超级计算机 Roadrunner 即采用 CBE 架构的 PowerX-Cell 8i.在 3.2GHz 主频下单个 Cell 处理器可提供超过 200Gflops 的单精度浮点运算能力,并通过片上 204.8GB/s 的高速互连总线 EIB(Element Interconnect Bus)与主存和 IO 相连. Cell 处理器内置一个通用 PPE(PowerPC Processing Element)和 8 个 SPE(Synergistic Processing Element),PPE 和 SPE 具有不同的 ISA(Instruction Set Architecture). Cell 处理器的计算能力主要来自 SPE,每个 SPE 有 256KB 可直接寻址的片上(on-chip)本地存储 LS(Local Storage).SPE 只能直接访问 LS 中的数据和代码,当 SPE 上运行的计算代码和数据大小超过 256KB 限制时,需将部分数据和代码放在片外(off-chip)主存,必要时通过 DMA 操作从片外主存中获取.目前, Linux 可以支持基于 Cell 的平台,抢占式 SPE 任务调度功能已被集成入 Linux 源代码树.

虽然以 CBEA 为代表的异构多核架构具有很高的理论计算峰值和片上传输带宽,但除面临与通用多核架构类似的任务级并行问题外,异构多核架构还面临着使用不同 ISA 和显式管理多级存储层次^[6-7]等因素带来的一系列可编程性与性能问题.本文通过在 Cell 平台下研究实现 MPI 标准通信库和相关运行时支持向用户提供 MPI 并行编程模型接

口,同时针对 CBEA 的多层存储结构通过在底层提供支持多种访存技术充分挖掘架构的性能.

2 相关工作

IBM 的 CBE SDK 提供 PPE 和 SPE 两种 ISA 的 C 编译器及相关多线程管理控制、软件管理 Cache(Software Managed Cache)^[8-10]、Code Overlay 等基本编程支持,但直接基于 CBE SDK 进行 Cell 编程还相对比较困难.目前多数 Cell 编程模型建立在 CBE SDK 基础上,主要可分为如下两类:

第一类包括随 CBE SDK 一起发布的 ALF(Accelerated Library Framework)并行编程模型,它可用于解决数据并行问题;Sequoia^[6-7,11]是一种基于内存层次感知(memory hierarchy aware parallel programming)的编程语言,其任务可以递归调用自己;CellSs^[12]在函数级支持类似于 OpenMP 方法;Ohara 等人提出的 MPI Microtask^[13]通过将任务分解为可在 SPE 中运行的微任务并最终转换为流处理模型^[14]来达到充分挖掘 Cell 计算能力的目的;Charm++是一种支持面向对象的可移植通信库,目前已移植到 Cell 平台^[15].

以上编程模型多采用流处理模式,要求计算子任务代码和数据可完全容纳于片上 LS,SPE 使用“批量访存”(bulk data transfer)与主存交换数据,以充分挖掘 Cell 的高带宽访存和计算性能.但由于任务划分及数据分割(data partition)受 LS 容量限制,并要求程序员提供额外数据分割信息,加重了程序员的负担.

由于科学计算的可扩展性,支持不受限的可访内存空间更符合人类的思维与编程习惯,如虚拟存储技术和 64 位计算技术为软件提供了更大的寻址空间,极大方便了程序员的编程.在传统计算领域中,非规则应用大量存在,因此为用户提供支持更大寻址范围的编程模型是有必要的.

第二类编程模型通过支持 SPE 对片外大访存范围的访问来支持非规则应用.它们多采用软件管理 Cache 或类似技术来实现对主存的按需访问,

CBE SDK 提供了软件管理 Cache; Eichenberger 等人研究的 OpenMP 编译器^[16]在访存时使用软件管理 Cache; 国防科学技术大学的 CCRG (Creative Compiler Research Group)也对基于 Cell 的 OpenMP 编译技术进行了相关研究^[17], NestStep^[18]支持 BSP 并行模型. Kumar 等^[19]提出了一种 Cell 上的 MPI 实现, 参考文献[17-19]均使用软件管理 Cache 的方法访问主存中的大数据. 但目前基于软件管理 Cache 的 CBEA 片外大访存范围的访问仍存在使用不便、效率较低等问题.

我们在对 CBEA 和相关研究分析的基础上, 研究实现了一个同时支持“批量访存”与非规则访存应用的 MPI 编程支撑环境, 并提供访存剖分信息用于指导用户选择合适的访存技术及对访存进行应用级优化以改善非规则访存应用的性能.

3 支持多种访存技术的 CBEA 并行编程模型

MPI 是目前应用最为广泛的并行编程模型, 具有很好的可移植性和可扩展性, 被许多并行计算平台所支持. 在 CBEA 中, 可将同一节点内每个 SPE 视作 MPI 计算中的分立节点; 同时, 每个 SPE 通过 DMA 访问统一编址的主存, 实现 MPI 的高效通信. 参考文献[20-21]对基于共享存储的多线程 MPI 编程模型做了相关研究; 参考文献[13, 18-19]在 Cell 上实现了不同种类的消息传递并行编程模型, 但文献[13]仅支持“批量访存”应用, 参考文献[18-19]的访存依赖于现有的软件管理 Cache, 性能和可编程性都受到访存方式的制约.

选择 MPI 作为 Cell 平台的并行编程模型在可

充分利用 MPI 固有的可移植性和可扩展性外, 还具有如下的优势: (1) MPI 有大的用户群体和丰富的应用软件资源; (2) 实现少量 MPI 函数即可初步验证模型的可行性. 无论是理论证明还是实际应用均表明只使用很小的 MPI 通信原语子集即可很好地解决任务级并行问题, 如 NPB (NAS Parallel Benchmarks) 软件包中主要调用了 4 个点-to-point 通信 (point-to-point communication) 函数和 5 个集合通信 (collective communication) 函数^[22]; 分子动力学 (MD) 软件 Gromacs^[23]共有 30 多万行代码, 但只使用了约 15 个 MPI 函数; (3) MPI 模型由程序员和应用来维护和并行相关的数据分解及一致性约束, 虽然可编程性受到影响, 但却降低了模型设计与开发的复杂度、难度及系统开销.

3.1 总体结构

基于 Cell 的应用一般被编译为 PPE 和 SPE 两种映像并分别运行于 PPE 和 SPE 侧, 本文提出的支持多种访存技术的 CBEA MPI 并行编程模型主要包括如图 1 所示 4 部分: ①由控制主进程 mpirun 创建的 MPI 运行时环境, 主要根据运行时参数完成对 MPI 通信缓冲区、信号量及公共数据结构 (如 MPI_COMM_WORLD) 的初始化及应用任务进程组的创建和管理, 其中应用任务进程组中的进程数量由 mpirun 运行时的 -np 参数指定, 该进程组中的每一个进程创建两个线程, 分别属于下文的②、③两个线程组; ②PPE 侧用户逻辑: 由任务进程组创建的一组用户线程组成, 主要完成用户逻辑中的控制部分, 如主存的对齐分配与释放、MPI 通信、IO 管理及对 SPE 侧线程的同步控制等; ③SPE 侧用户逻辑: 由任务进程组创建的另一组用户线程组成, 执行 SPE 映像以完成用户逻辑中计算密集部分, 包括通

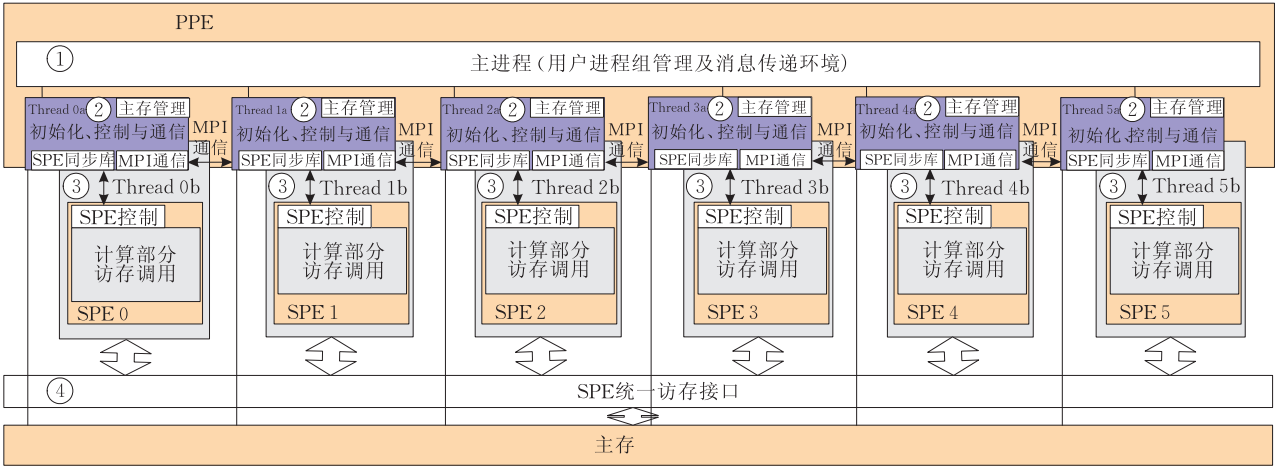


图 1 支持多种访存技术的 CBEA MPI 并行编程模型

过访存库访问主存和完成与 PPE 侧对应线程的同步;④支持多种访存技术的访存库,主要包括 SPE 访问主存的支持函数库和相应的访存剖分机制。

为了提高可编程性,本文对系统支持与用户逻辑进行了分解,为在进一步工作中实现源到源编译器创造条件.对于图 1 中①及②包含的 SPE 同步库、主存对齐分配与释放、MPI 通信库及③中 SPE 控制和④等与用户逻辑无关的部分全部封装为系统库函数的形式。

基于本文模型的应用使用与标准的 MPI 相同的执行方式,如启动当前目录应用 application 的两个任务,可使用如下命令行命令:mpirun -np 2 ./application.

3.2 基于 CBEA 的 MPI 通信

Cell 提供片上高速互连总线,用于片上核之间的低延迟、高带宽通信.但对于 MPI 模型中任务间的通信本文并未使用 SPE 间的直接通信,而是使用 PPE 侧基于共享主存的通信.原因如下:(1)SPE 间直接通信,通信数据块大小受到 LS 容量限制,从而限制编程模型的适用范围;(2)SPE 作为物理设备,它们之间的直接通信如果没有操作系统参与则不易直接寻址其它逻辑通信目标.如,若没有操作系统参与,SPE 直接寻址某线程或 MPI 某个通信域中指定 rank 的任务存在困难;如果操作系统参与通信,则要求执行操作系统的 PPE 也参与其中,其繁琐的操作会失去直接通信的优势.目前 SPE 间的直接通信,通常使用 spe_get_ls 得到某个 SPE 线程用户空间在主存中映射的有效地址 EA (Effective Address)再进行 DMA 操作,并且第一通信还会引起一次缺页异常,因此在目前情况下,SPE 核间的直接通信在可编程性与性能上并不具备太多优势.但随着硬件与软件技术的发展,在高层次并行模型中基于核间的片上直接通信仍然是一个值得研究的问题。

选用基于 PPE 侧共享主存通信模式也便于计算任务在 PPE 与 SPE 间的分解,传统应用及编程方法中将计算与通信相分离的原则为方便地将通信分解在 PPE 侧创造了条件.类似于流处理将访存与计算分离的原则,在并行计算中将通信与计算相分离可以提高计算与通信效率,在设计良好的应用中存在明显的计算与通信区域的划分.如在 Gromacs 的主要计算模块 do_force()中,其通信集中在函数入口处的 move_x()与出口处的 move_f()函数内,而数据处理和计算部分的 ns()和 force()函数则位

于两个通信函数之间.另一个 MD 应用 Moldy^[24],计算集中在 do_step()函数的 force_calc()中,而通信则紧随其后封装在 par_dsum()和 par_rsum()函数中.SPE 专门设计用于计算,对控制逻辑执行效率不高,且只有很小的存储空间,将通信与控制置于 PPE 侧,而将计算密集部分置于 SPE 侧不仅便于计算任务在 PPE 与 SPE 侧的分解也可充分发挥各个核的特长。

Linux 操作系统支持 UNIX V 共享主存技术,在图 1 中 mpirun 控制进程通过调用 shmget()及 semget()创建通信用共享主存缓冲池和相应的信号量机制.假设硬件系统为 PS3 (PLAYSTATION 3)有 6 个 SPE,信号量机制除了管理共享缓冲区之外,另有两个如图 2 所示的信号量矩阵,一个用于管理 send 类操作,一个用于管理 recv 类操作.当线程 i 要与线程 j 通信时,使用矩阵中第 i 行 j 列的信号量管理相应的 send 类、recv 类操作.使用基于共享主存的内存拷贝技术实现 MPI 点到点通信库,在点到点通信库的基础上实现集合通信库.目前主要实现了:MPI_Init(),MPI_Comm_size(),MPI_Comm_rank(),MPI_Finalize()等 MPI 控制函数和 MPI_Send(),MPI_Recv()等点到点通信函数及 MPI_Bcast()等集合通信函数.基于这个 MPI 函数子集,可以进行基本的 MPI 应用开发和移植。

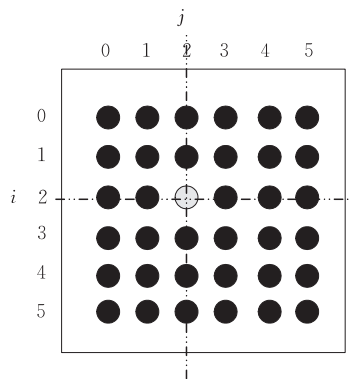


图 2 二维信号量数据结构

3.3 基于统一接口的多种访存技术支持

存储屏障(memory wall)一直是制约计算性能的重要因素^[25],它影响片上多核处理器中每个核的计算性能,从而影响整个多核系统的性能.Cell 处理器存储架构非常适合流处理模式的计算,本文通过对 CBEA 显式软件管理访存技术进行细分优化,在兼顾支持流处理类应用的基础上,扩展编程模型支持的应用范围,为 MPI 编程模型提供高效的多种访存支持。

3.3.1 CBEA 访存研究与分类

如图 3 所示,在 Cell 显式软件管理的数据移动中,要求在数据使用点 U 前的数据读入点 Ar 将计算区域 Ca 所用的数据通过 DMA 操作从片外主存读到片上数据缓冲区 B 中;数据使用后,将脏数据 (dirty data) 在写回点 Aw 处由软件写回主存. 本文将满足如上条件、覆盖计算区域 Ca 及 Ar 和 Aw 的计算部分定义为一个计算单元 Cu. 由 Ar 保证 Ca 所涉及的计算数据在 LS 中,由 Aw 保持片上与片外数据的一致性. 如下文所述,随编程模型的不同,计算区域 Ca 的粒度可以大到一个函数,也可以小到一条计算语句. 一个计算子任务可以是一个单独的 Cu,也可以由多个 Cu 组成.

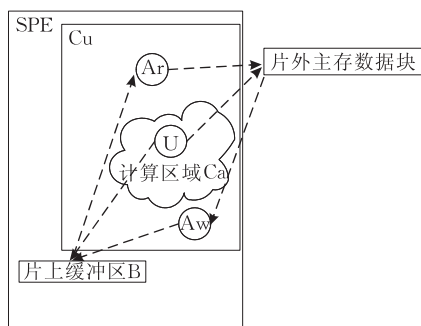


图 3 Cell 显式软件管理的数据移动

计算单元 Cu 的约束条件除要求缓冲区 B 中数据满足 Ca 需要外,还要求 Cu 的代码和缓冲区 B 占用的 LS 空间总和不超过 LS 容量限制. 在保证上述约束的前提下,通常对具有良好局部性的计算任务,计算单元 Cu 的粒度和其使用的数据块越大性能越好. 当 Cu 粒度达到函数一级,即一个子任务只含一个 Cu 时,Cu 可以看作作为一个“批量操作”(bulk operation). 一个“批量操作”由“批量访存”(Ar 和 Aw: Bulk Data Transfer)和“批量计算”(Ca: Bulk Computation)两部分组成^[6]. “批量操作”通过将访存与计算分离,将芯片制造中难以解决的访存延迟问题转化为一个访存带宽问题;另外,粗粒度 Cu 中的大数据块也增加了数据复用机会,因计算不用停等片外数据,又可充分挖掘专用处理单元的计算能力,因此性能很好. 但“批量操作”要求程序员对计算进行严格的划分和数据分割使子任务能容纳于 LS 中,并需程序员提供数据分割信息以帮助完成“批量访存”,对程序员要求较高.

随着计算区域 Ca 的减小,消费的数据、计算量、数据复用机会及对计算划分和数据分割的约束相对减少. 极端情况下,当计算区域 Ca 小到一条计算语句这样最基本的单位时,由于 Ca 数据需求减

少,不需要太大的片上缓冲区 B,所需数据按需通过 DMA 操作由主存获取,此时访存几乎不受 LS 空间限制,对程序员的计算划分和数据分割要求也降到最低. 由于访存约束较少,所以可以支持非规则访存应用. 但这时一个计算子任务通常被分割为多个 Cu,Ar 和 Aw 的点数增加,DMA 访存次数也随之增加,带宽利用率降低,计算性能变差. 相对于“批量访存”,本文将此类计算子任务不受 LS 空间约束的访存方式定义为“按需访存”,

目前“按需访存”多基于软件管理 Cache 或类似技术,通过在片上缓冲区 B 中读入比计算实际需求数据大得多的数据块以更好地利用片上总线带宽,通过在 Ar 点加入查表功能 (lookup) 增加对缓冲区 B 中数据的复用减少访存次数,但查找功能会带来大的性能开销. “按需访存”可采用以下 3 种技术:

(1) 单缓冲区,即在 LS 中开辟单个缓冲区,利用数据的局部性提高访存性能,缓冲区一般不超过 16KB (Cell 处理器一次 DMA 操作的传输上限为 16KB),该技术仅适用于顺序访存,并不是最主要的“按需访存”方法.

(2) 组相联软件管理 Cache,本质上是更复杂的缓冲区管理,它通过细粒度的缓冲区置换提高数据复用率和降低对带宽的浪费,是主要的“按需访存”技术. 针对顺序访存,虽然它也可以被配置为单缓冲区,但其效率不如专门定制的单缓冲区技术.

(3) 多路缓冲区数据预取技术,通过使用异步 DMA 操作将访存与 SPE 计算重叠,以隐藏 DMA 延迟. 如文献[12, 15-16]及 ALF 中均使用了该技术,但目前尚没有一个通用的多路缓冲区数据预取编程接口,另外数据预取需要程序员额外提供数据块的界限信息防止预取时的访存越界.

3.3.2 不同访存技术同时存在的必要性分析

由于片外与片内访存延迟差异的因素,CBEA 上不同访存技术的时间与空间性能差异很大. 以单个 SPE 上执行分块矩阵乘法 $C = A \times B$ 为例,其中 A, B 均为 512×512 的单精度浮点矩阵,分析这种差异. 分块矩阵乘法计算子任务逻辑如图 4 所示.

```
for (i=0; i<BK_ROW; i++)
  for (j=0; j<BK_COL; j++)
    for (k=0; k<BK_ROW; k++)
      C[i * BK_COL + j] += A[i * BK_COL + k] *
                          B[k * BK_COL + j];
```

图 4 分块矩阵相乘

在分块矩阵乘法中,矩阵 A, C 为顺序访存,而

矩阵 **B** 为非顺序访存,因此对矩阵 **B** 的访问是计算子任务的性能瓶颈.通过调整计算时分块大小可以模拟具有不同访存空间范围的计算任务.如,当分块矩阵大小为 32×32 ($BK_ROW=32, BK_COL=32$) 时, **A**, **B** 和 **C** 的 3 个分块矩阵分别占用 $32 \times 32 \times 4Byte=4096Byte$ 存储空间,3 个分块矩阵可以同时全部载入 LS,这时可将整个分块相乘子任务和相关访存定义为一个计算单元 Cu,并选用“批量访存”完成整个计算子任务;当分块为 256×256 时, **A**, **B** 和 **C** 的 3 个分块分别占用 $262114Byte$ 存储空间,每个分块均大于 LS 容量,显然无法将整个子任务定义为一个 Cu,只能将最内层循环(inner loop)中的浮点计算语句和相关访存设定为一个计算单元 Cu,并采用“按需访存”与主存交换数据.

图 5 列出“批量访存”和 3 种“按需访存”实现 5 种不同分块大小矩阵乘法的执行时间.其中“批量访存”使用 ALF 实现;单缓冲区及组相联 Cache 使用 CBE SDK 的软件管理 Cache 实现;四路缓冲区数据预取使用本文实现的访存接口库(每路缓冲区大小为 $128Byte$).在“批量访存”中,当分块大于等于 128×128 时,3 个分块及计算代码所需内存空间超过 LS 容量,ALF 无法完成该种规模的计算子任务,因此图 5 只实现 $32 \times 32, 64 \times 64$ 两种分块情况.

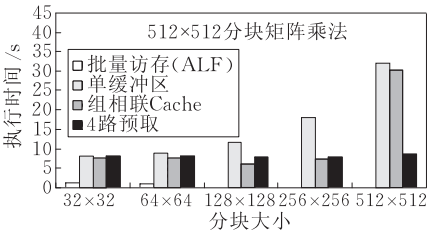


图 5 不同访存方式的时间性能比较

从图 5 可得出,同样规模的计算任务在 LS 容量允许的条件下对非顺序访存分块 B 选用“批量访存”性能最好;组相联 Cache 次之(512×512 分块下,分块过大,使组相联 Cache 无法在 LS 空间允许条件下取得好的性能);单缓冲区访存性能表现最差;四路缓冲区预取则保持一个相对稳定的值.在图 5 中最大的性能差异约 27 倍.

Cache 通过空间以数据复用来换取时间,因此对缓冲区大小有一定要求,要取得好的性能,一般会存在一个缓冲区容量的“阈值”,这个“阈值”在某些情况下可能需要精心构造.如单缓冲区技术若缓冲区大小大于数据块大小,即使对于非顺序访存也能取得好的性能.在组相联 Cache 情况下存在类似的情况,如图 6 所示,使用 CBE SDK 软件管理 Cache,在 512×512 的单精度浮点矩阵乘法中对分块 B 使

用 $64Byte$ cacheline 的 4 路组相联 Cache,通过调整组数(sets)来调整 Cache 缓冲区总大小来分析这种现象.对于不同分块大小的计算,当 Cache 的容量达到分块大小的二分之一时,计算性能获得很大提高,其后再增加组数(sets)作用不再明显.当分块过大,如分块超过 512×512 时,组相联 Cache 因受 LS 空间限制无法使计算达到一个比较优的性能. Cache 总容量“阈值”对计算性能的影响,也是一些应用采用 Cache Oblivious^[26]算法的原因.

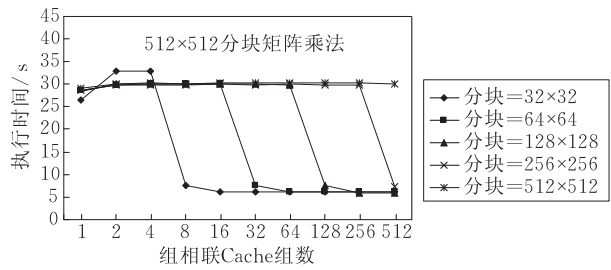


图 6 组相联缓冲区空间对计算性能的影响

图 7 显示 512×512 的单精度浮点矩阵乘法要达到理想性能水平,缓冲区大小与数据块大小的关系.如图 7 所示,“批量访存”与组相联 Cache 所需缓冲区大小与计算子任务访问数据块大小成正比,单缓冲区因受一次 DMA 最大传输 $16KB$ 的限制及较低的数据复用与带宽利用率,因此本文只将其用于顺序访存;多路缓冲区数据预取则对 LS 的需求相对稳定,一般使用 DMA 传输效率较高的 $128Byte$ 的缓冲区.因此选用何种访存技术与计算子任务需访问数据块大小和访问模式等特性有关.

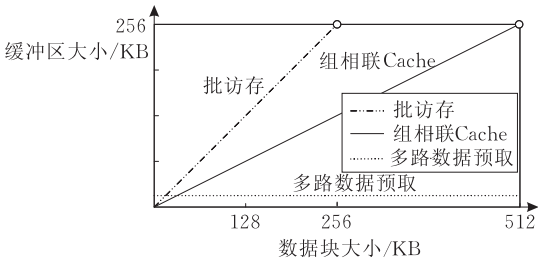


图 7 不同访存技术的空间性能比较

如上所述,不同的访存技术存在较大的性能差异,并且各自具有自身的特点和适用场合,系统有必要提供对这些技术的全面支持.另外由于各种技术实现不同,所需参数及访问机制有很大差异.加之由于 SPE 控制功能较弱,分别设计不同访存技术而非一个通用的实现,不仅可以减少 LS 空间的占用,而且可以提高执行效率.如,将只读接口与读写接口分别实现,可以将只读访存性能提高近 20%.但不同访存技术的同时存在,会加重程序员的负担.因此为

不同的访存技术提供统一的调用接口,并向用户提供相应的优化指导信息,指导用户选择相应的访存技术和配置合理的参数对提高可编程性和性能是有意义的.

3.3.3 基于统一接口的访存库与运行时剖分机制

为了同时支持多种访存技术和方便进一步访存性能优化,本文在以上分析及前期研究^[27-28]实现的“批量访存”、单缓冲区、组相联软件管理 Cache 和四路缓冲区数据预取访存库的基础上,进一步为之提供统一访存接口和增加访存运行时剖分机制以方便对前期研究工作中访存库的使用和访存性能优化.

表 1 为本文为 4 种访存技术定义的 5 个标准访存原语,在计算程序源码中(图 1 中线程组③)统一使用标准的访存原语,程序员可通过配置文件配置和选择访存技术,降低编程与优化的难度.

在 CBE SDK 的 `_malloc_align` 和 `_free_align` 里增加对内存管理的跟踪记录,在访存原语 `CACHE_R_RD`, `CACHE_W_RD` 和 `CACHE_W_WR` 中增加对主存使用的跟踪记录,以记录应用的访存行为.记录内容主要包括:(1) 访存涉及变量的起始与结束地址;(2) 访存区域上、下限及差值;(3) 访存频率;(4) 访存步长等信息.

表 1 统一访存接口定义		
访存原语	功能	
	批量访存	按需访存
CACHE_INIT	读入数据块	初始化 Cache 或缓冲区参数
CACHE_R_RD	读只读数据	读只读数据,命中直接读,否则由主存获取
CACHE_W_RD	读可写数据	读可写数据,命中则直接读,否则置换数据
CACHE_W_WR	写 LS 数据	缓冲区命中直接写,否则置换缓冲区后写
CACHE_FLUSH	写回数据块	写回数据块

计算任务运行时收集的信息可以帮助程序员对应用进行进一步优化,选择合适的访存接口或在某种接口下配置合理的参数.如上述信息中(1)和(2)可用于数据预取的上下界;信息(2)可用于程序员判断数据块大小决定是否将访存定义为“批量访存”或选择使用组相联 Cache;信息(4)可帮助判断访存是顺序的还是规则非顺序或是随机的,从而为选择单缓冲区、组相联 Cache 还是多路缓冲区数据预取提供指导.

4 验证与测试

4.1 测试环境与测试方案

在 PS3 上实现了本文提出的并行编程模型和

相关访存技术,PS3 配置了一个工作主频为 3.2GHz 的 Cell 处理器,主存 256MB,6 个 SPE,操作系统为 Linux Fedora Core 6.本文基于 CBE SDK 2.1 使用 C 语言实现了本文 3.1 节中的 MPI 运行时支持程序 `mpirun`(图 1 中①),其中 PPE 侧与 SPE 侧的通信与同步控制由 `mailbox` 机制实现(包含于图 1 中的②和③);基于共享主存技术实现基本的 MPI 通信库函数(包含于图 1 中的②);最后在 SPE 侧实现访存库及相应剖分机制(图 1 中④).

测试分两部分进行:第一部分侧重测试对多种访存技术支持及访存性能,通过在单个 SPE 上执行“批量访存”应用、“按需访存”应用及非规则应用和 LS 空间不足情况下使用多路缓冲区数据预取来验证本文对多种访存技术的支持、必要性及其与相关技术的性能比较.第二部分测试“批量访存”与“按需访存”应用的任务级并行,在使用不同访存技术情况下与相关技术的并行性能进行比较.

在整个测试中使用了 3 个测试用例:分块矩阵乘法、分块稀疏矩阵 LU 分解和 Gromacs Benchmark d.poly 及 d.villin.其中前两个测试用例代码的核心逻辑使用参考文献[12]的测试代码,并分别使用 512×512 及 1024×1024 单精度浮点数矩阵.

由于本文侧重于分析访存技术对性能的影响,因此测试用例并未采用 SIMD 等额外优化措施,而是仅依赖于编译器的自动优化(编译选项-O3),实数类型全部使用单精度浮点数.

4.2 多种访存技术支持及性能测试

4.2.1 “批量访存”测试

以分块矩阵乘法和分块稀疏矩阵 LU 分解为例进行测试,两个用例均取 32×32 分块大小并使用“批量访存”.

图 8 显示分别使用 CellSs、ALF 和本文统一接口访存库的“批量访存”实验结果.如图 8 所示,测试用例在同等条件下使用本文统一接口访存库“批量访存”的执行时间大约分别是 CellSs 和 ALF 的 39%~71%.

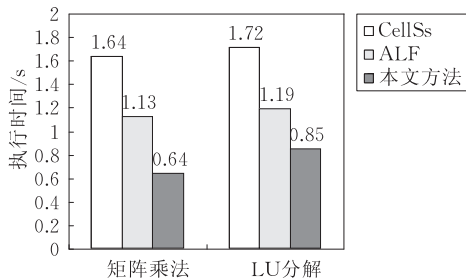


图 8 “批量访存”测试

4.2.2 “按需访存”测试

图 9 模拟“按需访存”计算任务,两个用例均取 256×256 分块大小并使用“按需访存”.使用 CBE SDK 单缓冲区、组相联 Cache 和本文统一接口访存库实现组相联 Cache、四路数据缓冲区预取的实验结果.图 9 显示测试用例基于本文四路数据缓冲区预取的执行时间仅为 CBE SDK 单缓冲区的 38%和 55%左右;在组相联 Cache 测试中,本文方法执行时间为基于 CBE SDK 实现的 59%和 71%左右.

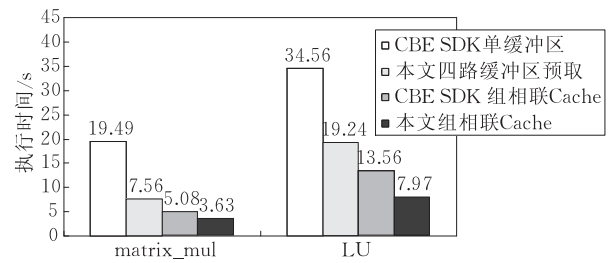


图 9 模拟大访存应用测试

4.2.3 非规则访存及剖分优化测试

非规则应用对主存数据的访问模式往往很复杂,如在分子动力学软件 Gromacs 的 kernel 参数表中有 31 个数据指针指向主存中不同的数据结构,对其访存模式的人工分析与优化是困难的.以 Gromacs Benchmark d.poly 运行 kernel010 及 Benchmark d.villin 运行 kernel112 1000 步的执行时间为例进行测试.如图 10 所示,初步使用本文的“按需访存”库执行时间约为使用同样配置的 CBE SDK Cache 执行时间的 76%和 80%.

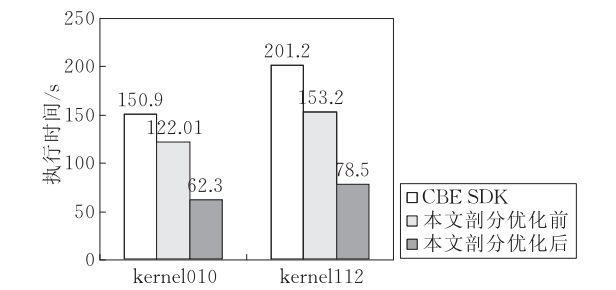


图 10 Gromacs kernel 测试

表 2 以 kernel010 为例列出使用本文访存库运行时返回的访存主要剖分信息(限于篇幅,这里仅列出 31 个指针中的 8 个).根据访存次数可以判断 pos 和 faction 两个数组为主要的访问对象,也是主要的优化对象,步长为 1 的项表示对该内存区域的访问是顺序的,可以采用单缓冲区技术.步长为 unknown 表示该数组的访问是非规则的,剖分机制无法判定其步长,这种情况多为基于索引的访问,事实上 Gromacs 以邻居列表(neighbour list)来索引

参与计算力的粒子,对于这种步长不确定的非规则访问在 LS 空间允许的情况下宜采用“批量访存”或组相联的 Cache. Block Size 项表示的访问区域大小可以用于指导分配 LS 空间.

表 2 kernel010 访存剖分信息

Ea 映射的数组	访问次数	步长	writable	Block size
jjnr	3062143525	1	0	1131508
type	3062154350	unknown	0	48000
faction	18372926100	unknown	1	148088
jindex	21650	1	0	43304
shift	10825	1	0	43304
gid	3062143525	1	0	43304
pos	18372926100	unknown	0	144000
iinr	10825	1	0	43304

根据表 2 信息,将 kernel010 可写数组 faction 改用“批量访存”,type、pos 使用组相联 Cache,其它变量使用单缓冲区访存技术.用同样方法对 kernel112 进行优化,如图 10 所示,经剖分优化可将测试用例的执行时间进一步优化到使用 CBE SDK 执行时间的 40%左右.

4.2.4 LS 空间受限时数据预取测试

如上所述,在 LS 空间允许的条件下“批量访存”与组相联 Cache 性能最好,但由于 LS 只有 256KB 空间,当 LS 空间不足时多路缓冲区数据预取库可以取得好的性能.图 11 显示当选用 2048×2048 矩阵乘法,选 512×512 分块大小并分别使用 1,2,4 个 SPE 时的并行执行时间.如本文 3.3.2 节所述,由于受 LS 空间限制,CBE SDK 和本文的组相联 Cache 此时均调整不出优的性能,且二者执行时间相差不大.但如图 11 所示,这种情况下选用具有很好空间性能的数据预取访存库,执行时间仅为组相联 Cache 的 30%左右.

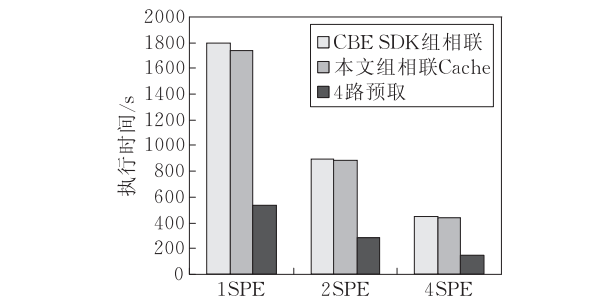


图 11 LS 空间受限时 4 路数据预取测试

4.3 MPI 任务级并行测试

4.3.1 “批量访存”应用任务级并行测试

以 1800×1800 分块矩阵乘法,分块大小取 60×60 ,分别使用 CellSs、ALF 和本文的 MPI 运行环境,测试“批量访存”访存应用并行性能.实验结果如图 12 所示,基于本文 MPI 支持环境的应用具有同

CellSs 和 ALF 相近的并行加速比,由于使用了更高效的访存库,性能比 ALF 提高 33%~40%,比 CellSs 提高 56%~70%左右。

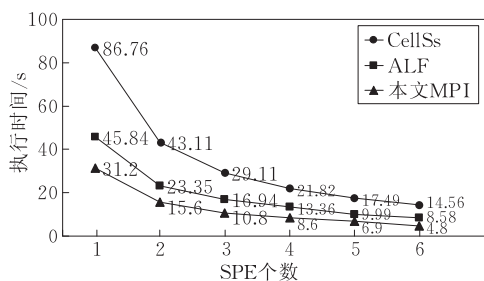


图 12 模拟小访存并行应用测试

4.3.2 “按需访存”应用任务级并行测试

以 1800×1800 分块矩阵乘法,分块大小取 150×150 模拟大访存应用,3 个分块总大小为 270000Byte 无法在 LS 中容纳下,CellSs 和 ALF 这样的编程模型暂时不能很好地支持此类大访存应用. 基于本文的 MPI 支撑环境,分别使用 CBE SDK 软件管理 Cache 及本文访存技术实现“按需访存”库实现测试用例,实验结果分别如图 13 中实线与破折线所示。

如图 13 所示,在同样的并行方式下使用本文访存库的执行时间约为 CBE SDK 访存库的 70%左右. 本文访存库提供的剖分信息显示,分块 B 大小为 90000Bytes 小于 LS 容量,且为非顺序访存,因此将其优先选用“批量访存”并置入 LS 可以显著改善计算性能. 按剖分信息优化后的执行时间如图 13 中虚线所示. 其执行时间约为 CBE SDK 的 50%左右,接近图 12 中 CellSs 的“批量访存”性能。

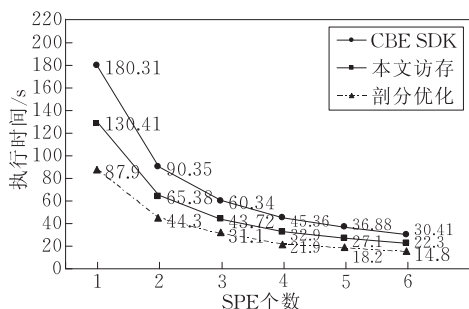


图 13 模拟大访存并行应用测试

5 结束语与进一步工作

本文在对 CBEA 架构分析的基础上,研究实现了一个 MPI 并行编程模型,在统一访存接口下同时支持“批量访存”和非规则访存应用,将通信分解在 PPE 上,拓宽模型的适用范围. 通过对访存技术进行细分优化,并提供运行时访存剖分信息帮助用户

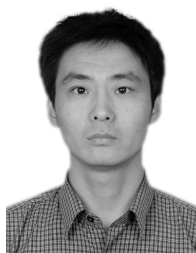
进行应用性能优化. 实验结果表明,基于本文编程模型的应用不仅可支持多种访存模式并具有很好的并行加速比,且性能显著超过同类相关技术。

本文模型目前尚不支持 Code Overlay 技术,任务级并行受限于核间与节点间二级并行的负载均衡等问题,暂不支持节点间的并行。

参 考 文 献

- [1] Hu Wei-Wu, Tang Zhi-Min. Architecture of godson-1 processor. Chinese Journal of Computers, 2003, 26(4): 385-396 (in Chinese)
(胡伟武, 唐志敏. 龙芯一号处理器结构设计. 计算机学报, 2003, 26(4): 385-396)
- [2] Hu W W, Zhao J Y, Zhong S Q, Yang X, Guidetti E, Wu C. Implementing a 1GHz four-issue out-of-order execution microprocessor in a standard cell ASIC methodology. Journal of Computer Science and Technology, 2007, 22(1): 1-14
- [3] Gschwind M. Chip multiprocessing and the Cell Broadband Engine//Proceedings of the 3rd Conference on Computing Frontiers. Ischia, Italy, 2006: 1-8
- [4] Kahle et al. The Cell processor architecture//Proceedings of the 38th Annual IEEE/ACM International Symposium on Micro-Architecture (MICRO-38). Barcelona, Spain, 2005: 3-3
- [5] Williams S, Shalf J, Oliker L, Kamil S, Husbands P, Yelick K. The potential of the Cell Processor for scientific computing//Proceedings of the 3rd Conference on Computing Frontiers. Ischia, Italy, 2006: 9-20
- [6] Knight T J, Park J Y, Ren M, Houston M, Erez M, Fatahalian K, Aiken A, Dally W J, Hanrahan P. Compilation for explicitly managed memory hierarchies//Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'07). New York, NY, USA, 2007: 226-236
- [7] Houston M, Park J Y, Ren M, Knight T J, Fatahalian K, Aiken A, Dally W J, Hanrahan P. A portable runtime interface for multi-level memory hierarchies//Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOPP 2008). Salt Lake City, Utah, USA, 2008: 143-152
- [8] Silberstein M, Schuster A, Geiger D, Patney A, Owens J D. Efficient computation of sum-products on GPUs through software-managed cache//Proceedings of the 22nd ACM International Conference on Supercomputing. Island of Kos, Greece, 2008: 309-318
- [9] Benthin C, Wald I, Scherbaum M, Friedrich H. Ray tracing on the Cell Processor//Proceedings of the IEEE Symposium on Interactive Ray Tracing 2006. Salt Lake City, Utah, 2006: 15-23
- [10] Balart J, Gonzalez M, Martorell X, Ayguade E, Sura Z, Chen T, Zhang T, O'brien K. A novel asynchronous software cache implementation for the Cell-BE Processor//Proceedings of the 2007 Workshop on Languages and Compilers for Parallel Computing. Urbana, Illinois, 2007: 125-140

- [11] Fatahalian K, Knight T J, Houston M, Erez M, Horn D R, Leem L, Park J Y et al. Sequoia: Programming the memory hierarchy//Proceedings of the ACM/IEEE Conference on Supercomputing. Tampa, FL, 2006: 83
- [12] Bellens P, Perez J M, Badia R M, Labarta J. CellSs: A programming model for the Cell BE architecture//Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing. Tampa, FL, 2006: 86
- [13] Ohara M et al. MPI microtask for programming the Cell Broadband Engine Processor. IBM Systems Journal, 2006, 45(1): 85-102
- [14] Dally W J, Hanrahan P, Erez M, Knight T J, Labonte F, Ahn J, Jayasena N, Kapasi U J, Das A, Gummaraju J, Buck I. Merrimac: Supercomputing with streams//Proceedings of the 2003 ACM/IEEE Conference on Supercomputing. Phoenix, Arizona, 2003: 35
- [15] Kunzman D, Zheng G, Bohm E, Kalé L V. Charm++, Ofload API, and the Cell Processor//Proceedings of the Workshop on Programming Models for Ubiquitous Parallelism. Seattle, WA, USA, 2006: 32-41
- [16] Eichenberger A E, O'Brien J K, O'Brien K M, Wu P, Chen T, Oden P H, Prener D A, Shepherd J, So B, Sura Z, Wang A, Zhang T, Zhao P, Gschwind M, Archambault R, Gao Y, Koo R. Using advanced compiler technology to exploit the performance of the Cell Broadband Engine Architecture. IBM Systems Journal, 2006, 45(1): 59-84
- [17] Sun Shou-Hang, Yang Can-Qun, Li Chun-Jiang, Wang Feng. The implementation of OpenMP C compiler on Cell Processor. Computer Science, 2007, 34(9A): 22-25 (in Chinese)
(孙守航, 杨灿群, 李春江, 王锋. OpenMP C 编译器在 Cell 上的实现. 计算机科学, 2007, 34(9A): 22-25)
- [18] Markus ? lind, Eriksson M, Kessler C. BlockLib: A skeleton library for Cell Broadband Engine//Proceedings of the International Workshop on Multicore Software Engineering (IWMSE-2008) at ICSE-2008. Leipzig, Germany, 2008: 7-14
- [19] Kumar A, Jayam N, Srinivasan A, Senthilkumar G, Baruah P K, Kapoor S, Krishna M, Sharma R. Brief announcement: Feasibility study of MPI implementation on the Heterogeneous multi-core Cell BE TM architecture//Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA). San Diego, California, USA, 2007: 55-56
- [20] Tang Hong, Yang Tao. Optimizing threaded MPI execution on SMP clusters//Proceedings of the 15th ACM International Conference on Supercomputing. Denver, USA, 2001: 381-392
- [21] Tang H, Shen K, Yang T. Program transformation and runtime support for threaded MPI execution on shared-memory machines. ACM Transactions on Programming Languages and Systems, 2000, 22(4): 673-700
- [22] Yuan Wei, Zhang Yun-Quan, Sun Jia-Chang et al. Performance analysis of NPB benchmark on domestic tera-scale cluster systems. Journal of Computer Research and Development, 2005, 42(6): 1079-1084 (in Chinese)
(袁伟, 张云泉, 孙家昶等. 国产万亿次机群系统 NPB 性能测试分析. 计算机研究与发展, 2005, 42(6): 1079-1084)
- [23] van der Spoel D, Lindahl E, Hess B, Groenhof G, Mark A E, Berendsen H J C. Gromacs: Fast, flexible, and free. Journal of Computational Chemistry, 2005, 26(16): 1701-1718
- [24] Refson K. Department of earth sciences, moldy: A portable molecular dynamics simulation program for serial and parallel computers. Computer Physics Communications, 2000, 126(3): 309-328
- [25] Wulf W, McKee S. Hitting the memory wall: Implications of the obvious. ACM SIGARCH Computer Architecture News, 1995, 23(1): 20-24
- [26] Frigo M, Leiserson C E, Prokop H, Ramachandran S. Cache-oblivious algorithms//Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS'99). Washington, DC, USA, 1999: 285
- [27] Feng Guo-Fu, Dong Xiao-She, Wang-Hao, Chu Ying, Zhang Xing-Jun. An efficient software-managed cache based on Cell Broadband Engine architecture//Proceedings of the International Symposium on Computer and Sensor Networks and Systems (ISCSNS2008). Zhengzhou, China, 2008: 33-40
- [28] Feng Guo-Fu, Dong Xiao-She, Ma Si-Yuan, Feng Jing-Hua. A profile-based memory access optimizing technology on CBE architecture//Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08). Dalian, China, 2008: 382-388



FENG Guo-Fu, born in 1971, Ph. D. candidate. His research interests include parallel computer architecture, parallel programming environment and operating system.

DONG Xiao-She, born in 1963, Ph. D., professor, Ph. D. supervisor. His current research interests include parallel computer architecture and grid computing.

HU Bing, born in 1984, M. S. candidate. His current research interests include parallel programming environment and fault-tolerant computing.

WANG Xu-Hao, born in 1984, M. S. candidate. His research interests include parallel computer architecture and operating system.

WANG En-Dong, born in 1966, professor. His research interests include computer system architecture and computer network.