一个数据放置 *place*;

移动次数 *shift*;

一个记录产生移动操作的连续访问的数据对列表 *shift_data*;

一个记录产生移动操作的移动次数的列表 *shift_count*.

1. *list*0←统计入度为 0 的节点;*list*1←统计入度不为 0 节点的入度;
2. **while** *len*(*sch*)<*len*(*V*) **do**
3. **while** *D* 不为空 **do**
4. **if** *list*0 中有访问此时 port 所指向域中数据的指令 **then**
5. *sch*←满足条件的指令;更新 *list*0,*list*1;
6. **elif** *list*0 中无访问此时 port 所指向域中数据的指令且此时 port 所指向域有空闲域 **then**
7. *sch*←随机取一条满足条件的指令;更新 *list*0,*list*1;
8. 被调度指令所访问的数据 *d* 放置在此时 port 所指向的任一空闲域;
9. *D*=*D*−{*d*};
10. **else then**
11. 磁带上的数据整体左移一次,重复第 4 行~第 9 行;
12. **endif**
13. **endwhile**
14. **if** *list*0 中有访问此时 port 所指向域中数据的指令 **then**
15. *sch*←满足条件的指令;更新 *list*0,*list*1;
16. **else then**
17. *sch*←选取 *list*0 中访问数据所在域距离此时 port 所指向域最近的域中数据的指令;
18. 更新 port 的状态;重复第 14 行~第 18 行;
19. **endif**
20. **endwhile**
21. 根据 *sch* 和 *place* 统计总移动次数 *shift*,产生移动操作的连续访问的数据对 *shift_data* 及其移动次数 *shift_count*;
22. **return** *sch*,*place*,*shift*,*shift_count*,*shift_data*.

算法 2 的主要思想是:根据算法 1 生成的结果计算连续访问数据对的相关度,依据将相关度大的数据放在等价位置的原则对数据放置进行改进.如算法 2 所描述,输入为 *IAFG G*、目标系统架构以及算法 1 的输出;输出为比算法 1 总移动次数少的数据放置顺序 *place*、总的移动次数 *shift*、一个记录移动次数的列表 *shift_count*.

算法第 1 行,根据算法 1 生成的产生移动操作时连续访问的数据对的列表 *shift_data* 计算数据对连续访问的次数,并存入列表 *count* 中.第 2 行,根据列表 *count*,*shift_count* 和 *shift_data*,将连续次数 *count* 与对应数据对的总移动距离相乘得到数据对的相关度,并根据数据对的相关度的降序排列将相关度和数据对存入列表 *relate* 中.从最大相关度的数据对入手,如果数据对(*m*,*n*)中的数据 *m* 和 *n* 放置在同一个位置段,则执行第 8 行和第 9 行;否则,执行第 5 行和第 6 行.在将数据 *m* 放在等价位置处后,则原位置处的数据 *r* 被交换到数据 *m* 的原位置处.进行数据交换之后,就会导致 *r* 与原来相邻位置及等价位置处的数据的距离增加,所以需要将 *r* 当前所在域到 *m* 当前所在域之间的数据(包括 *r*,不包括 *m*)向 *r* 所在域的方向循环移一个位置.为了减少 *r* 与之前相邻域及等价域中数据的相对距离,需要对所有段中该区间的域中的数据循环移动一个位置.在进行交换位置及移动操作之后,会重新计算总的移动次数:如果移动次数减少,则会保留此次交换,更新数据放置 *place*,并重新计算相关度;如果移动次数不变或者增加,则本次交换取消,进行下一个相关度数据对的位置重置.

在交换到数据对的相关度为 1 的时候,则停止位置重置,此时的数据放置作为新的放置策略进行输出.

算法 2. GISDP 算法第 2 部分——修改数据放置.

输入:一个 *IAFG G*=⟨*V*,*E*,*D*⟩;

磁带的容量 *N*,*M* 个读/写头;

算法 1 的输出的 *sch*,*place*,*shift*,*shift_count*,*shift_data*;

输出:比算法 1 总移动次数少的数据放置顺序 *place*;

总的移动次数 *shift*;

一个记录移动次数的列表 *shift_count*.

1. *shift*←统计 *shift_data* 中每对连续访问数据的连续次数;
2. *relate*←计算 *shift_data* 中的每对连续访问数据的相关度,并根据相关度降序排序数据对;
3. **while** *relate*[*i*]>1 **do**
4. **if** 数据对(*m*,*n*)不在同一个位置段 **then**
5. 将数据 *m* 与在本位置段中和数据 *n* 偏移位置相同的数据 *r* 交换位置;
6. 对所有段中,将 *r* 所在域(及其等价域)到 *m* 所在域(及其等价域)之间的数据(包括 *r*,不包括 *m*),向 *r* 所在域的方向循环移动一个位置;
7. **else then**
8. 将其他任意一个位置段中与 *n* 偏移位置相同的数据 *r* 与 *m* 交换位置;
9. 对所有段中,将 *m* 原来放置域(及其等价域)到 *m* 当前放置域(及其等价域)之间的数据,向 *m* 原来放置域的方向循环移动一个位置;
10. 统计交换位置后的移动次数;
11. **if** 本次统计移动次数<*shift* **then**
12. *shift*=本次统计移动次数;*place*=本次放置结果;
13. 重新统计相关度,更新 *relate*;*i*=0;
14. **else then**
15. *shift* 不变;*place*=之前放置结果;*i*=*i*+1;
16. **end if**
17. **end while**
18. **return** *place*,*shift*.

在算法 2 对数据的放置进行改进之后,算法 3 根据算法 2 的数据放置和 *IAFG G* 对指令进行了重调度,以获得与新数据放置策略相适宜的指令调度.算法 3 的主要思想是:尽量将访问当前 port 所在域中数据的指令进行调度,没有可调度的指令则移动一个位置.当 port 移动到有指令可调度的域的时候,将会更新 port 的状态.最后统计该调度下的移动次数,与重调度之前的移动次数进行比较,如果移动次数减少,则保留此次的调度;否则不更新调度.

算法 3. GISDP 算法第 3 部分——改进指令调度.

输入:一个 *IAFG G*=⟨*V*,*E*,*D*⟩;

磁带的容量 *N*,*M* 个读/写头;

算法 1 的输出 *sch*,算法 2 的输出 *place*,*shift*;

输出:一个指令调度列表 *sch*;

总的移动次数 *shift*.

1. *list*0←统计入度为 0 的节点;*list*1←统计入度不为 0 的节点;
2. *i*=1;
3. **while** *len*(*sch*)<*len*(*V*) **do**
4. **if** port 此时状态下有可调度的指令 **then**

5. 调度 *list*0 中访问当前 *m* 个域中数据的指令;更新 *list*0,*list*1;
6. **else then**
7. **if** port 左移 *i* 位不会移到其他位置段 **then**
8. **if** port 此时状态下没有可调度的指令 **then**
9. port 保持原状态;
10. **if** port 右移 *i* 位不会移到其他位置段 **then**
11. **if** port 此时状态下没有可调度的指令 **then**
12. port 保持原状态;*i*=*i*+1;
13. **else then**
14. 更新 port 状态,*i*=1;调度 *list*0 中访问当前 *m* 个域中数据的指令;更新 *list*0,*list*1;
15. **end if**
16. **end if**
17. **else then**
18. 更新 port 状态 *i*=1;调度 *list*0 中访问当前 *m* 个域中数据的指令;更新 *list*0,*list*1;
19. **end if**
20. **end if**
21. **end if**
22. **end while**
23. **if** 本次统计移动次数<*shift* **then**
24. *shift*=本次统计移动次数;*sch*=本次调度结果;
25. **else then**
26. *shift* 不变;*sch*=之前调度结果;
27. **end if**
28. **return** *shift*,*sch*.

算法 1 的时间复杂度为 $O(|V|\times|D|)$,在一个磁带中,$|D|$最大为磁带容量,即数据量,$|V|$为应用中访存指令数量;算法 2 的时间复杂度为 $O\left(\frac{|V|}{2}\times(|V|+1)\right)$,其中,$\frac{|V|}{2}\times(|V|+1)$ 表示应用中连续访问数据的相关度大于 1 的数据对可能的最大个数;算法 3 的时间复杂度为 $O(|V|)$.

5 实 验

本节首先介绍实验设置,然后通过实验对比展示本文所提出的 ILP 和 GISDP 算法的效率,并对实验结果进行分析.

5.1 实验设置

本文从 MediaBench[26]基准套件中选择 8 个应用程序进行实验,这 8 个应用程序分别为 epic,ghostscript,jpeg,mesa,mpeg,pegwit,pgp 和 rasta.将本文所提出的 ILP 和 GISDP 算法与不同的算法进行比较:(1) SSDP 算法,将指令进行顺序调度并且按照数据的访问顺序进行顺序放置的一种算法; (2) S-DBC-P 算法,Chen 等人[20]提出的在固定调度的情况下,对数据进行分组放置的一个算法;(3) GISDP 算法,本文所提出的启发式算法;4) ILP 方法,本文所提出的可以得到最优解的模型.将本文提出的算法与其他算法比较之后,本文继续对启发式算法自身的三段算法进行性能提升的比较与讨论.

本文使用 SimpleScalar 模拟器[27]对基准程序进行指令提取并生产指令访问流图,同时,本文设计了磁畴壁存储器访问行为模拟器.将指令访问流图以及每条指令所访问数据输入到模拟器中,使用不同的算法可以在模拟器中生成指令调度与数据放置方案,并获得总的移动次数.在模拟器中,假设数据能够全部存储在磁畴壁存储

器中.本文分别对读/写头数量为 2,4,8 时的磁畴壁存储器进行了设计空间的探索实验.

对于 ILP 模型,使用 python3 调用 Gurobi 中的接口[28]对 ILP 模型进行编写,并运行 ILP 程序.对于某些测试程序,ILP 不能在可接受的时间范围内找到最优解,所以本文设定在 ILP 模型被执行 12 小时(43 200 秒)之后便停止执行,并将当前的结果输出.其中,最优解使用“*”标记.其他的算法均是可以在几百毫秒时间内完成的.

5.2 实验结果与分析

本节对我们的实验结果进行展示与分析.

表 2 是在配备 2 个读/写头的磁畴壁存储器上不同算法产生的实验结果.其中:“SSDP”列表示 SSDP 算法得到的移动次数;“S-DBC-P”列是 Chen 等人[20]提出的算法得到的移动次数,以及相较于 SSDP 算法移动次数减少的百分比;“GISDP”列是本文所提出的启发式算法求得的移动次数,以及分别相较于 SSDP 算法和 S-DBC-P 算法移动次数减少的百分比;“ILP”列是整数线性规划在 12 小时内求得的移动次数,以及分别相较于 SSDP 算法和 S-DBC-P 算法移动次数减少的百分比.从表 2 可以看出:本文所提的 GISDP 算法相较于 SSDP 算法,移动次数平均减少了 86.7%,相较于 S-DBC-P 算法,移动次数平均减少了 79.6%;本文所提出的 ILP 模型虽然没有在 12 小时内求出最优解,但是相较于 SSDP 算法,移动次数平均减少了 75.0%,相较于 S-DBC-P 算法,移动次数平均减少了 61.8%.

Table 2 Performance comparison for 2-port

表 2 2 个 port 的性能比较

	SSDP	S-DBC-P		GISDP			ILP		
基准测试程序	移动操作次数	移动操作次数	相对SSDP减少(%)	移动操作次数	相对SSDP减少(%)	相对S-DBC-P减少(%)	移动操作次数	相对SSDP减少(%)	相对S-DBC-P减少(%)
epic	8 394	3 492	58.4	630	92.5	81.9	1 648	80.3	52.8
ghostscript	3 308	2 116	36.0	208	93.7	90.2	642	80.6	69.6
jpeg	4 152	2 459	40.7	525	87.4	78.6	1 288	68.9	47.6
mesa	5 248	2 601	50.4	883	83.2	66.1	1 121	78.6	56.9
mpeg	8 944	4 726	47.1	1 630	81.8	65.5	2 364	73.6	49.9
pegwit	1 721	1 487	13.6	217	87.4	85.4	390	77.3	73.8
pgp	3 043	3 503	−15.1	495	83.7	85.9	973	68.0	72.2
rasta	3 253	2 117	4.1	519	84.0	83.3	887	72.7	71.5
平均减少(%)	–	–	29.4	–	86.7	79.6	–	75.0	61.8

表 3 是在配备 4 个读/写头的磁畴壁存储器上不同算法产生的实验结果.从表 3 可以看出:本文所提的 S-DBC-P 算法相较于 SSDP 算法移动次数平均减少了 93.7%,相较于 S-DBC-P 算法移动次数平均减少了 80.7%;本文所提出的 ILP 模型在 12 小时内求出的局部最优解,相较于 SSDP 算法,移动次数平均减少了 82.6%,相较于 S-DBC-P 算法,移动次数平均减少了 46.3%.

Table 3 Performance comparison for 4-port

表 3 4 个 port 的性能比较

	SSDP	S-DBC-P		GISDP			ILP		
基准测试程序	移动操作次数	移动操作次数	相对SSDP减少(%)	移动操作次数	相对SSDP减少(%)	相对S-DBC-P减少(%)	移动操作次数	相对SSDP减少(%)	相对S-DBC-P减少(%)
epic	8 359	1 912	77.1	404	95.2	78.9	1 482	82.3	22.5
ghostscript	3 286	1 124	65.8	124	96.2	88.9	466	85.8	58.5
jpeg	4 125	1 188	71.2	247	94.0	79.2	865	79.0	27.2
mesa	5 221	1 424	72.7	367	92.9	74.2	851	83.7	40.2
mpeg	8 907	2 515	71.7	679	92.4	73.0	974	89.0	61.3
pegwit	1 699	778	54.2	136	91.9	82.5	274	83.9	64.8
pgp	3 008	1 087	63.9	192	93.6	82.3	742	75.3	31.7
rasta	3 223	1 634	49.3	225	93.0	86.2	583	81.9	64.3
平均减少(%)	–	–	65.7	–	93.7	80.7	–	82.6	46.3

表 4 是在配备 8 个读/写头的磁畴壁存储器上不同算法产生的实验结果,从表 4 可以看出:本文的 S-DBC-P

算法相较于 SSDP 算法移动次数平均减少了 97.3%,相较于 S-DBC-P 算法移动次数平均减少了 85.6%;本文提出的 ILP 模型相较于 SSDP 算法移动次数平均减少了 94.8%,相较于 S-DBC-P 算法移动次数平均减少了 75.6%.

Table 4 Performance comparison for 8-port

表 4 8 个 port 的性能比较

	SSDP	S-DBC-P		GISDP			ILP		
基准测试程序	移动操作次数	移动操作次数	相对SSDP减少(%)	移动操作次数	相对SSDP减少(%)	相对S-DBC-P减少(%)	移动操作次数	相对SSDP减少(%)	相对S-DBC-P减少(%)
epic	8 342	1 019	87.8	165	98.0	83.8	434	94.8	57.4
ghostscript	3 275	618	81.1	64	98.0	89.6	204	93.8	67.0
jpeg	4 111	630	84.6	116	97.2	81.6	84*	98.0	86.7
mesa	5 207	719	86.2	136	97.4	81.1	98*	98.1	86.4
mpeg	8 888	1 383	84.4	250	97.2	81.9	156*	98.2	86.4
pegwit	1 688	455	73.0	65	96.1	85.7	50*	97.0	88.7
pgp	2 991	1 024	65.8	89	97.0	91.3	383	87.2	62.6
rasta	3 208	829	74.2	82	97.4	90.1	273	91.5	67.1
平均减少(%)	–	–	79.6	–	97.3	85.6	–	94.8	75.6

从 3 个表中可以看出:ILP 只有在配备 8 个读/写头的磁畴壁存储器上的实验结果有求出最优结果的情况,且本文的 GISDP 算法所求得结果与 ILP 最优结果相近.如表 4 中基准测试程序 pegwit 的结果,GISDP 算法的移动次数分别相较于 SSDP 算法和 S-DBC-P 算法减少了 96.1%和 85.7%,ILP 模型的移动次数分别相较于 SSDP 算法和 S-DBC-P 算法减少了 97.0%和 88.7%,所以本文的 GISDP 算法可以在多项式时间内求得近似最优解.在配备 2 个读/写头和 4 个读/写头的磁畴壁存储器上的实验结果均没有在 12 小时内找到最优解,所以这两个表中的 ILP 结果均是 12 小时得到的局部最优解.

在本文提出的 ILP 模型中,ILP 的约束条件的数量是固定的,影响 ILP 求解效率的主要因素是输入的 IAFG G 中$|V|$,$|E|$和$|D|$的大小以及读/写头的数量.当读/写头的数量为 2 时,在满足依赖关系的前提下,对指令进行调度和数据进行放置,相当于是进行全排列,并选择最优的情况.当$|E|$很小,即依赖关系很少,满足依赖的排列数接近全排列.当$|V|$和$|D|$很大时,全排列的复杂度使得 ILP 无法在多项式时间内求得最优解决策略.不过,当读/写头的数量为 8 时,一部分基准测试程序的 ILP 收敛速度有所提高(如表 4),但是所需时间仍不能与启发式算法所需的时间相比.可以看出:ILP 模型虽然可求最优解,但是不能在多项式时间内完成,不适合用在数据密集型应用的系统中.

表 5 是在配备 2 个读/写头的磁畴壁存储器上 GISDP 算法中三段算法分别得到的结果,其中,算法 2 相对于算法 1 移动次数平均减少了 9.81%,算法 3 相对于算法 2 移动次数平均减少了 9.43%.从表 5 中可以看出:部分基准测试程序在算法 2 及算法 3 中,移动次数减少了 0%,是因为不同的测试程序具有不同的数据访问特性;并且算法 2 和算法 3 均是在自身所求得的结果与前一段算法所求得的结果中选取更优的结果.因此对于部分应用,会在算法 1 或算法 2 中就求得近优解.

Table 5 Performance comparison for 3-segment of GISDP (2-port)

表 5 GISDP 的三段算法性能比较(2 个 port)

	算法 1	算法 2		算法 3	
基准测试程序	移动次数	移动次数	相对算法 1 减少(%)	移动次数	相对算法 2 减少(%)
epic	630	630	0	630	0
ghostscript	356	254	28.7	208	18.1
jpeg	911	784	13.9	525	33.0
mesa	943	883	6.4	883	0
mpeg	1 860	1 860	0	1 630	12.4
pegwit	277	243	12.3	217	10.7
pgp	522	501	4.0	495	1.2
rasta	598	519	13.2	519	0
平均减少(%)	–	–	9.81	–	9.43

实验结果显示:本文所提的算法可以找到较优的指令调度和数据放置策略,并且有效地减少了总的移动次

数.因为考虑了连续访问数据对的相关度对数据放置进行了改进,同时根据放置进行了指令重调度,所以本文所提的算法可以到达较好的性能.

6 结 论

本文针对数据密集型应用,面向配备多个读/写头的磁畴壁存储器的单核处理器系统研究最优指令调度与数据放置方案来获得最少的移动操作次数,以提升磁畴壁存储器的存储访问性能,进而提升系统性能.本文提出了可以在配备多个读/写头的磁畴壁存储器上生成最优的指令调度和数据放置方案的 ILP 模型,可以求得最小的移动次数.因为 ILP 模型的时间复杂度呈指数级,本文提出了可以在配备多个读/写头的磁畴壁存储器上,在多项式时间内生成近似最优的指令调度和数据放置方案(GISDP)的启发式算法,以此来减小移动次数.对配备不同数量读/写头的磁畴壁存储器的存储访问性能进行设计探索与实验,表明本文所提出的 ILP 模型和 GISDP 算法能够生成最优和近似最优的指令调度与数据放置方案.

由于研究问题的复杂性,本文主要考虑的是配备多个读/写头的磁畴壁存储器的单核处理器系统,但在实际用中,多核处理器系统使用更加广泛.在未来的工作中,将会研究配备多个读/写头的磁畴壁存储器的多核处理器系统中指令调度与数据放置策略,减少移动次数并提高磁畴壁存储器的存储访问性能,进而提高整个体统的性能.

References:

[1] Yang D, Ren H. The research on the technology of Internet of things and embedded system. In: Proc. of the IEEE Int'l Conf. on Software Engineering and Service Science (ICSESS). Piscataway: IEEE, 2017. 395−398. [doi: 10.1109/ICSESS.2017.8342940]

[2] Gong XQ, Jin CQ, Wang XL, Zhang R, Zhou AY. Data-Intensive science and engineering: requirements and challenges. Chinese Journal of Computers, 2012,35(8):1563−1578 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2012.01563]

[3] Dias WP, Colonese E. Performance analysis of cache and scratchpad memory in an embedded high performance processor. In: Proc. of the Int'l Conf. on Information Technology: New Generations (ITNG 2008). Piscataway: IEEE, 2008. 657−661. [doi: 10.1109/ITNG.2008.226]

[4] Chang DW, Lin IC, Chien YS, Lin CL, Su AWY, Young CP. CASA: Contention-aware scratchpad memory allocation for online hybrid on-chip memory management. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2014,33(12): 1806−1817. [doi: 10.1109/TCAD.2014.2363385]

[5] Gu SZ, Sha EHM, Zhuge QF, Chen YR, Hu JT. A time, energy, and area efficient domain wall memory based SPM for embedded systems. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2016,35(12):2008−2017. [doi: 10.1109/TCAD.2016.2547903]

[6] Gu SZ. Task scheduling and data allocation for systems with non-volatile memory [Ph.D.Thesis]. Chongqing: Chongqing University, 2016 (in Chinese with English abstract).

[7] Wang Z, Gu ZH, Yao M, Shao ZL. Endurance-Aware allocation of data variables on NVM-based scratchpad memory in real-time embedded systems. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2015,34(10):1600−1612. [doi: 10.1109/TCAD.2015.2422846]

[8] Mao HY, Zhang C, Sun GY, Shu JW. Exploring data placement in racetrack memory based scratchpad memory. In: Proc. of the IEEE Non-Volatile Memory System and Applications Symp. (NVMSA). Piscataway: IEEE, 2015. 1–5. [doi: 10.1109/NVMSA.2015.7304358]

[9] Chen XZ, Sha EHM, Jiang WW, Zhuge QF, Chen JX, Qin JJ, Zeng YS. The design of an efficient swap mechanism for hybrid DRAM-NVM systems. In: Proc. of the Int'l Conf. on Embedded Software (EMSOFT). New York: ACM, 2016. [doi: 10.1145/2968478.2968497]

[10] Chen XZ. File systems and swap mechanism for non-volatile memory systems [Ph.D.Thesis]. Chongqing: Chongqing University, 2017 (in Chinese with English abstract).

[11] Chen XZ, Sha EHM, Zhuge QF, Dai PL, Jiang WW. Optimizing data placement for reducing shift operations on domain wall memories. In: Proc. of the 2015 52nd ACM/EDAC/IEEE Design Automation Conf. (DAC). Piscataway: IEEE, 2015. 1−6. [doi: 10.1145/2744769.2744883]

[12] Parkin SSP, Hayashi M, Thomas L. Magnetic domain-wall racetrack memory. Science, 2008,320(5873):190−194. [doi: 10.1126/science.1145799]

[13] Wang YH, Yu H. An ultralow-power memory-based big-data computing platform by nonvolatile domain-wall nanowire devices. In: Proc. of the 2013 Int'l Symp. on Low Power Electronics and Design (ISLPED 2013). New York: ACM, 2013. 329−334.

[14] Venkatesan R, Kozhikkottu V, Augustine C, Raychowdhury A, Roy K, Raghunathan A. TapeCache: A high density, energy efficient cache based on domain wall memory. In: Proc. of the ACM/IEEE Int'l Symp. on Low Power Electronics & Design. New York: ACM, 2012. 185−190. [doi: 10.1145/2333660.2333707]

[15] Venkatesan R, Kozhikkottu VJ, Sharad M, Augustine C, Raychowdhury A, Roy K, Raghunatha A. Cache design with domain wall memories. IEEE Trans. on Computers, 2016,65(4):1010−1024. [doi: 10.1109/TC.2015.2506581]

[16] Mao MJ, Wen WJ, Zhang YJ, Chen YR, Li H. Exploration of GPGPU register file architecture using domain-wall-shift-write based racetrack memory. In: Proc. of the 2014 51st ACM/EDAC/IEEE Design Automation Conf. (DAC). Piscataway: IEEE, 2014. 1−6.

[17] Venkatesan R, Ramasubramanian S, Venkataramani S, Kaushik R, Raghunathan A. STAG: Spintronic-tape architecture for GPGPU cache hierarchies. In: Proc. of the 41st Annual Int'l Symp. on Computer Architecture (ISCA 2014). New York: ACM, 2014. 253−264. [doi: 10.1145/2678373.2665710]

[18] Wang YH, Yu H, Sylvester D, Kong PF. Energy efficient in-memory AES encryption based on nonvolatile domain-wall nanowire. In: Proc. of the Conf. on Design, Automation & Test in Europe (DATE 2014). New York: ACM, 2014.

[19] Zhang C, Sun GY, Zhang XY, Zhao WS. Thermal modeling and management for shift operations of racetrack memory. Journal of Computer Research and Development, 2017,54(1):154−162 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2017.20150903]

[20] Chen XZ, Sha EHM, Zhuge QF, Xue C, Jiang WW, Wang YG. Efficient data placement for improving data access performance on domain-wall memory. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, 2016,24(10):3094−9104. [doi: 10.1109/TVLSI.2016.2537400]

[21] Motaman S, Iyengar A, Ghosh S. Synergistic circuit and system design for energy-efficient and robust domain wall caches. In: Proc. of the 2014 Int'l Symp. on Low Power Electronics and Design (ISLPED 2014). New York: ACM, 2014. 195−200. [doi: 10.1145/2627369.2627643]

[22] Liu BC, Gu HF, Chen MS, Gu SZ, Chen WJ. An efficient processing in memory framework based on skyrmion material. Journal of Computer Research and Development, 2019,56(4):798−809 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2019.20180157]

[23] Liu Y, Liu X, Zhu JG. Tailoring the current-driven domain wall motion by varying the relative thickness of two heavy metal underlayers. IEEE Trans. on Magnetics, 2018,54(11):1−5. [doi: 10.1109/TMAG.2018.2827046]

[24] Hu JT, Zhuge QF, Xue C, Tseng WC, Sha EHM. Management and optimization for nonvolatile memory-based hybrid scratchpad memory on multicore embedded processors. ACM Trans. on Embedded Computing Systems, 2014,13(4):1−25. [doi: 10.1145/2560019]

[25] Gu SZ, Sha EHM, Zhuge QF, Chen YR, Hu JT. Area and performance co-optimization for domain wall memory in application-specific embedded systems. In: Proc. of the 52nd Annual Design Automation Conf. (DAC 2015). New York: ACM, 2015. 1−6. [doi: 10.1145/2744769.2744800]

[26] Lee C, Potkonjak M, Mangione-Smith WH. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. In: Proc. of the 30th Annual ACM/IEEE Int'l Symp. on Microarchitecture. Piscataway: IEEE, 1997. 330−335. [doi: 10.1109/MICRO.1997.645830]

[27] Austin T, Larson E, Ernst D. SimpleScalar: An infrastructure for computer system modeling. Computer, 2002,35(2):59−67. [doi: 10.1109/2.982917]

[28] Optimization G. Gurobi optimizer reference manual. 2015. http://www.gurobi.com

附中文参考文献:

[2] 宫学庆,金澈清,王晓玲,张蓉,周傲英.数据密集型科学与工程:需求和挑战.计算机学报,2012,35(8):1563−1578. [doi: 10.3724/SP.J.1016.2012.01563]

[6] 谷守珍.面向非易失性存储器系统的任务调度与数据分配研究[博士学位论文].重庆:重庆大学,2016.

[10] 陈咸彰.面向非易失性内存的文件系统与页面交换机制研究[博士学位论文].重庆:重庆大学,2017.

[19] 张超,孙广宇,张学莹,赵巍胜.赛道存储器移动操作的温度模型及控制策略.计算机研究与发展,2017,54(1):154−162. [doi: 10.7544/issn1000-1239.2017.20150903]

[22] 刘必成,顾海峰,陈铭松,谷守珍,陈闻杰.一种基于斯格明子介质的高效存内计算框架.计算机研究与发展,2019,56(4):798−809. [doi: 10.7544/issn1000-1239.2019.20180157]

许瑞(1996－),女,学士,CCF 专业会员,主要研究领域为调度及优化算法,非易失性存储.

谷守珍(1987－),女,博士,讲师,CCF 专业会员,主要研究领域为非易失性存储优化研究,多核并行任务调度,高性能存储优化技术.

沙行勉(1964－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为并行计算和系统,嵌入式软件和系统,物联网和智能计算,先进存储,信息安全,调度及优化算法,实时系统,大数据处理,云计算.

诸葛晴凤(1970－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为并行结构,存内计算,资源分配与调度,大数据处理系统,先进存储系统,嵌入式系统.

石亮(1987－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为计算机系统,嵌入式系统,操作系统,实时系统.

高思远(1996－),女,学士,主要研究领域为调度及优化算法,非易失性存储.