通信优化技术在电力终端多线程系统的应用

于 杨[1,2],蔡田田[1,2],匡晓云[1,3],余 憨[4]

(1.南方电网科学研究院有限责任公司,广东 广州 510663; 2.南方电网数字电网研究院,广东 广州 510670;
3.广东省电力系统网络安全企业重点实验室, 广东 广州 510663;4.浙江大学信息与电子学院,浙江 杭州 310000)

摘 要:多线程技术在电力终端多核芯片的广泛应用,使电力终端系统的系统通信时间显著上升,电力应用的线程数也随之增加,线程间通信更加频繁。针对系统通信时间开销问题和线程间开销问题,首先引入通信流水线技术,通信流水线技术能让相同线程内的通信和运算同时工作,同时还可以降低通信传输时间;此外引入消息集聚技术将通信通道集聚,以提高单位时间数据传输量并减少通信次数。为降低线程切换次数及减少系统同步时间,引入通信队列技术;然而,线程间的依赖关系会影响通信队列技术的使用,最后提出一种面向依赖环的优化方法,该方法能够有效解决依赖环带来的限制,并提升通信队列利用率及线程间通信效率。

关 键 词:流水线技术;消息集聚技术;通信队列;线程切换;依赖环

DOI:10.19781/j.issn.1673-9140.2021.05.002 **中图分类号**:TM732 **文章编号**:1673-9140(2021)05-0010-10

Application of communication optimization technology in multithread system of power terminal

YU Yang[1,2], CAI Tiantian[1,2], KUANG Xiaoyun[1,3], YU Min[4]

(1. Electric Power Research Institute, CSG, Guangzhou 510663, China; 2. Digital Grid Research Institute, CSG, Guangzhou 510670, China; 3. Guangdong Provincial Key Laboratory of Power System Network Security, Guangzhou 510663, China; 4. College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310000, China)

Abstract: The wide application of multi-threading technology in power terminal multi-core chips has significantly increased the system communication time of the power terminal system. The number of threads in power applications has also increased, and the communication between threads has become more frequent. Aiming at the problem of system communication time overhead and inter-thread overhead, this article first introduces communication pipeline technology. The communication pipeline technology allows communication and calculations in the same thread to work at the same time, while also reducing communication transmission time; In addition, this article also introduces message aggregation technology, the communication channels are gathered to increase the amount of data transmission per unit time and reduce the number of communications. In order to reduce the number of thread switching and reduce system synchronization time, the communication queue technology is also introduced; however, the dependency between threads will affect the use of communication queue technology. Finally, an optimization method is proposed for the dependency loop, which can effective solve the limitations caused by the dependency ring, and improve the utilization

收稿日期:2019-12-21;**修回日期**:2020-10-01
基金项目:国家重点研发计划(2018YFB0904900,2018YFB0904902)
通信作者:余 憨(1988-),男,助理研究员,博士,主要从事多核处理器及其芯片设计研究;E-mail:yu_min@zju.edu.cn

of communication queues and the efficiency of communication between threads.

Key words: pipeline technology; message aggregation technology; communication queue; thread switching; dependency ring

随着电力系统朝着智能化、现代化、节能高效的目标进一步迈进，为满足日益复杂的电力应用场景，智能电表、变压器、变电站等电力终端设备也在不断升级[1-4]。电力芯片作为电力终端设备的核心，往往通过增加处理器核数来提升电力芯片的性能以满足更复杂的需求。然而，电力芯片处理器核数的增加会降低处理器的利用率，同时也会进一步增加系统通信开销和提升线程通信频率。因此如何减少系统通信开销和实现高效的线程间通信，成为优化多线程系统在电力终端应用的关键[5-6]。

为了解决处理器利用率不高及系统开销过大的问题，文献[7-9]基于对工具 SPIN(simple promela interpreter)的研究，提出一种最小化 SDF(synchronous data flow)系统所需队列的方法；文献[10]提出一种队列分派策略，能同时优化队列需求量及系统性能，并使两者达到平衡。然而，上述技术的使用基于一种理想状态，即任务的运行时间始终保持稳定不变。此外，通信队列技术不适合用于存在依赖环的系统中。文献[11]提出一种能提升系统性能的重定时技术，但该技术并未考虑依赖环问题。

为了实现高效的线程通信，文献[12]在 LESCEA 平台中采用了消息集聚技术。消息集聚技术的使用会减少系统通信通道数，但也会带来下列问题：消息集聚技术的不当使用可能会导致系统出现死锁，这是需要严格避免的；消息集聚技术在启动数据发送过程中，必须严格等待待处理的数据准备就绪，这就引入了通信延时并可能导致处理器发生阻塞，进一步增加了系统通信时间。

该文针对系统通信开销和线程开销 2 个关键因子，基于相关研究提出一套通信优化技术，同时结合静态分析和动态仿真的方法，可以显著降低系统通信时间并提升系统性能。

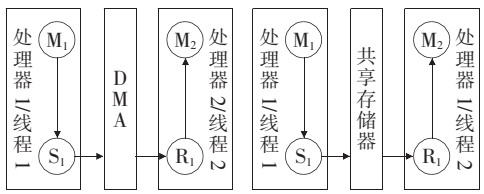
1　方法研究

1.1　通信队列

通信队列是处理器中一块特定的存储区域，发送模块与接收模块通过该区域进行数据传输，通信队列可以预处理模块数据以消除数据发送端和接收端速率不匹配造成的影响，同时能连续多次执行数据操作而处理器不发生阻塞。根据数据处理的方式，通信队列可分为核间通信队列和核内通信队列。如图 1 所示，图 1(a)和图 1(b)分别为核间通信队列系统模型和核内通信队列系统模型。

通信队列条目数会影响核间通信队列系统性能，为进一步分析通信队列条目数对核间通信系统产生的影响，图 2 给出了一个核间通信实例。由图 2 可知，条目数为 2 的核间通信队列系统在相同时间内能处理更多的模块数据，系统同步开销更小。

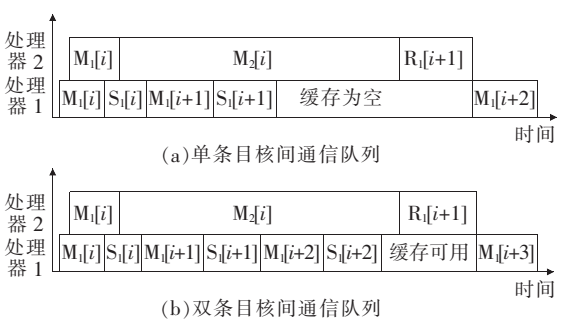
对于核内通信队列系统，如图 3 所示，系统在进行数据通信时需进行线程切换。在条目数为 1 的核内通信队列系统中，每循环一次需要切换 2 次线程；而条目数为 2 的核内通信队列系统只需切换一次线程。通信队列条目数越多，线程切换次数减少越显著，线程切换开销越小。



(a)核间通信队列　　(b)核内通信队列

图 1　通信队列的类型

Figure 1　Type of communication queue



(a)单条目核间通信队列

(b)双条目核间通信队列

图 2　核间通信队列的执行序列

Figure 2　Execution sequence of inter-core communication queue

(a)单条目核内通信队列

(b)双条目核内通信队列

图 3 核内通信队列的执行序列

Figure 3 Execution sequence of the intra-core communication buffer

1.1.1 核内通信队列

由以上分析可知,线程切换次数是影响核内通信队列系统性能的重要因素。为了得到核内通信队列分派的一个较优解,该文提出一种分派算法,下面引入该算法用到的变量定义,其中存在以下约束关系为

$$\begin{cases} S=\sum_{v\in \boldsymbol{V}} v_{\mathrm{dep}} \cdot v_{\mathrm{sz}} \\ S \leqslant S_{\mathrm{all}} \\ A=\sum_{t\in T} t_{\mathrm{sw}}=\sum_{t\in T} \dfrac{1}{\min\{v_{\mathrm{dep}} \mid v \in V_t\}} \end{cases} \tag{1}$$

式中 T 为系统划分的所有线程;$\boldsymbol{V}$ 为核内通信向量;V_t 为核内通信向量的一个子集,与线程 $t \in T$ 存在依赖关系;v_{dep} 为核内通信向量 $v\in\boldsymbol{V}$ 的队列条目数;v_{sz} 为 v 的单个队列空间所占的存储大小;t_{sw} 为 t 执行一次循环的线程切换次数;S 为核内通信队列所需的总存储空间大小;S_{all} 为系统可用存储空间大小;A 为线程总和执行一次循环的线程切换次数。

基于上述定义,该文提出如下算法,主要实现流程为每次选择一个通信向量,为其分配一个队列空间,使得系统当前线程切换总次数最小,直至系统可用存储空间耗尽。算法具体表示如下。

核内通信队列分派算法:in_buffer_alloc(S_{all} , $\boldsymbol{V}$, T);

输入:线程集合 $T=\{t_i\}$,其中 $i\in[1,n]$,核内通信向量 $\boldsymbol{V}=\{V_t\}$,其中 $t\in T$,可用存储空间大小 S_{all} ;

输出:核内队列条目数 v_{dep}。

```
begin
    for ∀ v∈V do
        v_dep=1;
    end
    A=|T|;
    S=Σ_{v∈V} v_sz ;
    do
        T'=∅;
        for ∀ t∈T do
            V_m =argmin{v_dep | v∈V_t};
            S(t)=S+ Σ_{v∈Vm} v_sz ;
        if S(t)≤ S_all then
            A(t)=A-1/min{v_dep | v∈V_t}+1/(min{v_dep | v∈V_t+1);
            T'=T'∪{t};
        end
    end
    if T'≠∅ then
        t=argmin{S(t)×A(t) | t∈T'};
        S=S(t);
        A=A(t);
        V_m =argmin{v_dep | v∈V_t};
        for ∀ v∈V_m do
            v_dep=v_dep+1;
        end
    else
        break;
    end
while;
end
```

1.1.2 核间通信队列

基于 1.1.1 节提出的算法,能够有效地分配核内通信队列,该节将介绍一种用于分配核间通信队列的方法。为了得到准确的实验结果,该文把 FPGA 仿真应用于核间通信分派过程,具体实现步骤如图 4 所示。

该文将经典的模拟退火思想应用于队列分派过程,并使用相应的算法实现。其主要实现流程为在循环进行前,首先基于静态分析的方法给系统通信队列进行分派,然后根据静态分析结果引入 FPGA 仿真,接着判断队列分派方法是否有效,即是否提高

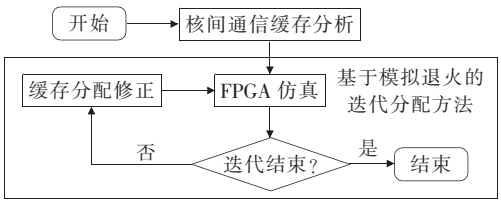


图 4　基于 FPGA 的核间通信队列分派

Figure 4　FPGA-based inter-core communication queue allocation

了系统性能。若分派结果有效则接受优化结果，否则按 $e^{(T_f-T_c)/(T_f-T_t)}$ 的概率接受结果，其中 T_f、T_c、T_t 分别为队列分派前后的系统性能和目标系统性能，上述变量值由设计者自行确定。如果系统循环次数达到上限或者系统已经无法进行优化，则算法结束并输出系统队列分派结果；否则算法一直进行下去直至满足上述要求。算法具体表示如下。

核间通信队列分派算法：buffer_alloc_SA(i_{max}，$R_{ej\text{-}max}$，T_t，P，C，M_{avl})；

输入：循环次数上限 i_{top}，连续拒绝次数阀值 $R_{ej\text{-}top}$，系统目标性能 T_t，处理器 P，通信向量 $\mathbf{V}$，总存储空间 S_{all}；

输出：核内通信队列分派结果 D_{intra}，核间通信队列分派结果 D_{inter}。

begin

$T_c=\infty$；

$i=0$；

D_{inter_curr} = buffer _ analysis (D_{inter_curr}，D_{intra_curr}，P，$\mathbf{V}$，S_{all})；

do

T_c=FPGA_Emulation(D_{inter_curr}，D_{intra_curr})；

if ($T_c>T_f$ or $e^{(T_c-T_f)/(T_t-T_f)}>$random())

then

$D_{inter}=D_{inter\_curr}$；

$D_{intra}=D_{intra\_curr}$；

$T_f=T_c$；

$R_{ej}=0$；

else

$R_{ej}=R_{ej}+1$；

end

$i=i+1$；

if $i\geqslant i_{top}$ or $T_c\geqslant T_t$ or $R_{ej}\geqslant R_{ej\text{-}top}$ then

break；

end

{D_{inter_curr}，D_{intra_curr}} = buffer _ refinement (D_{inter}，D_{intra}，parameter，P，V，S_{all})；

while；

end

1.1.3　通信队列的局限性

虽然通信队列能减少线程切换次数和降低系统同步开销，但在特定的环状拓扑结构中，通信队列技术并不适用。

核内通信队列系统划分为线程 1 和线程 2，二者形成了依赖环，如图 5 所示。由于依赖环的存在，即使为任意一个通信向量分派的队列条目数为 2，通信队列只能利用其中一个，无法体现队列条目数大于一的优势。因此，依赖环会限制通信队列技术的使用。

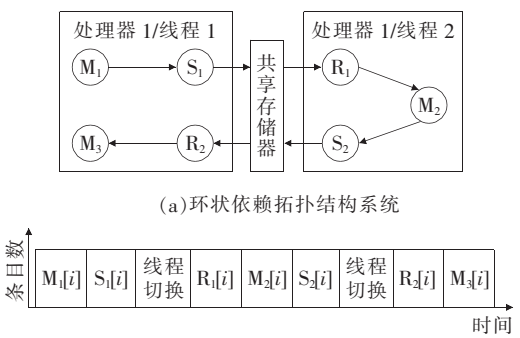


图 5　依赖环对通信队列的影响

Figure 5　The effect of ring-dependent topology on communication queue

1.2　依赖环的优化

由前文分析可知，通信队列的作用取决于系统是否存在依赖环。为解决依赖环问题，该文引入如下几个定义。

1)线程环(thread count，TC)。2 个或多个线程间存在依赖环。

2)模块环(block cycle，BC)。2 个或多个模块间存在依赖环。

3)伪线程环(dummy thread cycle,DTC):任意一个线程环中不包括模块环。

为使通信队列技术更好的提升系统性能,该文提出一种面向依赖环的线程再划分技术。首先使用 Tarjan 的搜索策略[13]定位线程环,然后构造系统有向图以重新进行线程划分,系统有向图可以分为线程有向图和任务有向图。在线程有向图中,把线程作为节点,线程间的通信向量作为有向边。在任务有向图中,把任务作为节点,任务间存在的依赖关系作为有向边。

为了使通信向量更好应用通信队列技术,需要消除 DTC。如果某个 TC 中不包含 BC,将线程中不存在依赖的部分重新进行线程划分,使得新产生的线程间不存在依赖环。具体算法实现如下。

伪线程环优化算法:dtc(T);

输入:$T=\{t_i\}1\leqslant i\leqslant n$ 为线程集合;

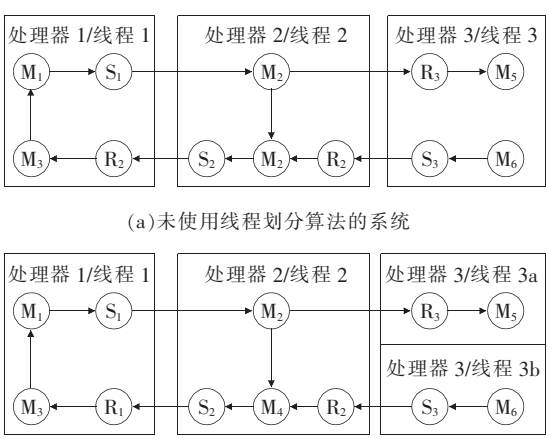
输出:新的线程集合 T。

```
begin
    T_C=t_c(T);
    for ∀ S∈T_C do
      B_C=b_c(S);
      d_ummy=false;
      for ∀v_1=s->r, v_2=s'->r'∈S do
        if ∃ t∈S-B_C s.t. r,s'∈t ∧ ∄B∈B_C s.t. v_1,v_2∈B
then
            if s'∉p=R_each(r)then
              d_ummy=true;
  T_s=(T_s∪{p}∪{t-p})\t;
              T=T∪dtc(T_s)\t;
              break;
            end
          end
        end
    end
end
```

上述算法主要实现流程为首先定位系统中存在的所有 TC;然后对每个 TC 内部的 BC 进行定位;再判断是否存在 $v_1(s->r)$和 $v_2(s'->r')$,其中 v_1 和 v_2 属于不同的 BC,同时 r 和 s'属于同一线程 t;最后通过判定条件确定 TC 是否为 DTC,若为 DTC,则将线程 t 切分以消除 DTC,直到系统不存在 DTC,算法终止。

为了进一步理解算法 3 的运行流程,以图 6 模型为例进行说明。使用 Tarjan 的搜索策略定位 TC,得到 6(a)所示的模型,其包括 3 个线程。在 TC 搜索 BC,得到模块环 $B_{C1}\{M_1,S_1,M_2,M_4,S_2,R_1,M_3\}$。但存在不属于该 BC 的通信向量 $M_2->R_3$ 和 $S_3->R_2$,且 R_3 和 S_3 属于线程 3,因此该 TC 为 DTC。

在线程 3 中,产生了 2 个新线程:线程 3a 和线程 3b。然后对新的线程集 T 进行迭代执行操作 dtc(T)。对 T 进行搜索得到 TC1{线程 1,线程 2},在 TC1 搜索 BC 得到 BC1,由于 TC1 的通信向量都属于 BC1,因此无法继续划分 TC1,迭代结束,即得到图 6(b)所示的模型。



(a)未使用线程划分算法的系统

(b)使用线程划分算法的系统

图 6 线程划分算法对系统的影响

Figure 6 The impact of thread division algorithm on the system

1.3 通信流水线技术

1)通信队列技术结合依赖环的优化方法,能够有效解决依赖环带来的限制,并提升通信队列利用率及线程间通信效率,但未对系统通信开销进行优化。

2)直接存储器访问(direct memory access,DMA)是一种内存访问技术,其可以独立地直接读

写系统内存而不需要 CPU 的介入，因此可以显著改善系统性能及提升系统吞吐量。核间通信传输具有多种架构方式，该文采用经典的基于 DMA 的分布式存储架构，其操作比较灵活，在 DMA 启动传输时，处理器对 DMA 进行配置；在 DMA 传输数据时，无需处理器干预。数据发送与运算操作往往可以同时执行，处理器可以忽略这部分通信传输开销。但是，绝大多数运算操作行为依赖于接收的数据，处理器就需要考虑这部分的通信传输开销。

为了降低通信传输时间，该文引入了通信流水线技术。其核心思想是待处理数据提前存储在缓存中，运算操作可以直接调用缓存中的待处理数据，这样便实现了运算和通信的并行执行。

1.3.1　通信流水线技术的使用策略

假设线程 T 存在一个功能模块 B，该模块取决于同线程 T 的接收模块 R。若 B 中的待处理数据已经提前被 R 接收，并存于缓存中，这样在 B 处理当前第 n 周期数据的同时，R 可以提前接收第 $n+1$ 周期的待处理数据以减少系统通信时间。

如图 7 所示，应用在不同处理器上被划分成 2 个线程，其分别映射到处理器 1 和处理器 2 上。

线程 2 存在 3 个模块，当不使用通信流水线技术时，模块 R_1、B_2 和 S_2 按顺序执行，在每一次循环中线程各模块都处理同一个周期的数据，前一个模块数据处理尚未完成时无法执行下一模块的操作；当使用通信流水线技术时，接收模块提前接收待处理数据，无需等待数据同步，这样各模块可以同时处理不同周期的数据，具体的线程执行序列如图 8 所示：如图 8(a)所示，未使用通信流水线技术时，B_2 只有在 R_1 执行完毕后才可以启动传输；如图 8(b)所示，使用通信流水线技术后，R_1、B_2 和 S_2 可以并行执行，从而可以降低通信传输时间。

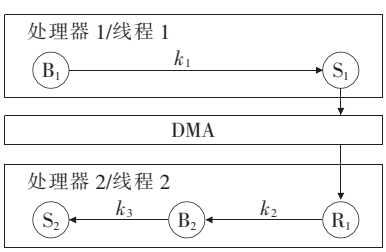


图 7　基于 DMA 的系统模型

Figure 7　DMA-based system model

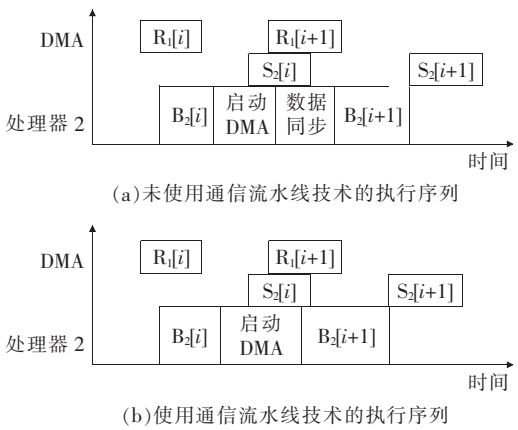


图 8　通信流水线的优势

Figure 8　Advantages of communication pipeline technology

1.3.2　通信流水线技术的局限性

由 1.3.1 节可知，通信流水线技术可以缩短关键路径的运行时间，从而降低系统的通信时间，但不是所有的通信向量都适合采用通信流水线技术。通信流水线技术高效工作的前提是数据接收和运算操作能够并行执行，因此通信流水线技术的使用前提是线程间不存在环状依赖路径。同时，通信流水线技术需要 DMA 作为中间介质来进行数据的传输，因此其不适用于核内通信向量。

1.4　消息集聚技术

通信流水线可以减少通信传输时间，但未对通信启动时间进行优化。通信启动时间是指处理器配置一次 DMA 所需的时间。显然，系统通信次数越多，通信启动时间越大。消息集聚技术将具有相同属性的通信通道集聚以降低系统通信次数，同时提高了单位时间的数据传输量，可以有效地降低系统通信启动时间。

1.4.1　消息集聚技术的使用策略

消息集聚技术将具有相同源和目的的通信通道合并，从而达到减少系统通信通道数的目的。如图 9 所示，系统在使用消息集聚技术后，将相同类型的发送模块 S_1 和 S_2 合并成一个新的模块 S_{12}，同时对接收模块做同样的处理，即把 R_1 和 R_2 合并为 R_{12}，通过这种集聚策略可以降低系统通信次数。但该技术的使用也存在前提：集聚后系统模型不存在环状依赖路径。具体实例见图 10，原系统模型不存在环状依赖路径，为一个非死锁系统，当应用了消息集聚

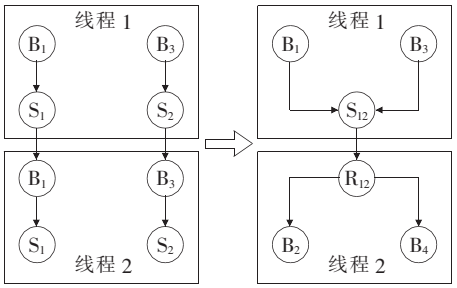


图 9 消息集聚技术的应用

Figure 9 Use of message aggregation technology

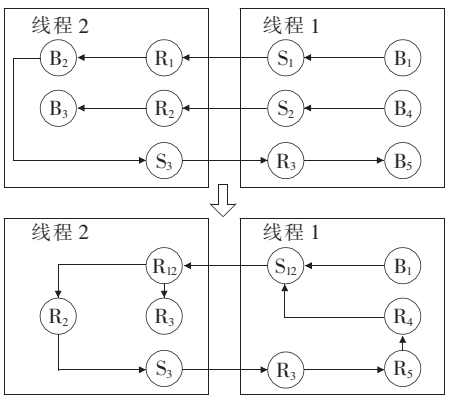


图 10 消息集聚技术引起的系统死锁

Figure 10 System deadlock caused by message aggregation technology

技术后，存在一个 $R_{12}->B_2->S_3->R_3->B_5->B_4->S_{12}->R_{12}$ 的环状依赖路径，可能会引起系统死锁[12]。

此外，消息集聚技术在数据传输时必须等待所有被集聚的数据准备完毕，这就增加了数据发送的延迟时间，由此可能导致处理器发生阻塞，影响系统性能。

从处理器处理数据的方式看，消息集聚技术分为核内消息集聚和核间消息集聚，该文将根据以上两方面分析消息集聚技术对系统性能的影响。

1.4.2 核内消息集聚

核内消息集聚导入的延时时间会一定程度的降低系统吞吐量，在系统分析延时时间对系统吞吐量的影响前，先引入如下定义。

关键模块：若某个模块被延时执行，从而导致系统吞吐量降低，则称该模块为关键模块。

关键模块是任意一个可能对系统吞吐量产生影响的通信模块。消息集聚技术延迟了通信数据的处理，为了研究消息集聚技术对系统吞吐量的影响，该文将重点考察关键模块的调用时间。如果消息集聚技术会提前执行关键模块，则认为消息集聚技术可以提升系统吞吐量。

如图 11 所示，线程 1 和线程 2 映射到同一处理器，同时假设线程 1 和线程 2 分别被调用执行了 k_1 和 k_2 次循环，模块 B_5 为关键模块。模块 B_i($i=1,2,3,4,5$)的运行时间用 t_{B_i} 表示，发送模块 S_i($i=1,2,12$)的运行时间用 t_{S_i} 表示，接收模块 R_i($i=1,2,12$)的运行时间用 t_{R_i} 表示，线程切换所需要的时间用 t_{sw} 表示。

发送模块 S_1 和 S_2 被集聚为 S_{12}，接收模块 R_1 和 R_2 被集聚为 R_{12}，则线程 2 存在 3 种不同模块调用方式。

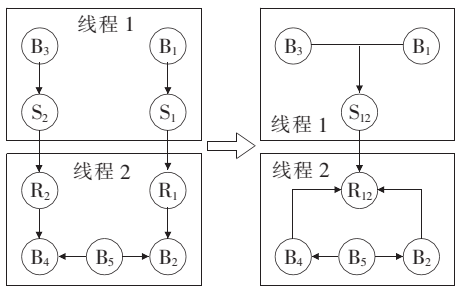


图 11 核内消息集聚实例

Figure 11 Intracore message aggregation instance

1)B_5 在 R_1 和 R_2 之前被调用，则使用消息集聚技术后，B_5 在集聚后的 R_{12} 之前被调用。

2)B_5 在 R_1 和 R_2 之后被调用，则使用消息集聚技术后，B_5 在集聚后的 R_{12} 之后被调用。

3)B_5 在 R_1 和 R_2 之间被调用。模块的不同执行顺序会破坏模块间的数据依赖关系，在使用消息集聚技术后，原本在 R_1 和 R_2 之间被调用的模块均在 R_{12} 之后被调用。因此，B_5 在集聚后的 R_{12} 之后被调用。

为进一步分析关键模块对系统吞吐量的影响，引入如下几个时间定义。未使用消息集聚技术时，B_5 执行第 j 周期的启动时间记为 $t_c[j]$；使用消息集聚技术时，B_5 执行第 j 周期的启动时间记为 $t_s[j]$。处理器在执线程 1 前，线程 2 已经被执行了 k_1 次循环，其中 $j\in[1,k_2]$。情况 1 到 3 的等式变换为

$$t_c[j]=T_1\cdot k_1+t_{sw}+T_2\cdot(j-1) \quad (2)$$

$$t_c[j]=T_1\cdot k_1+t_{sw}+t_{R1}+t_{R2}+T_2\cdot(j-1) \quad (3)$$

$$t_c[j]=T_1\cdot k_1+t_{sw}+t_{R1}+T_2\cdot(j-1) \quad (4)$$

情况 1 的等式变换为

$$t_s[j]=T_3\cdot k_1+t_{sw}+T_4\cdot(j-1) \quad (5)$$

情况 2 和 3 的等式变换为

$$t_s[j]=T_3\cdot k_1+t_{sw}+t_{R_{12}}+T_4\cdot(j-1) \quad (6)$$

其中,

$$T_1=t_{B_1}+t_{B_3}+t_{S_1}+t_{S_2}$$

$$T_2=t_{B_2}+t_{B_4}+t_{B_5}+t_{R_1}+t_{R_2}$$

$$T_3=t_{B_1}+t_{B_3}+t_{S_{12}}$$

$$T_4=t_{B_2}+t_{B_4}+t_{B_5}+t_{R_{12}}$$

通过比较上述 $t_c[j]$ 和 $t_s[j]$ 可以验证消息集聚技术对关键模块调用时间的影响。系统通信启动时间记为 t_0,R_2 的通信传输时间记为 t_1。对于情况 1 至 3,有

$$t_c[j]-t_s[j]=t_0\cdot(k_1+j-1) \quad (7)$$

$$t_c[j]-t_s[j]=t_0\cdot(k_1+j) \quad (8)$$

$$t_c[j]-t_s[j]=t_0\cdot(k_1+j-1)-t_1 \quad (9)$$

核内通信向量的通信传输时间是一个固定的常量,且低于通信启动时间 t_0。由式(7)~(9)可知,应用消息集聚技术使得任意情况下的关键模块被提前调用,即消息集聚技术可以提升系统吞吐量。

1.4.3　核间消息集聚

核间通信向量基于 DMA 进行数据传输,由于发送操作和接收操作能同时进行,因此利用上述方法无法验证消息集聚技术对系统吞吐量的影响。

通信流水线技术的使用增加了系统的延时,且该延时大于消息集聚技术引入的延时,因此当系统同时使用这两种技术时,通信流水线技术可以抵消部分消息集聚技术对系统吞吐量的不利影响。因此,该文只在应用了通信流水线技术的核间通信模块应用消息集聚技术。

2　实验测试与验证

2.1　硬件平台简介

该实验采用一个可自主配置的多核片上系统硬件平台,其通信网络可以很好满足实验需求,同时可以灵活的扩展处理器。该多核片上系统包含 16 个 CPU 核,其硬件架构在 Xilinx V6VLX760 FPGA 上实现。

2.2　实验方法

该实验基于 LESCEA[12,14-15] 平台实现用于电力安全通信 IPSec VPN 程序,同时应用通信队列分派技术,并使用该软件程序进行 TCP/IP 加密通信。

2.3　实验结果及分析

实验组分别在不同核架构上实验,并以 LESCEA 平台为基础。如表 1 所示,包含 9 组实验(LESCEA、G_0、G_1、G_2、G_3、G_4、G_5、G_6、G_7)。

通过表 2 数据可知,实验组 LESCEA 随着核数的增加,解码率也随之增加,但在核数较小时解码率增加并不明显;实验组 G_2 同时引入了通信流水线技术和消息集聚技术,其解码率比实验组 LESCEA 均有不同程度的提升,在核数增多时提升效果更为明显,这是因为通信流水线技术的使用降低了大部分通信传输时间,系统通信时间比例下降比较可观,同时通信流水线降低了系统同步时间;由实验组 G_3 和实验组 G_4 对比可知,只使用基于 DTC 的线程再划分技术对系统性能提升并不明显;由实验组 G_5、G_6 和 G_7 的对比可知,静态分析和动态仿真相结合的通信队列分派方法对系统性能的提升效果优于单独使用静态分析方法。由以上分析可知,通信流水线技术、消息集聚技术和通信队列技术的结合使用

表 1　实验组的实验编号及技术

Table 1　Experimental group number and technique

应用技术	实验组号								
	LESCEA	G_0	G_1	G_2	G_3	G_4	G_5	G_6	G_7
通信流水线	—	√	—	√	—	√	—	√	√
消息集聚	—	—	√	√	—	√	—	√	√
针对 DTC 的线程再划分	—	—	—	—	√	√	—	√	√
基于静态分析的通信队列分派	—	—	—	—	—	—	√	√	√
基于动态仿真的通信队列分派	—	—	—	—	—	—	√	—	√

可以显著提升系统性能，且提升效果在核数增加时更为明显。

如表 3～5 所示，实验组 LESCEA 的系统额外开销随着处理器数目的增加急剧提升，因此 LESCEA 平台无法适用于多核多线程系统；实验组 G_0 使用了通信流水线技术，其处理器传输状态比例相对于实验组 LESCEA 大幅下降，这是因为通信流水线技术隐藏了大部分系统通信传输开销，同时处理器空闲状态比例也有所下降，这是因为通信流水线技术减少了系统同步开销；实验组 G_1 使用了消息集聚技术，其处理器通信启动开销比例大幅下降，但处理器空闲比例小幅上升，这是因为消息集聚技术导致数据传输延时；实验组 G_4 使用了线程再划分技术，其系统额外开销显著下降，这是因为线程再划分技术显著降低了环状通信拓扑结构对通信流水线技术的限制，大幅减少了系统通信传输开销。

表 2 不同架构的系统性能

Table 2 System throughput for different architectures

组号	不同核数处理器解码率						
	4	6	8	10	12	14	16
LESCEA	3.9	5.1	5.8	6.0	6.1	6.2	6.4
G_0	4.0	5.7	6.2	7.0	7.8	8.6	10.1
G_1	3.9	5.1	5.8	6.1	6.3	6.5	7.1
G_2	4.0	6.0	6.4	7.8	8.7	9.4	11.0
G_3	3.8	5.2	5.9	6.1	6.3	6.5	6.9
G_4	4.1	6.1	7.0	8.4	9.8	11.7	12.6
G_5	3.9	5.1	6.2	6.4	7.2	7.8	8.1
G_6	4.1	6.0	8.0	9.8	11.8	13.2	14.1
G_7	4.1	6.0	8.0	10.0	12.0	14.0	16.0

表 3 不同架构的系统空闲状态比例

Table 3 The proportion of system idle states of different architectures %

组号	不同核数处理器空闲状态比例						
	4	6	8	10	12	14	16
LESCEA	20	22	31	35	41	43	44
G_0	14	18	25	24	33	32	31
G_1	21	24	35	40	44	47	48
G_2	15	19	27	28	35	33	32
G_3	20	23	28	31	36	38	39
G_4	15	16	21	21	24	24	23

表 4 不同架构的系统传输状态比例

Table 4 The proportion of system transmission status of different architectures %

组号	不同核数处理器传输状态比例						
	4	6	8	10	12	14	16
LESCEA	10.0	11.0	12.0	13.0	15.0	17.0	18.0
G_0	3.1	3.8	3.9	4.0	5.9	6.6	6.2
G_1	7.2	8.8	8.6	10.2	10.4	12.6	12.1
G_2	2.8	3.3	2.6	3.4	5.3	6.2	5.2
G_4	1.0	1.2	1.0	1.1	0.9	0.8	0.7

表 5 不同架构的系统启动状态比例

Table 5 Proportion of system startup status of different architectures %

组号	不同核数处理器启动状态比例						
	4	6	8	10	12	14	16
LESCEA	8.0	7.0	5.0	6.0	6.0	4.0	4.0
G_0	2.1	1.9	2.0	2.1	2.1	2.1	2.2
G_1	1.8	1.6	1.7	1.8	1.3	1.7	1.8
G_2	8.1	7.1	5.8	6.6	6.9	6.2	6.7
G_4	2.7	2.7	2.9	3.2	3.1	3.2	3.4

3 结语

该文针对电力终端多线程系统的性能优化问题，提出了一种基于静态分析和动态仿真的通信队列分派技术，并结合通信流水线技术和消息集聚技术，通信流水线技术可以隐藏系统通信传输开销，进而有效降低通信传输时间；消息集聚技术可以大幅降低系统通信通道数，进而大幅减少系统启动开销，并在此基础上引入了面向依赖环的优化技术，进一步提升了系统性能。实验数据表明，此优化方案可以有效地提升多线程系统性能。

参考文献：

[1] 何金栋，王宇，赵志超，等. 智能变电站嵌入式终端的网络攻击类型研究及验证[J]. 中国电力，2020，53(1)：81-91.
HE Jindong，WANG Yu，ZHAO Zhichao，et al. Type and verification of network attacks on embedded terminals of intelligent substation[J]. Electric Power，2020，53(1)：81-91.

[2] 黄吉涛，周媛奉，梁飞，等. 以 IR46 电表测试为例的硬件检测综述[J]. 电力系统保护与控制，2020，48(3)：99-105.
HUANG Jitao，ZHOU Yuanfeng，LIANG Fei，et al. A survey of hardware detection using IR46 meter test as an example[J]. Power System Protection and Control，2020，48(3)：99-105.

[3] 刘远龙，潘筠，王玮，等. 用于泛在电力物联网的配电变压器智能感知终端技术研究[J]. 电力系统保护与控制，2020，48(16)：140-146.
LIU Yuanlong，PAN Jun，WANG Wei，et al. Research on intelligent sensing terminal technology of a distribution transformer for ubiquitous power internet of things [J]. Power System Protection and Control，2020，48 (16)：140-146.

[4] 郑思达，梁琪琳，彭鑫霞，等. 基于模糊聚类的异常用电行为识别研究[J]. 电测与仪表，2020，57(19)：40-44.
ZHENG Sida，LIANG Qilin，PENG Xinxia，et al. Research on abnormal power consumption behavior identification based on fuzzy clustering[J]. Electrical Measurement & Instrumentation，2020，57(19)：40-44.

[5] 王天伟. 一种 Linux 系统下的线程通信方法[J]. 计算机应用与软件，2017，34(10)：330-333.
WANG Tianwei. A method of communication between threads under linux system[J]. Computer Applications and Software，2017，34(10)：330-333.

[6] 张晓濛. 面向 MPSoC 通信优化的任务映射和调度研究[D]. 杭州：浙江大学，2018.

[7] Holzmann G J. Model checking software[M]. Berlin：Springer Berlin Heidelberg，2012：155-171.

[8] Liu W C，Gu Z H，Xu J，et al. An efficient technique for analysis of minimal buffer requirements of synchronous dataflow graphs with model checking[C]//Proceedings of the 7th IEEE/ACM International Conference on Hardware/software Codesign and System Synthesis，New York，USA，2009：61-70.

[9] Hartel P H，Ruys T C，Geilen M C W. Scheduling optimisations for SPIN to minimise buffer requirements in synchronous data flow[C]//Proceedings of the 2008 International Conference on Formal Methods in Computer-Aided Design，Portland，USA：IEEE，2008.

[10] Stuijk S，Geilen M，Basten T. Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs[C]//2006 43rd ACM/IEEE Design Automation Conference，San Francisco，USA：IEEE，2006.

[11] Cong J，Han G，Jiang W. Synthesis of an application-specific soft multiprocessor system[C]//Proceedings of the 2007 ACM/SIGDA 15th International Symposium on Field Programmable Gate Arrays，Los Angeles，USA，2007.

[12] Brisolara L，Han S，Guerin X，et al. Reducing fine-grain communication overhead in multithread code generation for heterogeneous MPSoC[C]//Proceedings of the 10th International Workshop on Software & Compilers for Embedded Systems，Nice，France，2007.

[13] Tarjan R. Depth-first search and linear graph algorithms[J]. SIAM Journal on Computing，1972，1(2)：146-160.

[14] Han S I，Chae S I，Jcrraya A A. Functional modeling techniques for efficient SW code generation of video codec application[C]//Asia and South Pacific Conference on Design Automation，Yokohama，Japan：IEEE，2006.

[15] Yan R，Huang k，Yu M，et al. Communication pipelining for code generation from simulink models[C]//2013 12th IEEE International Conference on Trust，Security and Privacy in Computing and Communications (TrustCom)，Melbourne，Australia：IEEE，2013.