软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(9):2945−2962 [doi: 10.13328/j.cnki.jos.005978]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

神威太湖之光上分子动力学模拟的性能优化*

田 卓, 陈一峯

(北京大学 信息科学与技术学院,北京 100871)

通信作者: 田卓, E-mail: t.z@pku.edu.cn

摘 要: “神威•太湖之光”国产超级计算机的特点是适用于高通量计算系统,此类系统往往存储器访问延迟,网络延迟较长.在实际应用中,有一大类问题是时间演化的模拟问题,往往需要高频状态迭代,每次迭代需要通信.此类应用问题的典型代表是分子动力学模拟,分子的性质依赖于时间演化,导致状态相关的时间尺度上难以并行化.实际应用中,全原子模型需要模拟超过μs 时间尺度,每一步的物理时间为 1fs~2.5fs,这意味着所需时间步个数超过 10^{12} 个.众核处理器中,不同核心访存时需较长的“排队”等待,造成访存延迟.另外,网卡通信延迟以及较长的数据通路会带来网络延迟,由此导致在长延迟的众核处理器上进行一次有效的模拟几乎是不可能的.解决此类问题的主要挑战是提高迭代频率,即每秒执行尽可能多的迭代步.针对神威高性能芯片处理器的体系结构特点,以分子动力学模拟为例,研究了一系列优化策略以提高迭代频率:(1) 单核通信与片上核间同步相结合,降低通信成本;(2) 共享内存等待与从核同步相结合,优化异构体系结构中的核间同步;(3) 改变计算模式,减少核间数据关联和依赖关系;(4) 数据传输与计算重叠,掩盖访存延迟;(5) 规则化问题,以提高访存凝聚性.

关键词: 神威太湖之光;分子动力学;迭代;异构;同步

中图法分类号: TP302

中文引用格式: 田卓,陈一峯.神威太湖之光上分子动力学模拟的性能优化.软件学报,2021,32(9):2945−2962. http://www.jos.org.cn/1000-9825/5978.htm

英文引用格式: Tian Z, Chen YF. Performance optimization of molecular dynamics simulation on Sunway TaihuLight system. Ruan Jian Xue Bao/Journal of Software, 2021,32(9):2945−2962 (in Chinese). http://www.jos.org.cn/1000-9825/5978.htm

Performance Optimization of Molecular Dynamics Simulation on Sunway TaihuLight System

TIAN Zhuo, Chen Yi-Feng

(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract: Sunway TaihuLight supercomputer is suitable for high-throughput computing systems, which tend to have memory access latency and network latency. There is a large class of problems namely time-to-solution, which requires high frequency iterations. The typical application of time-to-solution problems is molecular dynamics simulation. Computations in molecular dynamics simulation depend on the time. Therefore, the iterative computations are difficult to be parallelized. Time scale usually exceeds microsecond, which means that the number of steps is more than 10^{12}. It is impossible to finish effective simulation in a limited time on long latency system. Therefore, the main performance bottleneck on long latency Sunway system is how to increase the iterative frequency. This study proposes a series of optimization strategies to improve the iterative frequency: (1) Reducing communication overhead and network competition costs through single-core communication combined with on-chip synchronization; (2) Optimizating the speed of synchronization between cores through waiting the shared memory variable and synchronizing the computing processing elements; (3) Reducing the data dependencies by changing the computation patterns; (4) Covering up the memory access latency by overlapping computation and communication; (5) Regulating the data structure to improve accessibility.

* 基金项目: 国家重点研发计划(2017YFB0202001); 国家自然科学基金(61432018, 61672208)

Foundation item: National Key Research and Development Program of China (2017YFB0202001); National Natural Science Foundation of China (61432018, 61672208)

收稿时间: 2018-11-08; 修改时间: 2019-10-25; 采用时间: 2019-11-06

Key words: Sunway TaihuLight; molecular dynamics; iteration; heterogeneous; synchronization

分子动力学模拟是考察系统随时间演化的行为[1,2].在一定时间内,靠计算机模拟分子和原子体系运动,广泛应用于物理、化学、生物等领域.用分子动力学模拟微观生物现象如蛋白质折叠[3,4]所需时间尺度为毫秒,为捕获原子震荡现象,每个迭代步设为 1fs~2.5fs,模拟总时长若为毫秒,需要 10^{12} 个迭代步.相邻两个迭代步间具有相关性,需要一次通信,而频繁通信成为此类问题计算性能的主要瓶颈.

时间演化类问题从时间轴来模拟体系演化.时间被离散化为一定步长为间隔的时间点,下一个时间点的状态依赖上一个时间点的状态,两个状态之间需要一次通信以交换数据.典型的应用是分子动力学模拟[5,6],分子沿着这些时间点运动,每个时间点对应分子的一个状态.分子在 t 时刻的状态用向量 $X(t)$表示,记录 t 时刻所有粒子的位置,速度等状态信息.t+1 时刻的状态 $X(t+1)$由 t 时刻状态 $X(t)$演算而来,即 $X(t+1)=f(X(t))$.因而,不同时刻粒子状态间的关系是相互依赖的,需要通信.而神威上的通信延迟较长[7],包括网卡上的通信延迟以及较长的数据通路所带来的通信延迟都太长了.此类计算模式的时间演化类程序在高延迟的神威集群上如何优化,规避通信延迟,提高性能,是本文的宗旨.

神威太湖之光超级计算机以理论峰值 125.4PFLOPS/s 位居世界第一,其 SW26010 处理器是针对神威定制的多核处理器,每个处理器由 4 个核组组成,每个核组包含 1 个主核和 64 个从核.神威超级计算机上共安装了 40 960 个 SW26010 处理器,共 10 649 600 个核[8].在神威上,大部分应用是以一个 CPU 作为单位编程[9],只用 4 个核组中的一个核组进行计算和通信,如图 1 所示.CPU 上的 4 个核组共享一个网络端口,若问题规模不变,这就迫使我们充分利用所有核组的计算能力,以提高性能,如图 2,4 个主核同时计算和通信.由图可见,两种模式的区别是:(1) 进程数不同;(2) 消息个数不同.设问题规模为 N,进程个数为 P,模式 1,进程上任务量为 N/P,消息个数 m.模式 2,每进程上任务量 $N/4P$,消息个数 $4\times m$.消息个数增加了 4 倍,因而带来了通信延迟.如何在以核组为单位的编程模式中减少消息个数,优化通信延迟.

如图 3 所示,每个 CPU 只选一个进程用于与外界通信[10],片上所有核间的数据同步是通过共享片外存储来实现,减少了用于通信的进程数,即实现了消息个数的减少,优化了通信延迟.这是神威上较快的共享数据的方式[7],但由于多核访存需要经过访存队列,这种方式仍然较慢,带来访存延迟.

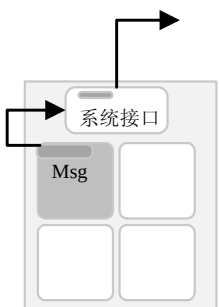

Fig.1 Node master process mode

图 1 节点主进程方式

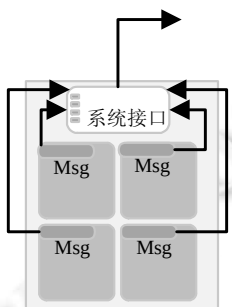

Fig.2 Core groups programming

图 2 核组为单位编程

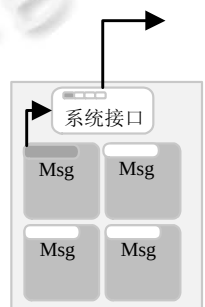

Fig.3 Reducing the number of processes

图 3 减少进程数方式

针对诸如分子动力学模拟等延迟敏感的时间演化类应用,本文的主要贡献是,给出了完整的主从核异构的国产神威处理器上的一系列优化技术,为类似的通信受限类程序提供了参考.

(1) 异构体系结构下核间同步的优化,采用共享内存等待与从核同步相结合的方式,优化访存延迟,进而优化通信延迟;

(2) 改变计算模式,采取有利于减少核间数据关联和依赖关系的计算模式,减少相邻迭代步间的通信次数和数据等待,优化通信延迟;

(3) 高效访存及规则化以提升性能.数据传输与计算重叠,以优化从核访主存效率.规则化数据结构以提

升性能,使得向量长度对齐、连续、大小规整.

本文第 1 节对神威太湖之光的结构及实例的计算模式进行描述.第 2 节给出神威上的几种优化技术,并进行详细阐述.第 3 节对实现结果进行描述.第 4 节对已有研究工作进行分析和总结.第 5 节总结全文.

1 神威结构及计算实例

1.1 神威结构

神威太湖之光超级计算机是基于主从核异构的体系结构[7],如图 4 所示.处理器芯片 SW26010 由 4 个核组(core groups,简称 CGs)构成,每个核组包含一个主核 MPE(运算控制核心,management processing element,简称 MPE)和一个 CPE 集群(computing processing elements,简称 CPEs).CPE 集群由 64 个从核按照 8×8 的网状格式组成,64 个从核阵列之间可采用低延迟的寄存器方式通信.4 个核组间通过片上网络(network on chip,简称 Noc)连接,每个核组有自己的内存空间,通过内存控制器(memory controller,简称 MC)连接到 MPE 和 CPEs 上.协议处理单元(protocol processing unit,简称 PPU)负责处理来自 CPEs 和 MPE 不同类型的请求.所有核组通过系统接口(system interface,简称 SI)与外界相连.

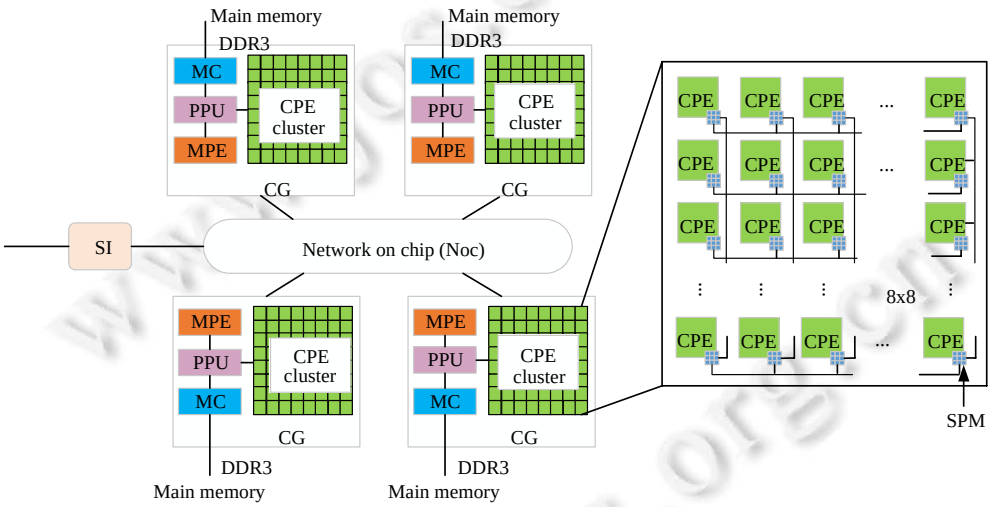

Fig.4 Architecture of the SW26010 processor

图 4 神威体系结构

每个从核有 64KB 的 SPM(scratch pad memory).为了提高内存带宽的使用率,从核可通过 DMA 方式批量访问主存(main memory),DMA 通道能够高效地在主存和 SPM 间传递数据.国产众核服务器的高性能集群上,每个节点大小为 4,节点内相对进程号为 0 的进程是节点主进程.节点内 4 个进程共享同一个网络端口,由系统接口 SI 连接.在优化算法中,我们选用 0 号主进程用于通信,其余 3 个进程与 0 号进程间通过片上共享数据交互信息.

申威异构的体系架构下,若问题本身的计算具有高频迭代性质,同时不同处理单元上的所分配的计算任务之间具有相互依赖性,由此产生的相互依赖的计算需要在多个异构单元下进行高频通信以及同步访存.本文提出的优化策略能够为此类延迟敏感类应用的性能优化提供一定的参考.本文的算例是基于时间演化类程序的典型代表分子动力学模拟为例,此外,本文提出的优化策略同样普适于类似的高频迭代类应用如 stencil 计算,它是很多科学计算类应用最为重要和耗时最多的计算核心.此类应用的特征是问题本身的求解需要大量高频迭代,迭代算法耗时较多.此类应用在异构体系结构下的性能瓶颈是系统较长的延迟,性能优化的重点在于提高迭代频率.

1.2 计算实例

分子动力学模拟是指用计算机模拟大量粒子系统中每个粒子的运动过程.系统中每个粒子在其他粒子所

形成的势场下运动,运动方程遵循牛顿定律[11].牛顿方程如下:

$$M_i\frac{\mathrm{d}^2r}{\mathrm{d}t^2}=F_i,i=1,2,...,N \tag{1}$$

$$F_i=-\nabla_i\Phi(r_1,r_2,\ldots,r_N) \tag{2}$$

其中:N 表示系统中的原子数;M_i,r_i,F_i 分别表示原子 i 的质量、位移以及受力;Φ表示势函数,势函数需能够描述系统在时间演化的各个阶段,即原子各种状态下的相互作用.复杂系统的描述多采用多体势,本文研究的硅材料,原子间的作用势取决于原子间的距离以及原子间的成键方向[12,13].硅原子间的多体作用模型采用典型的共价键势 Tersoff 势描述.Tersoff 势函数的数学表示为

$$E=\sum_i E_i=\frac{1}{2}\sum_{i\neq j}V_{ij} \tag{3}$$

$$V_{ij}=f_c(r_{ij})[f_R(r_{ij})+b_{ij}f_A(r_{ij})] \tag{4}$$

f_R 为排斥势,f_A 为吸引势,f_C 为截断函数,b_{ij} 为键级项:

$$f_R(r_{ij})=-A_{ij}\exp(-\lambda_j r_{ij}) \tag{5}$$

$$f_A(r_{ij})=-B_{ij}\exp(-\mu_j r_{ij}) \tag{6}$$

$$f_c(r_{ij})=\begin{cases}1, & r_{ij}<R_{ij}\\ \dfrac{1}{2}+\dfrac{1}{2}\cos\left[\dfrac{\pi(r_{ij}-R_{ij})}{S_{ij}-R_{ij}}\right], & R_{ij}<r_{ij}<S_{ij}\\ 0, & r_{ij}>S_{ij}\end{cases} \tag{7}$$

$$b_{ij}=(1+\beta^n\xi_{ij}^n)^{-1/2n} \tag{8}$$

$$\xi_{ij}=\sum_{k\neq i,j}f_c(r_{ik})g(\theta_{ijk})\exp(\lambda_3^3(r_{ij}-r_{ik})^3) \tag{9}$$

$$g(\theta)=1+\frac{c^2}{d^2}-\frac{c^2}{d^2+(h+\cos\theta)^2} \tag{10}$$

本文以分子动力学模拟中多体作用模型实现的单晶硅体系原子模拟为计算实例,说明这种计算模式的应用在神威集群上运行时所遇到的性能瓶颈,并给出几种优化技术.硅原子之间的多体作用模型采用 Tersoff 势函数[14],该模型的特点是原子之间的相互作用依赖于原子所处的局域环境.每个迭代步将原子 t 时刻的状态推演到 t+1 时刻,更新原子状态时,需获得该原子的所有邻居原子的状态信息,即每个迭代步需要一次同步.原子之间的依赖关系具有局部相关性,因而导致一定的异步性.此类时间演化类问题,硅原子的状态是由时间来推进演化的,因此,不同迭代步之间是相互依赖的,需要一次通信.

考虑常温下,硅晶体保持规则的金刚石晶胞结构,每个原子周围有 4 个邻近原子.如图 5 所示,硅原子 a 有 4 个邻居原子 b,c,d,e,每个原子由不同线程计算.t 时刻,每个线程更新了 5 个粒子的位置信息,进入下一个迭代步.a 原子的受力受到它的所有邻居原子的影响,需要得到 4 个邻居粒子的位置信息,来计算不同邻居对 a 的分子间作用力.因此,需要一次通信来获得所有邻居在 t 时刻的位置信息.

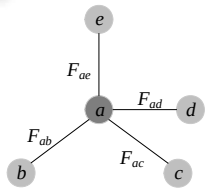

Fig.5 Schematic of interatomic interaction

图 5 硅原子间相互作用示意图

有效的原子模拟现象,需要 10^{12} 个迭代步,相邻迭代步间需要一次通信,那么在实验室环境下,观察到有效的

原子模拟现象是不现实的.为了提高硅原子模拟的迭代频率,有效的做法是减少迭代步间通信的延迟.由神威体系高性能集群的特点可知,其上的通信延迟较长.那么针对诸如分子动力学模拟的时间演化类程序,如何在神威上减少通信延迟、提高迭代频率,本文给出了一系列优化技术.

2 神威上的优化

CPU 的从核线程负责计算,两个从核线程位于 CPU0 和 CPU1 上,如图 6.CPU0 上的从核线程产生的数据,是 CPU1 上的从核线程下一个迭代步的输入,输出又作为 CPU0 上从核线程下个迭代步的输入.这样产生了数据依赖关系.

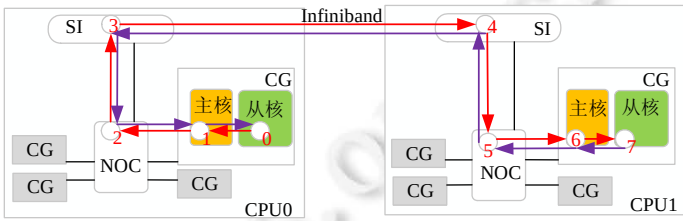

Fig.6 Data communication loop between different CPUs on Sunway

图 6 神威上不同 CPU 间的数据通信环路

若不做任何计算,数据在两个 CPU 之间传递一个来回也需要花费时间.CPU 计算能力即使再快,这个通信环路花费的时间长的话,也没有任何意义.不考虑任何计算时间,系统的速度极限即每秒能做多少次迭代,与这个环路每秒能做多少次通信是一样的.经测试,神威上的速度极限是每秒能做 3 000 次通信,但这个速度实际上太慢了.环路上的通信延迟,包括发生在网卡上和每台服务器上的延迟都太长了.对通信受限的时间演化类程序而言,通信延迟将极大地制约神威机器性能的发挥.因此,我们希望在神威上优化通信延迟.

SW26010 众核处理器采用片上融合的异构体系结构[15],包含 4 个核组,如图 7 所示.4 个核组之间通过片上互联网络(network on chip,简称 Noc)连接,片上互联网络与系统接口(system interface,简称 SI)相连,系统接口连接到以太网.由申威处理器 4 个核组与以太网连接的方式可以看出:4 个核组在硬件上共享同一个网络接口,系统的通信能力固定.如果 4 个核组的 260 个核并发通信,多核并发通信将造成进程间竞争网络资源;同时,较多的消息个数将导致网络阻塞,通信性能降低.因此,本文考虑选用一个主核负责与外界通信,芯片内部采用核间同步策略来降低竞争成本,实现网络通信消息个数的减少,优化通信延迟.

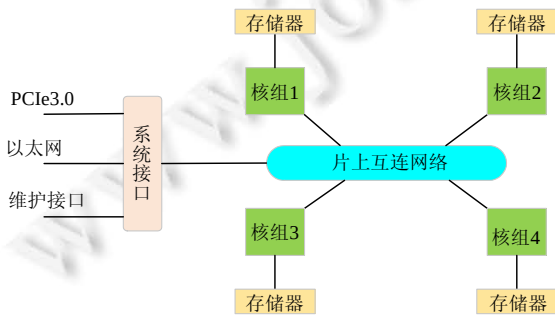

Fig.7 Connection of network on SW26010

图 7 申威芯片网络连接示意图

分子动力学模拟中,粒子的状态迭代函数需要获取其远程邻居粒子的状态信息,如更新后的位置,以进行新的受力计算,进而更新加速度、速度和位移.迭代函数的性质决定了不同处理器之间的计算是相互依赖的,需要大量频繁的数据同步.本文采用的单进程通信能够降低处理器内部的通信开销,以及维护大量通信链接对资源

的占用而带来性能优化.这部分的优势是因为分子动力学模拟程序是时间演化类以及高频迭代类程序的典型代表,不同处理器核间频繁大量的数据交换导致此类程序对延迟较敏感,网络延迟对算法的性能影响较大.因此,神威系统上单进程通信的优化算法主要针对对异构架构下类似的延迟敏感及通信受限类应用的优化提供了参考.

2.1 减少消息个数,优化通信延迟

2.1.1 片上同步减少消息个数,但带来访存延迟

SW26010 芯片由 4 个核组构成,以核组为单位的编程模式中,每个核组的 64 个从核在计算结束后,需要通过其所在主核与其他进程通信,交换粒子更新后的状态信息,以进入下一个迭代步.由 SW26010 芯片的结构特点可知:4 个核组通过系统接口 SI 共享同一个网络端口,系统的通信能力固定.那么,如何优化通信延迟成为首要解决的问题.

本文通过减少用于通信的进程数以减少消息个数,优化通信延迟.每个节点选取 0 号核组即 0 号进程作为通信进程,CPU 上其余 3 个进程与 0 号进程间通过片上同步进行通信.与以核组为单位的编程模式相比,通信进程数减少到原来的四分之一.

具体做法如图 8 所示,每个节点的 0 号主核首先进入通信等待状态,等待接收其他节点在上一时刻所更新的粒子信息.接收消息后,0 号主核需要将消息同步给本节点的所有从核以开始计算.由于神威处理器芯片是主从核异构模式,该同步过程包括两部分:主核同步和从核同步.0 号主核将消息同步给其他 3 个主核,此时,4 个主核完成同步.每个主核需要将消息同步给该核组的 64 个从核,实现主从核的同步.至此,所有负责计算的 256 个从核得到了其他节点的粒子信息,开始计算.由于主从核的异步执行,此时 4 个核组的 0 号主核等待本核组的所有从核计算结束.核组内的从核计算结束后,通过与该核组的主核执行一次主从核同步,使得该核组的主核获得本核组的所有从核已计算完成的消息.4 个核组执行一次主核同步,同步所有核组计算完成的状态.0 号主核将计算后的粒子状态通信给其他节点.至此,一个完整的迭代步结束.

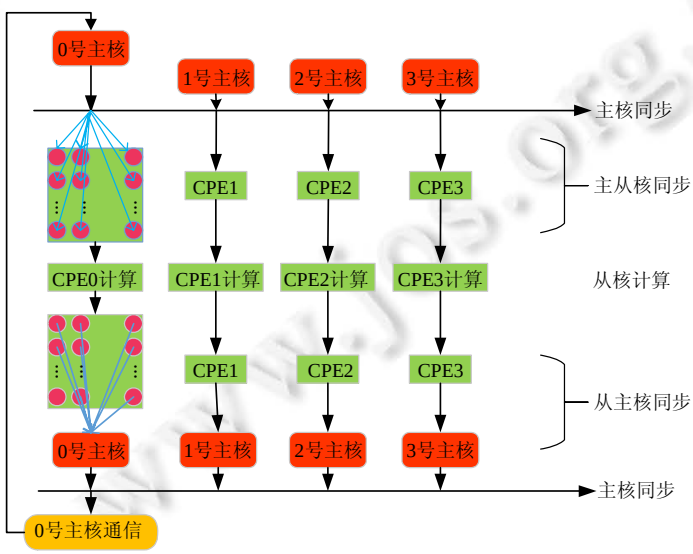

Fig.8 Process of synchronization on the chip

图 8 片上同步的实现流程

异构体系结构下代码分为两部分:主核代码和从核代码,如图 9、图 10 所示.

图 8 表示一个完整的迭代步,包括计算部分和通信部分.一个迭代步内,需要两次主核同步、两次主核与从核同步.由于神威处理器没有 cache,同步均需通过共享内存数据的方式来实现[16],即同步需访存.掣肘于神威处理器访存长延迟的特点,节点同步虽减少了网络通信的消息个数,但同时也带来了访存延迟,对优化消息个数带

来了挑战.那么,如何优化访存延迟?

申威处理器上内存总大小为 32GB,每个核组通过内存控制器与 8GB 的本核组内存相连,如图 7 所示,不同核组之间如需同步,需要访问片外存储器.那么,多核组之间共享内存空间在内存分配上将带来一定的性能损失[16].该部分的性能损失在本文的优化算法中,采用节点共享变量的方式规避.多核组间通过检测节点共享变量实现同步,而非显式同步的方式优化访存延迟.在节点主进程上设置节点共享变量,不同核组间通过共享变量以获取所有核组间的同步状态,以此规避了对大内存的频繁访问而带来的访存延迟.

```
//主核代码
1: for iter=0 to NITERS do
2:     sync(NODE)    //节点同步
3:     sync(ARRAY)   //主从核同步
4:     sync(NODE)    //从主核同步
5:     sync(ARRAY)   //节点同步
6:     if (ID==0)
7:         exchange(·)   //通信
8:     end if
9: end for
```

Fig.9 Code on MPE for node-sync algorithm

图 9 片上同步的主核代码

```
//从核代码
1: for iter=0 to NITERS do
2:     sync(ARRAY)   //主从核同步
3:     computing(·)   //从核计算
4:     sync(ARRAY)   //从主核同步
5: end for
```

Fig.10 Code on CPEs for node-sync algorithm

图 10 片上同步的从核代码

2.1.2 共享内存等待与核间同步相结合,优化访存延迟

神威处理器主从核异构的体系结构特点,导致主从核之间的同步需要通过访问片外存储器实现.多核访存需要经过访存队列,这种同步方式较慢,带来访存延迟.可见,片上同步的主要瓶颈在于同步需访存.神威体系结构下,主核间同步可通过节点共享变量实现,单个核组的从核间同步速度较快,无需访存.那么,如何实现所有主从核间的快速同步?我们采用共享内存等待与核间同步相结合的方式,具体做法如图 11 所示.

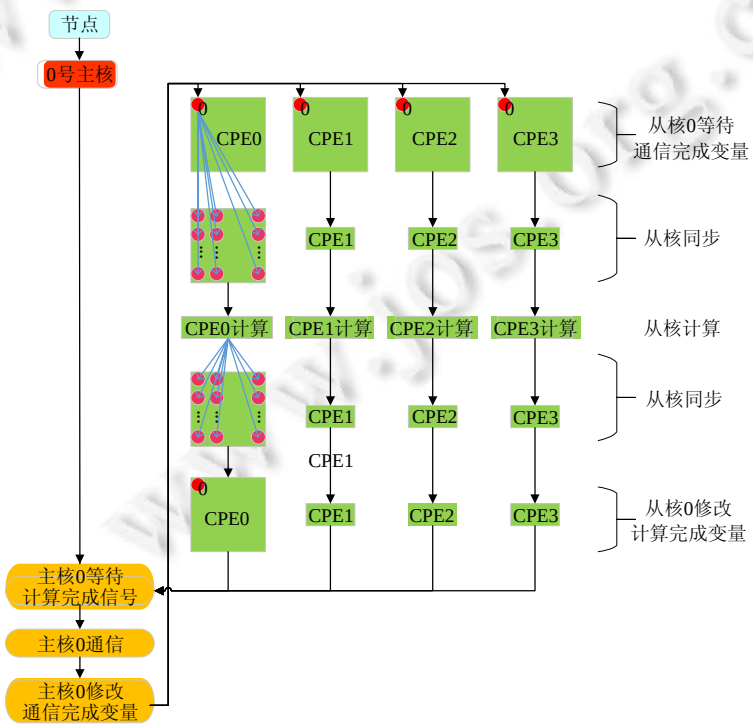

Fig.11 Process of combining shared memory waiting with inter-core synchronization

图 11 共享内存等待与核间同步相结合实现流程

流程如下:每个节点的 0 号主核负责通信,将通信完成指示变量定义为节点共享变量,节点内所有主核同时可见.那么节点内的主核间可通过节点共享变量进行同步,无需显式同步.此时,每个核组的 0 号从核等待通信完成指示变量的变化.当该变量加 1 后,说明主核已完成通信,接收到其他节点在 t 时刻的状态信息.0 号从核此时需要将此状态同步给所在核组的其余 63 个从核.同一核组的从核间同步速度较快,无需访存.每个核组的从核完成组内同步后,从核开始并行加速计算.核组内所有从核完成计算后,该核组内的 64 个从核经过一次从核同步,以保证 64 个从核状态统一.从核同步后,每个核组的 0 号从核将计算完成的节点共享变量加 1.该共享变量对所有核组的主核同时可见.所有从核加速计算的同时,0 号通信主核等待 4 个 0 号从核计算完成的节点共享变量的变化.当 0 号主核检测到 4 个 0 号从核均将计算完成指示变量加 1 后,0 号主核开始与其他节点进行通信.至此,完成一个完整的迭代步.主从核的代码实现参见图 12、图 13 的虚代码所示.

```
//主核代码
1:  for (NODE_ID==0)  //0号主核
2:    for iter=0 to NITERS do
3:      ral=r+1
4:      for v=0 to 4 do
5:        wait(waitcomp,ral)  //等待从核计算完成
6:      end for
7:      exchange(·)  //通信
8:      waitcomm=ral  //修改通信完成变量
9:    end if
10: end for
```

Fig.12 Code on MPE for shared memory waiting

图 12 共享内存等待算法的主核代码

```
//从核代码
1:  for iter=0 to NITERS do
2:    ral=r+1
3:    if (ARRAY_ID==0)
4:      wait(waitcomm,r)
5:    end if
6:    sync(ARRAY)  //从核同步
7:    computing(·)  //从核计算
8:    sync(ARRAY)  //从核同步
9:    if (ARRAY_ID==0)  //0号从核
10:     waitcomp[NODE_ID]=ral  //修改计算完成变量
11:   end if
12: end for
```

Fig.13 Code on CPEs for shared memory waiting

图 13 共享内存等待算法的从核代码

神威异构体系结构下,单核组通信的模式需利用核间同步,以同步通信数据.SW26010 芯片的硬件结构决定了主从核之间同步需要访问片外存储器,申威处理器较长的访存延迟导致该同步模式效率较低,本文采用共享内存等待的方式规避访存延迟.共享内存等待方式的优势是:核组间无需显式同步,只需要设置共享内存等待变量;该变量对节点内所有主核同时可见,只需要检测共享内存指示变量的变化来实现不同核组间的隐式同步,避免了核组同步时的访存操作,规避了申威处理器较长的访存时延.

2.2 改变计算模式,减少核间等待,优化通信延迟

2.2.1 基本的并行模式

共价键分子的原子间相互作用势不仅取决于原子间的距离,与原子间的成键方向有密切关系,因此需要考虑周围原子的影响.硅原子之间的多体作用模型采用 Tersoff 势函数[14],计算涉及两个以上的原子.计算原子 i 受到它的邻居 j 的作用力时,需要考虑 i 的其他 3 个邻居此时对 i 的影响.见图 14:由 Tersoff 势函数可得到邻居 j 对 i 的作用力 $func1$,力是相互作用力,原子 j 也受到了一个大小相等方向相反的作用力$-func1$,并且需要加到 j 原子上[17].在神威上是多线程计算,j 粒子同时在由另外一个线程计算 j 粒子与它的邻居粒子之间的受力.这个方向相反的作用力$-func1$ 在同一时刻不能同时加到 j 粒子身上.基本算法是:需要存储本线程所计算的对应原子受到其所有邻居原子的全部作用力,通过同步,传递给对应原子所在线程.所有邻居原子线程的受力计算结束后,接收消息,将原子受到的邻居粒子的反作用力叠加起来,保证所有原子作用力的计算是完整的、准确的.

对每个原子 i 而言,以计算它的邻居粒子 j=0 对它的作用力 $func1$ 为例,说明受力计算的过程.首先,计算邻居粒子 j 对 i 的作用力时,需要考虑 i 粒子其余 3 个邻居粒子 k_0,k_1,k_2 此时对 i 的影响.$func1$ 是 i 和 j 之间大小相等、方向相反的相互作用力,即 i 粒子对 j 粒子的反作用力$-func1$,需同时累加到 j 粒子上.若为并行程序,此时 j 粒子正在由其他线程计算 j 粒子与它的邻居粒子之间的受力.不能同时把$-func1$ 累加到 j 粒子上,需要交互粒子信息.最后,其他的 3 个邻居粒子也受到了反作用力$-func2$,同样需要在 3 个粒子的受力上减去.受力计算结束后,算法采用蛙跳格式更新位移.

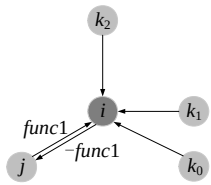

Fig.14 Inter-atomic forces

图 14 原子间作用力

在分子动力学模拟中,分子间作用力的计算时间占 80%以上[18,19],是最重要的部分,也是发挥神威机器计算性能的关键所在.在迭代过程中,计算核心需要频繁地交互粒子信息,以保证分子间作用力计算的准确性.神威机器高延迟的特点,将使粒子信息的交互遇到瓶颈,此类应用会限制神威性能的发挥.程序特点是典型的 BSP 模式,如图 15 所示.每个迭代步将粒子的位置,速度等信息存储在一个向量中,该向量包含多个分量.粒子间作用力的计算只涉及到其邻居粒子,向量的不同分量间具有局部相关性.即:后面的分量是由前面几个分量计算而来,仅与前面的几个分量有关.数据的局部相关性和依赖性,导致具有一定的异步性.

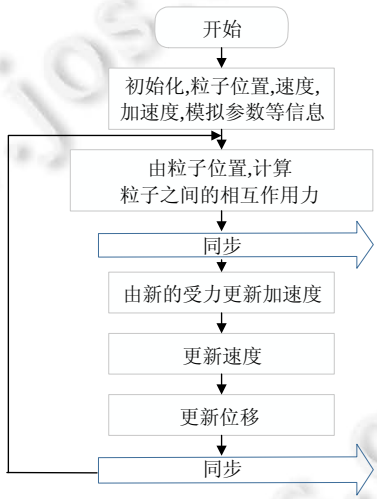

Fig.15 Implementation process of parallel algorithm

图 15 并行实现算法流程图

2.2.2 改变计算模式,减少通信次数,优化消息个数

在原来的计算模式中,向量的一个分量在计算过程中会产生一些中间数据,另外的分量可以利用.向量的不同分量之间是相互依赖的关系,是紧耦合的,导致芯片内部通过主存的这种数据传输和数据交换会大量增加.我们打破了这种依赖,变为松耦合,把原来互相写同步的计算模式变成了每个线程自己去做多步的计算,是独立计算,减少迭代步中间的同步次数,更流水化.代价是会造成计算量少量增加,但是整个吞吐率反而提高了.

多体作用模型中,如图 16 所示,计算 i 粒子受到邻居 j 的作用力时,其他 3 个邻居 k_0,k_1,k_2 也对 i 粒子产生了作用力.这意味着 j,k_0,k_1,k_2 分别受到了来自 i 粒子的反作用力,需要从它们的受力上减去.考虑 i 粒子的全部受力,不仅需要计算它受到其邻居粒子的作用力,还需要计算 i 粒子作为邻居粒子时,它所受到的反作用力.如图 17 所示,当 i 粒子作为 j 的邻居粒子时,它会受到 j 粒子对它的反作用力$-func3$.此外,如图 18 所示,j 粒子在计算它的邻居 k 对其作用力时,会对 i 产生一个反作用力$-func4$,也需要从 i 粒子的受力上减去.

具体做法如下:

(1) 计算 i 粒子受到的作用力.i 粒子所受到的作用力来自于它的所有邻居粒子.虚代码如图 19 所示.变量 j 循环 i 粒子的所有邻居,例如,i 粒子的第 0 个邻居 j=0 时,计算 j 对 i 的作用力 $func1$.变量 k 循环 i 粒

子除了 j 以外的所有邻居,它们对 i 粒子的力为 *func*2,不同的邻居,如 k_0,k_1,k_2 对 i 粒子的力不同,与位置有关;

(2) 计算 i 粒子受到的反作用力.i 粒子所受到的反作用力来自于以它的邻居粒子为主的计算过程中,如 i 粒子作为 j 粒子的邻居时,i 粒子所受到反作用力.计算 j 粒子受力时,遍历 j 的所有邻居.假设 j 的 0 号邻居为 i,计算得到 i 作为 j 的邻居时,i 受到的反作用力 *func*3,需要从 i 粒子的受力上减去.*atom*[*i*].*f*−= *func*3.若 0 号邻居不为 i,j 对 i 的反作用力 *func*4,*atom*[*i*].*f*−=*func*4.

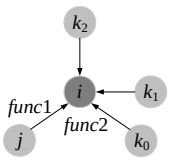

Fig.16 Force of i atom

图 16 i 粒子受到的作用力

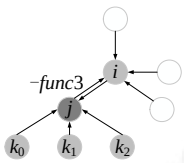

Fig.17 Reaction force *func*3

图 17 i 粒子受到的反作用力 *func*3

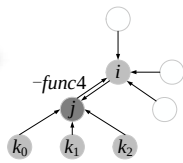

Fig.18 Reaction force *func*4

图 18 i 粒子受到的反作用力 *func*4

```
1:  for iter = 0 to NITERS do
2:    for i = 0 to NATOMS do
3:      for j = 0 to nbnum[i] do
4:        for k = 0 to nbnum[i] do
5:          if ( j != k )
6:            fijk = func0 (si , sj , sk)
7:          end if
8:          atom[i].f += func1 (si , sj , fijk)
9:        end for
10:       for k = 0 to nbnum[i] do
11:         if ( j != k )
12:           atom[i].f += func2 (si , sk , fijk)
13:         end if
14:       end for
15:       for nbj = 0 to nbj < nbnum[j] do
16:         for k = 0 to nbnum[j] do
17:           Zk0;
18:         end for
19:         if ( nblist[j][nbj] == i)
20:           atom[i].f −= func3 (j→i)
21:         end if
22:         if ( nblist[j][nbj] != i)
23:           for k = 0 to nbnum[j] do
24:             if ( i == nblist[j][k] )
25:               atom[i].f −= func4(j→i)
26:             endif
27:           end for
28:         end if
29:       end for
30:     end for
31:   end for
32: end for
```

Fig.19 Algorithm for changing the calculation mode

图 19 改变计算模式的算法

i 粒子受到的反作用力在原算法中是由负责计算它的邻居粒子受力的线程完成,如 *func*3 和 *func*4 是由负责计算 j 粒子的线程完成,那么线程间会产生相互依赖.在新的算法中,i 粒子受到的反作用力仍由负责计算 i 粒子受力的线程计算.即将互相写同步的紧耦合计算模式改为独立计算的松耦合模式.每个线程在计算粒子受力时独立计算,通信仅发生在迭代计算完成后,减少了通信次数和核间等待,优化了通信延迟.

硅粒子结晶的多体作用模型中,原计算模式下,粒子 i 只需要负责计算它的 4 个邻居粒子对它的作用力,其他粒子对 i 粒子的反作用力通过进程间同步而得.优化后的计算模式为:减少粒子 i 与其邻居粒子间的同步次数,由 i 粒子完成反作用力部分的计算.所增加的计算量是 i 粒子需计算其邻居粒子对它的所有反作用力.以 i 粒子受到其邻居粒子 j 的反作用力为例,当 i 粒子作为 j 粒子的第 1 个邻居粒子时,i 粒子受到 j 对 i 的反作用力 *func*3.

当 *i* 粒子作为 *j* 粒子的第 2 个、第 3 个或第 4 个邻居粒子时,*i* 粒子受到 *j* 对 *i* 的反作用力为 *func*4.那么 *func*3 和 *func*4 的值需要从 *i* 粒子的受力计算中减去.该部分的计算与原计算模式相比为增加的部分.计算模式的改变打破了数据依赖,将紧耦合变为松耦合,虽然带来了计算量的增加,但是提高了吞吐.

本文提出的计算模式的优化策略,适用于类似的时间演化类或高频迭代程序,当问题本身的结构特性为数据具有局部相关性和依赖性,由此可产生一定的异步性的性质,可利用本文提出的优化计算模式的方式,将不同数据之间紧耦合的依赖关系变为松耦合,将原计算模式中互相写同步的计算模式改为更流水化的独立计算.该部分的计算对计算强度影响的主要原因是,由于减少了进程间同步操作而避免了受力计算的同步更新,因此需要在本地进程额外替其他进程执行相应的计算,虽然计算强度少量增加,但该算法利用了处理器较强的计算性能,掩盖了因计算强度增加而带来的性能损耗,提高了吞吐,优化了性能.

2.3 高效访存及规则化

2.3.1 高效访存

在分子间作用力的计算过程中,从核需要频繁地访问粒子信息,从核访问主存的延迟极大地限制了神威性能的发挥.每个从核上提供了 64KB 的高速缓存空间 SPM,在算法实现上,需要频繁访问的粒子信息通过 DMA 方式批量传送至 SPM,以优化从核访主存效率.

每个线程负责 *N* 个粒子的计算,线程 *i* 将所负责的所有原子和其邻居原子的速度、位移等信息从主存加载到 SPM 中,如图 20 所示.迭代计算前,将粒子信息通过同步 DMA 加载到 SPM 中,*m_memcpy*(*MEM_TO_CACHE*),将所有粒子的邻居粒子信息通过异步 DMA 加载到 SPM 中,*m_memcpy_async*(*MEM_TO_CACHE*);迭代计算完成后,通过异步 DMA 方式,将本线程所负责粒子的更新信息从 SPM 拷贝回系统内存,*m_memcpy_async*(*CACHE_TO_MEM*).异步 DMA 方式将计算与主存访问相重叠.虚代码如图 21 所示.

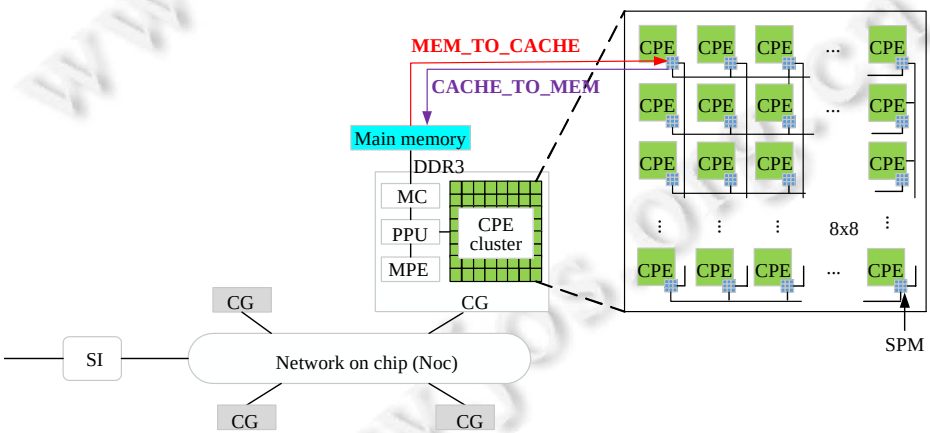

Fig.20 SPM and main memory exchange data in DMA mode

图 20 SPM 与主存间以 DMA 方式交互数据

```
1:  for iter=0 to NITERS do
2:    for i=0 to NASTH do
3:      m_memcpy(i,MEM_TO_CACHE)
4:      for nbi=0 to 4 do
5:        m_memcpy_async(nbi,MEM_TO_CACHE)
6:      end for
7:      computing(·)
8:      m_memcpy_async(i,CACHE_TO_MEM)
9:    end for
10: end for
```

Fig.21 Algorithm for efficient accessing memory

图 21 高效访存算法

2.3.2 规则化

分子动力学模拟中,以硅结晶过程为例.问题本身不一定是规则的,但是可以以尽量规则化的方法进行计算,在数据结构上尽量规则化.所谓规则化,就是按照向量长度是对齐,连续,大小规整[20].规整是按照硬件结构来规整的,具体做法是:将数据结构规则化,向量宽度规则化.程序中的主要数据结构是存储例子位移、速度、加速度等信息,有一些非规则的部分.由于神威芯片的核是向量化的、规则化的,即硬件是规则化的,我们在设计类型和数据结构的时候,将数据规则存储.由于芯片的向量化、规则化,因而规则化数据结构以提升性能.本文考虑常温下,晶体硅的模拟.硅晶体为规则的晶胞结构,有 4 个邻居粒子.受力计算中,需分别计算 4 个邻居粒子 *b*,*c*,*d*,*e* 与 *a* 粒子的距离,如图 22 所示.

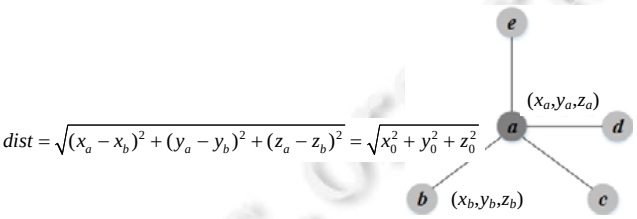

Fig.22 Distance between atoms

图 22 原子间的距离

4 次距离的计算是不凝聚的,我们针对神威核向量化的特点,考虑指令优化,进行规则化计算.三维距离是计算两点间 3 个维度上的差的平方和再开方,即:

$$dist=\sqrt{(x_a-x_b)^2+(y_a-y_b)^2+(z_a-z_b)^2} \tag{11}$$

由硅粒子的规则晶胞结构可知,*a* 粒子的受力需计算 4 个邻居粒子与 *a* 粒子的距离.我们利用规则化的数据结构,首先将粒子三维坐标的数据结构规则化.神威的并行 C 语言上扩展的数据类型为 doublev4,由 4 个双精度浮点数构成.粒子坐标即定义为 doublev4 类型的数据结构,如 doublev4 *a.s*;表示 *a* 粒子的位置,它的 4 个浮点数分别代表三维坐标和 0.规则化的数据结构去计算 *a* 与 *b*,*c*,*d*,*e* 之间距离时,每两点间需进行差平方后的求和操作.如图 23 所示,*a* 与 *b* 间的距离首先需计算公式(12):

$$(x_a-x_b)^2+(y_a-y_b)^2+(z_a-z_b)^2 \tag{12}$$

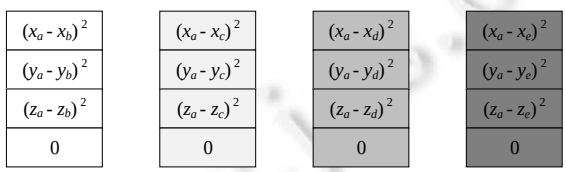

Fig.23 Vector representation of the distance between atoms

图 23 原子间的距离的向量表示

若顺序执行 4 次这样的求和指令,那么这 4 次计算是不凝聚的.考虑规则化计算,以提高访存凝聚性.规则化计算的方式进行指令优化,通过一个求和指令,实现 4 次求和计算.规则化的向量化求和是计算 4 个向量之和,而不是向量的 4 个分量之和.因而考虑将原始的 4 个向量进行转置,实现规则化计算.修改后的形式如图 24 所示.

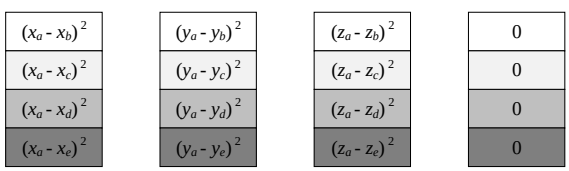

Fig.24 Distance representation after transposition

图 24 转置后的距离表示

由此可见:对转置后的 4 个向量进行向量化求和后再开平方,得到的向量的 4 个分量分别表示 4 个粒子与 *a* 粒子的欧几里得距离.由此可见:向量化的 SIMD 指令等价于一个 4 次循环操作,减少了指令数,降低了对指令访问带宽的要求,而且减少了循环引起的控制相关,提升了访存凝聚性,提高了效率.

3 实验结果

3.1 单节点性能

我们的实例是基于硅结晶的模拟,实验设计上,我们逐步对比了不同的优化方法对性能的影响,以及不同问题规模下性能的差异.在这一部分,我们主要阐述了在第 2 节中所提出的 3 种优化技术对性能的提升效果.为了说明可扩展性,我们测试了两种不同的问题规模,分别是 4 096 个粒子和 32 768 个粒子.我们的测试是运行在单节点上,启用 4 个核组.表 1 给出了神威的系统配置.

Table 1 Sunway TaihuLight supercomputer system configuration

表 1 神威太湖之光超级计算机的系统配置

处理器	SW26010 处理器
指令集	神威 64 位指令集
节点处理器	256CPEs 4MPEs
内存	每个 CG,8GB DDR3
编译语言	C 语言

图 25 描述了每一步采用不同的优化方法带来的加速比的提升,我们的优化方法最终能够实现 18x 的加速比.共享内存等待与从核同步相结合的方式对加速比的提升影响较大,紧耦合的计算模式、访存优化及规则化对性能提升也有一定的影响.从性能提升的角度,我们减少了消息个数,优化了通信以及访存延迟,对时间演化类问题迭代频率的提升做出了较大贡献.

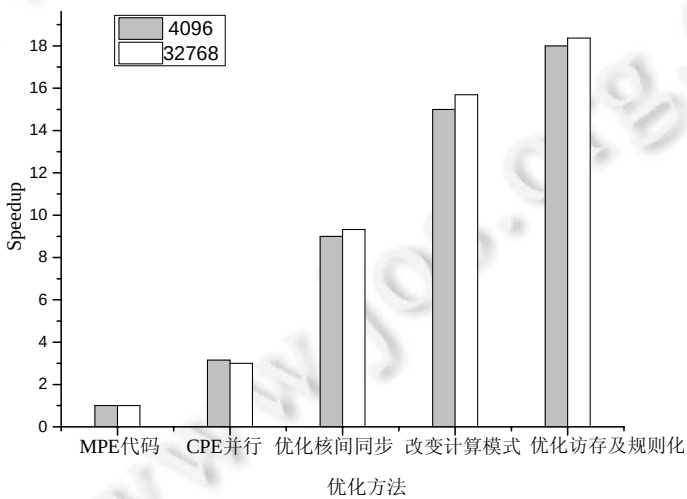

Fig.25 Speedup of different optimization methods

图 25 不同优化方法的加速比

3.2 多节点性能

我们的实验平台是基于神威太湖之光高性能集群.为测试程序的强可扩展性,我们首先测试了单个 CPU 的性能.单个 SW26010 处理器的性能取决于每个从核上所负责计算的原子个数,我们测试了每个从核上处理不同数目的原子时性能的变化,见表 2.模拟速度最大可以达到 120KSteps/s.每个 CPU 上所负责的原子个数最大为 98K,浮点利用率达到 15%.

Table 2 Single SW26010 processor performance

表 2 单 SW26010 处理器的性能

原子数	Steps/s	GFlops/s
64	120K	152
512	52K	264
2K	15K	305
98K	2.2K	450

程序的弱可扩展性测试是基于相同的数据规模,节点个数不同时,考察系统的性能.强可扩展性的测试是基于不同的粒子数,扩展系统的节点数,节点数目随着粒子数的增加而增长.我们的测试是针对典型的细粒度分布,来考察典型计算模式下,系统的速率及强可扩展性.

我们的测试利用了 SW26010 处理器的所有 4 个核组,每个处理器运行 4 个进程,但负责通信的进程只有 0 号主进程.为了验证程序的强可扩展性,我们考察了不同规模的节点个数,从 8 个节点到 32 768 个节点.

每个节点 4 个进程,最多可达到 131K 个进程.不同的系统规模下,每个进程上负责处理的原子个数固定不变,为 512 个原子.我们考察这种典型计算模式下系统的运算速率.

我们的实验为考察程序的强可扩展性,随着进程数的增加,粒子数同比例增长,即每个进程上所负责计算的粒子数目保持不变.在上述细粒度分布的条件下,我们得到这种典型计算模式下的速率,即每秒迭代的步数,见表 3 第 4 列.

Table 3 Comparison of software solution on Sunway

表 3 神威上不同系统规模下的对比

节点数	进程数	粒子数	Steps/s	us/step	Time
8	32	4 096	20 732	48.23	0.98
64	256	32 768	19 289	51.84	1.04
512	2 048	262 144	17 708	56.47	1.13
1 728	6 912	884 736	15 325	65.25	1.31
4 096	16 384	2 097 152	11 253	88.87	1.78
8 000	32 000	4 096 000	10 015	99.85	1.99
13 824	55 296	7 077 888	9 089	110.02	2.21
21 952	87 808	11 239 424	6 710	149.03	3.74
32 768	131 072	16 777 216	5 106	195.85	3.97

我们的测试通过改变系统规模,即进程数来观察系统性能的变化,如图 26 所示.纵坐标是每个迭代步所消耗的时间,横坐标表示了进程数的不同,在每个点上的数据,对应了不同的粒子数.

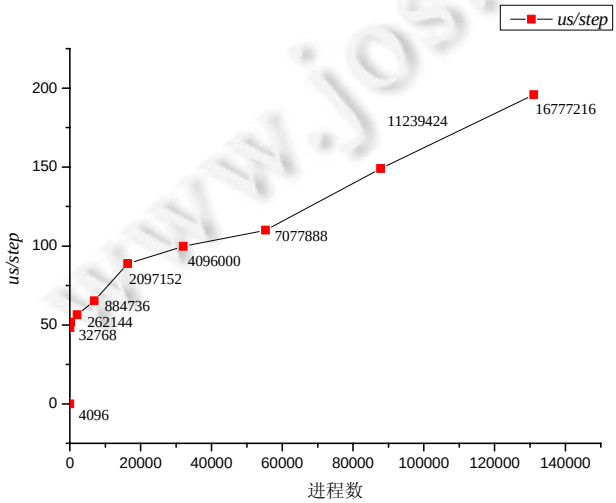

Fig.26 Time spent on each iteration step at different system sizes

图 26 不同系统规模下每个迭代步的耗时

同时,我们还测试了在不同系统规模下,程序总耗时的变化,如图 27 所示.在图 27 的折线图中,标出了每个进程点所对应的粒子个数,可观察到时间的变化.

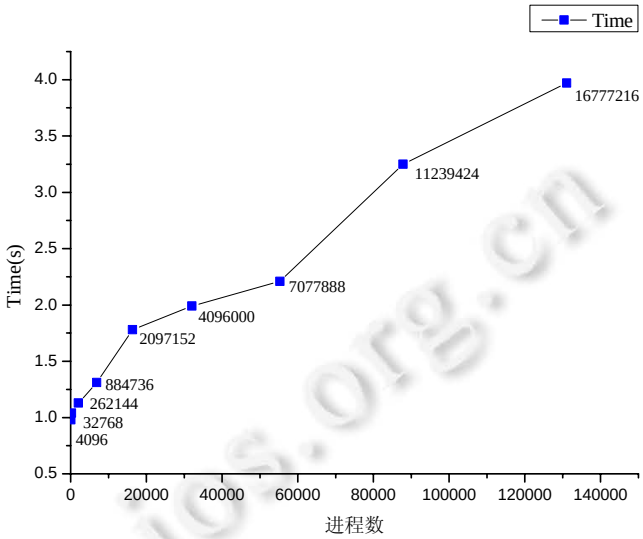

Fig.27 Time changes with different scales

图 27 不同系统规模下时间的变化

我们的目标是提高通信受限类程序的迭代频率,为了比较系统的实际性能,我们采用的度量公式如下:

$$performance=natoms\times nsteps/\text{sec}.$$

在不同的系统规模下,对应的粒子个数也不同,综合对比系统性能.如图 28 所示,系统性能随着进程数的增加而稳步提升.

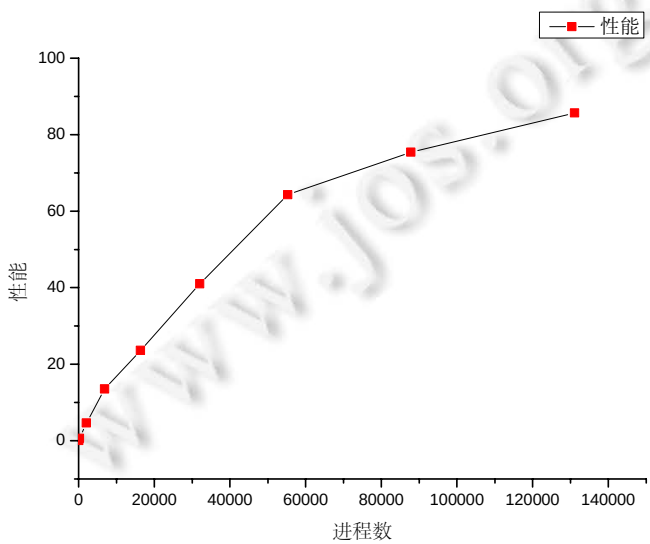

Fig.28 Performance with different system sizes

图 28 不同系统规模下的性能

由表 4 可知,本文给出的优化策略高于其他软件解决方案.当原子个数小于 10million 时,迭代速度大于 10Ksteps/s,该迭代速度高于现有的绝大部分软件解决方案.同时,较 Anton 硬件解决方案的优势是它的可扩展

性,系统规模可达到几万个处理器并行执行,同时可模拟的原子个数可达到 5 千万以上.这对大规模的分子动力学模拟提供了有价值的参考.

Table 4 Performance comparisons of hardware and software solutions for MD

表 4 分子动力学模拟软硬件解决方案的性能对比

处理器	原子数	节点数	速度(steps/s)
Anton II 2014[21]	2.2M	512	16.7K
Anton I 2007[22]	23.6M	512	67.3K
CPU+GPU 2015[23]	140K	128	1.12K
Cray XK7 2014[24]	224M	16 384	0.13K
Sunway	0.26M	512	17.8K
Sunway	0.9M	1 728	15.8K
Sunway	2.1M	4 096	11.8K
Sunway	4.1M	8 000	10.9K
Sunway	14.2M	13 824	8.2K
Sunway	22.4M	21 952	7.7K
Sunway	50.4M	32 768	5.1K

4 相关工作

计算密集型和访存密集型程序在神威“太湖之光”上得以优化.Stencil 问题具有较高的计算吞吐,在神威上实现计算-通信重叠,优化通信开销[25].优化神威上 HPCG 算法中的有效内存带宽以及增强算法的可扩展性[9].GTC-P 大规模并行模拟的高性能计算程序针对神威的访存带宽进行优化[26].

神威“太湖之光”超级计算机强大的运算能力,使其能够处理多种大规模的应用.在神威高性能集群上实现了超大规模的气象模拟[27].大规模非线性地震模拟[28]针对神威体系结构特点给出并行化解决方案.此类应用的特点是数据规模庞大,针对内存空间和带宽给出了优化方案.

时间演化类应用旨在解决的问题是提高迭代频率,加速时间演化过程.Anton[21,22]机器是针对分子动力学模拟设计的一款专用目的计算机,硬件上设计的低延迟、高带宽特点的网络以用于快速同步,但是限制了系统的物理规模.在神威上,从数据的预取和向量化角度优化[29]LAMMPS 中对内存数据的访问.针对计算密集型的 GROMACS 程序,在神威“太湖之光”上解决内存带宽限制的问题[30].

由于时间演化类应用本身数据依赖性的特点,不同处理器间的频繁通信将极大地制约迭代频率的提高.本文以减少延迟敏感的时间演化类程序的通信延迟为主要目标,优化通信,并提出几种有效的并行化策略,为类似的通信受限类程序在异构的国产化神威机器上的应用提供了蓝本.

5 总 结

在本文中,我们实现了分子动力学模拟程序在神威太湖之光超级计算机上的优化.我们的实现是基于以核组为单位的编程模式,在系统规模和网络通信能力不变的前提下,利用片上同步,减少了消息个数,优化了通信延迟.通过共享内存等待与从核同步相结合的方式,进一步优化了片上同步带来的访存延迟.同时,我们针对分子间多体作用力的计算模式进行修改,将互相写同步的紧耦合计算模式改为松耦合.减少了迭代步中间的同步次数,打破了不同线程间的依赖关系,提高了吞吐.此外,进行了访存优化以及规则化数据结构以提高访存凝聚性.我们的工作是针对诸如分子动力学模拟等延迟敏感的时间演化类应用如何提高迭代频率,给出的一系列优化技术,为类似的通信受限类程序在主从核异构的国产神威处理器上的优化提供了参考.今后的工作中,我们将进一步探索神威上的优化技术,对时间演化类程序进行高效模拟.

References:

[1] Donev A, Garcia AL, Alder BJ. Stochastic event-driven molecular dynamics. Journal of Computational Physics, 2008,227(4): 2644−2665.

[2] Evans DJ, Hoover WG, Failor BH, Moran B, Ladd AJC. Nonequilibrium molecular dynamics via Gauss's principle of least constraint. Physical Review A, 1983,28(2):1016−1021.

[3] Lai LS, Wu YQ, Shen T, Zhang N, Gao S. Molecular dynamics simulation of induced solidification process of pure liquid Fe by AI_2O_3 nanoparticles. Acta Physico-Chimica Sinica, 2012,28(6):1347−1354(8) (in Chinese with English abstract).

[4] Sugita Y, Okamoto Y. Replica-exchange molecular dynamics method for protein folding. Chemical Physics Letters, 1999,314,(1): 141−151.

[5] Yao WJ. Implementation and optimization of molecular dynamics application on Sunway TaihuLight supercomputer [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2017 (in Chinese with English abstract).

[6] Yu Y. Parallel implementation and performance optimization for refactoring GROMACS on the Sunway many-core architecture [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2018 (in Chinese with English abstract).

[7] Fu HH, Liao JF, Yang JZ, Wang LN, Song ZY, Huang XM, Yang C, Xue W, Liu FF, Qiao FL, Zhao W, Yin XQ, Hou CF, Zhang CL, Ge W, Zhang J, Wang YG, Zhou CB, Yang GW. The Sunway TaihuLight supercomputer: System and applications. Science China Information Sciences, 2016,59(7):109−124.

[8] Ni H, Liu X. Multi-core optimization technology of unstructured grid based on Sunway TaihuLight. Computer Engineering, 2019, 45(6):45−51 (in Chinese with English abstract).

[9] Ao YL, Yang C, Liu FF, Yin WW, Jiang LJ, Sun Q. Performance optimization of the HPCG benchmark on the Sunway TaihuLight supercomputer. ACM Trans. on Architecture and Code Optimization, 2018,15(1):1−20.

[10] Huang K. Many-core computing for molecular dynamic simulation [Ph.D. Thesis]. Beijing: Peking University, 2016 (in Chinese with English abstract).

[11] Rappe AK, Casewit CJ, Colwell KS, Goddard III WA, Skiff WM. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. Journal of the American Chemical Society, 1992,114(25):10024−10035.

[12] Zou XQ. Molecular dynamics simulation on physical properties of biomolecules [Ph.D. Thesis]. Beijing: Peking University, 2009 (in Chinese with English abstract).

[13] Kang JW, Hwang HJ. Gigahertz actuator of multiwall carbon nanotube encapsulating metallic ions: Molecular dynamics simulations. Journal of Applied Physics, 2004,96(7):3900.

[14] Tersoff J. New empirical approach for the structure and energy of covalent systems. Phys. Rev. B, 1988,37(14):6991−7000, 1988.

[15] National Supercomputing Center in Wuxi. Sunway User Guide (in Chinese). http://nsccwx.cn/ceshi.php?id=13

[16] Zhang L. Implementation and optimization of Samsara parallel algorithm's basic model [Ph.D. Thesis]. Beijing: Peking University, 2017 (in Chinese with English absrtract).

[17] Rapaport DC. The Art of Molecular Dynamics Simulation. 2nd ed., Cambridge: Cambridge University Press, 2004.

[18] van Meel JA, Arnold A, Frenkel D, Zwart SFP, Belleman RG. Harvesting graphics power for MD simulations. Molecular Simulation, 2008,34(3):259−266.

[19] Anderson JA, Lorenz CD, Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units. Journal of Computational Physics, 2008,227(10):5342−5359.

[20] Liu L, Tamer ÖM. Single instruction multiple data (SIMD) parallelism. In: Proc. of the Encyclopedia of Database Systems. Boston: Springer-Verlag, 2009.

[21] Shaw DE, Grossman JP, Bank JA, Batson B, Adam Butts J, Chao JC, Deneroff MM, Dror RO, Even A, Fenton CH, Forte A, Gagliardo J, Gill G, Greskamp B, Ho RC, Ierardi DJ, Iserovich L, Kuskin J, Larson RH, Layman T, Lee LS, Lerer AK, Li C, Killebrew D, Mackenzie KM, Mok SYH, Moraes MA, Mueller R, Nociolo LJ, Peticolas JL, Quan T, Ramot D, Salmon JK, Scarpazza DP, Ben Schafer U, Siddique N, Snyder CW, Spengler J, Tang PTP, Theobald M, Toma H, Towles B, Vitale B, Wang SC, Young C. Anton 2: Raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2014). 2014. 41−53.

[22] Shaw DE, Deneroff MM, Dror RO, Kuskin J, Larson RH, Salmon JK, Young C, Batson B, Bowers KJ, Chao JC, Eastwood MP, Gagliardo J, Grossman JP, Ho RC, Ierardi D, Kolossváry I, Klepeis JL, Layman T, McLeavey C, Moraes MA, Mueller R, Priest EC, Shan YB, Spengler J, Theobald M, Towles B, Wang SC. A special purpose machine for molecular dynamics simulation. ACM SIGARCH Computer Architecture News, 2008,51(2):91−97.

[23] Abraham MJ, Murtola T, Schulz R, Páll S, Smith JC, Hess B, Lindahl E. SGROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX, 2015, s1-2:S2352711015000059.

[24] Phillips JC, Sun YH, Jain N, Bohm EJ, Kalé LV. Mapping to irregular torus topologies and other techniques for petascale biomolecular simulation. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage & Analysis. 2014.

[25] Ao YL, Yang C, Wang XL, Xue W, Fu HH, Liu FF, Gan L, Xu P, Ma WJ. 26 PFLOPS stencil computations for atmospheric modeling on Sunway TaihuLight. In: Proc. of the Parallel and Distributed Processing Symp. IEEE, 2017. 535−544.

[26] Wang YC, Lin XH, Cai LJ, William T, Stephane E, Wang B, Simon S, Matsuoka S. Porting and optimizing GTC-P on TaihuLight Supercomputer with OpenACC. Journal of Computer Research and Development, 2018,55(4):875−884 (in Chinese with English abstract).

[27] Yang C, Xue W, Fu HH, You HT, Wang XL, Ao YL, Liu FF, Gan L, Xu P, Wang LN, Yang GW, Zheng WM. 10M-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis. IEEE, 2016. 6.

[28] Fu HH, He CH, Chen BW, Yin ZK, Zhang ZG, Zhang WQ, Zhang TJ, Xue W, Liu WG, Yin WW, Yang GW, Chen XF. 18.9-pflops nonlinear earthquake simulation on Sunway TaihuLight: Enabling depiction of 18-Hz and 8-meter scenarios. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis. 2017. 1−12.

[29] Dong WQ, Li KL, Kang LT, Quan Z, Li KQ. Implementing molecular dynamics simulation on Sunway TaihuLight system. In: Proc. of the IEEE Int'l Conf. on High-performance Computing and Communications, IEEE Int'l Conf. on Smart City, and IEEE Int'l Conf. on Data Science and Systems. IEEE Computer Society, 2016. 443−450.

[30] Yu Y, An H, Chen JS, Liang WH, Xu QQ, Chen Y. Pipelining computation and optimization strategies for scaling GROMACS on the Sunway many-core processor. In: Proc. of the Int'l Conf. on Algorithms and Architectures for Parallel Processing. Cham: Springer-Verlag, 2017. 18−32.

附中文参考文献:

[3] 赖莉珊,吴永全,沈通,张宁,高帅.纳米 AI_2O_3 颗粒对纯 Fe 液诱导凝固过程的分子动力学模拟.物理化学学报,2012,28(6):1347−1354(8).

[5] 姚文军.神威•太湖之光上分子动力学软件的实现与优化[博士学位论文].合肥:中国科学技术大学,2017.

[6] 余洋.面向申威众核架构的 GROMACS 并行实现与性能优化[博士学位论文].合肥:中国科学技术大学,2018.

[8] 倪鸿,刘鑫.基于神威•太湖之光的非结构网格众核优化技术.计算机工程,2019(6):45−51.

[10] 黄锟.分子动力学模拟中的众核并行计算技术[博士学位论文].北京:北京大学,2016.

[12] 邹雪晴.分子动力学模拟研究生物分子特性及相互作用[博士学位论文].北京:北京大学,2009.

[15] 国家超级计算无锡中心.神威太湖之光系统快速使用指南. http://nsccwx.cn/ceshi.php?id=13

[16] 张磊.轮回并行算法基本模型的实现与优化[博士学位论文].北京:北京大学,2017.

[26] 王一超,林新华,蔡林金,William T,Stephane E,王蓓,施忠伟,松岗聪.太湖之光上利用 OpenACC 移植和优化 GTC-P.计算机研究与发展,2018,55(4):875−884.

田卓(1984−),女,博士,CCF 专业会员,主要研究领域为高性能计算,并行计算.

陈一峯(1973−),男,博士,副教授,博士生导师,主要研究领域为并行编程语言,异构并行软件.