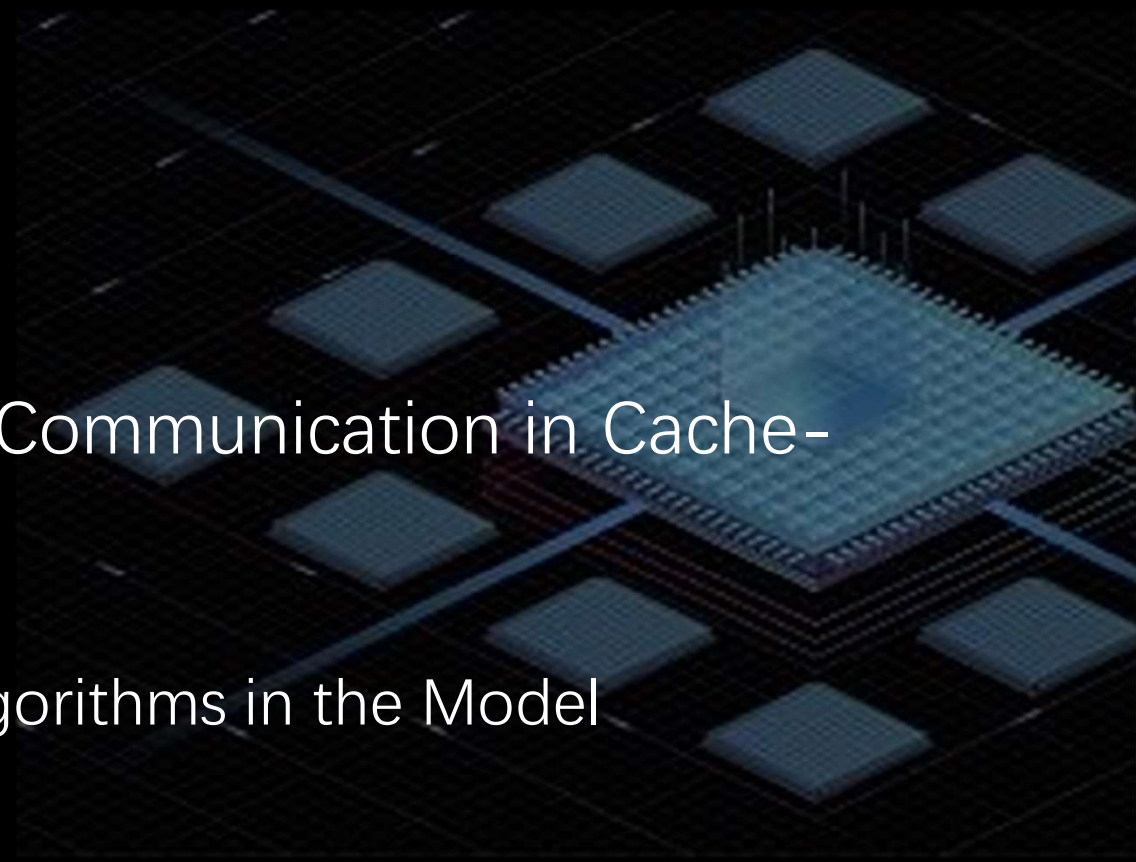
Modeling Communication in Cache-Coherent SMP Systems - A Case-Study with Xeon Phi

Sabela Ramos
Computer Architecture Group
University of A Coruna Spain
sramos@udc.es

Torsten Hoefler
Scalable Parallel Computing Lab
ETH Zurich Switzerland
htor@inf.ethz.ch

汇报人：徐诗怡 杨迪 杨智杰　指导老师：龚春叶、甘新标、杨博

content



1. Motivation
2. A Performance Model for Communication in Cache-coherent Systems
3. Communication Models
4. Designing Communication Algorithms in the Model
5. Evaluation
6. Conclusion

1 Motivation

- 处理器设计转向多核和众核方向发展。需要将传统架构（如x86）推向多核时代，转向流优化的计算。
- 传统架构为用户提供自动的cache一致性协议，是核间通信的唯一方式。它使用完全连接的交叉开关（crossbar）以及面向少量核的广播协议，这使得其不能扩展到用于更多的核间通信。因此通常被基于目录的方法所代替。
- 一些体系结构（如x86）不提供核间显式的通信，而是使用存取操作，一旦核数增多，理解其性能特性变得重要，可用于设计多核智能算法。
- 当前基于广播和目录的一致性协议由一个复杂的状态机实现，因此我们为其建立了一个性能模型，并降低其复杂度，使其能够用于算法设计和优化。

Main contributions

1.本文提出了一个新的基于状态的建模方法，为拥有cache一致性的系统提供内存通信的模型。

2.解释了这个模型如何用于Intel XEON phi平台并展示了其如何被简化后用于算法设计。

3.解释了模型是如何用于设计和优化算法并得到远远优于之前手工优化的版本的。

3. Cache一致性系统的内存通信性能模型

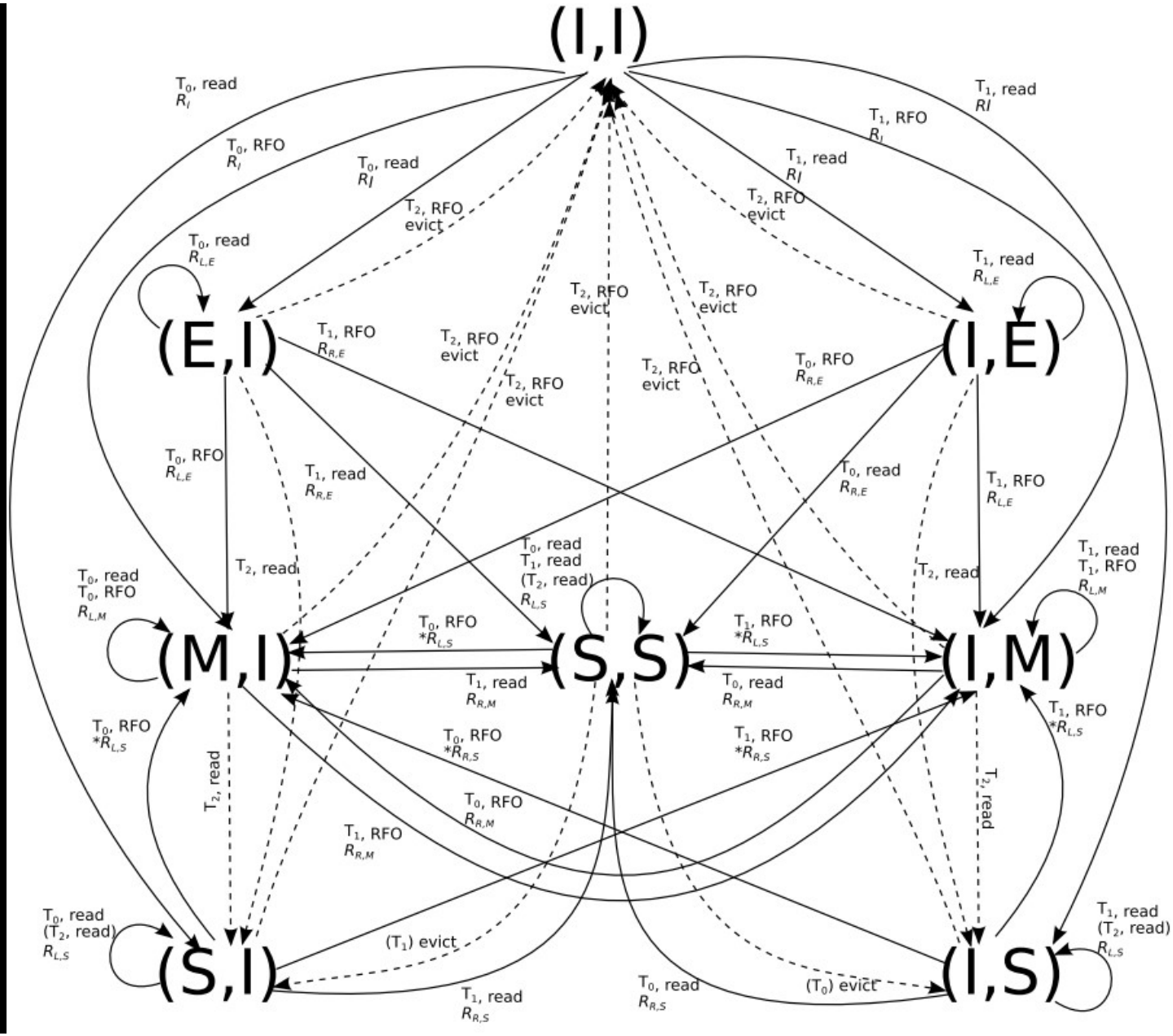
- 在大多数多核系统中，将数据从一个线程T_0传输到另一个线程T_1的唯一方法是从主存发出load和store指令。早期的多核系统，使用MESI协议。其状态如下：分为修改、专有、共享和失效4个状态。

Table 1: MESI protocol states

		Who may own it		
		This core	Other cores	Is it Modified?
M	Modified	Yes	No	Yes
E	Exclusive	Yes	No	No
S	Shared	Yes	Yes	No[1]
I	Invalid	No	Yes	-

MESI协议的转换图

两个核之间的一个cache行通信。顶点组合的状态见上表。边缘是状态转换的成本和相关操作。



2.1 Intel Xeon Phi 协处理器架构

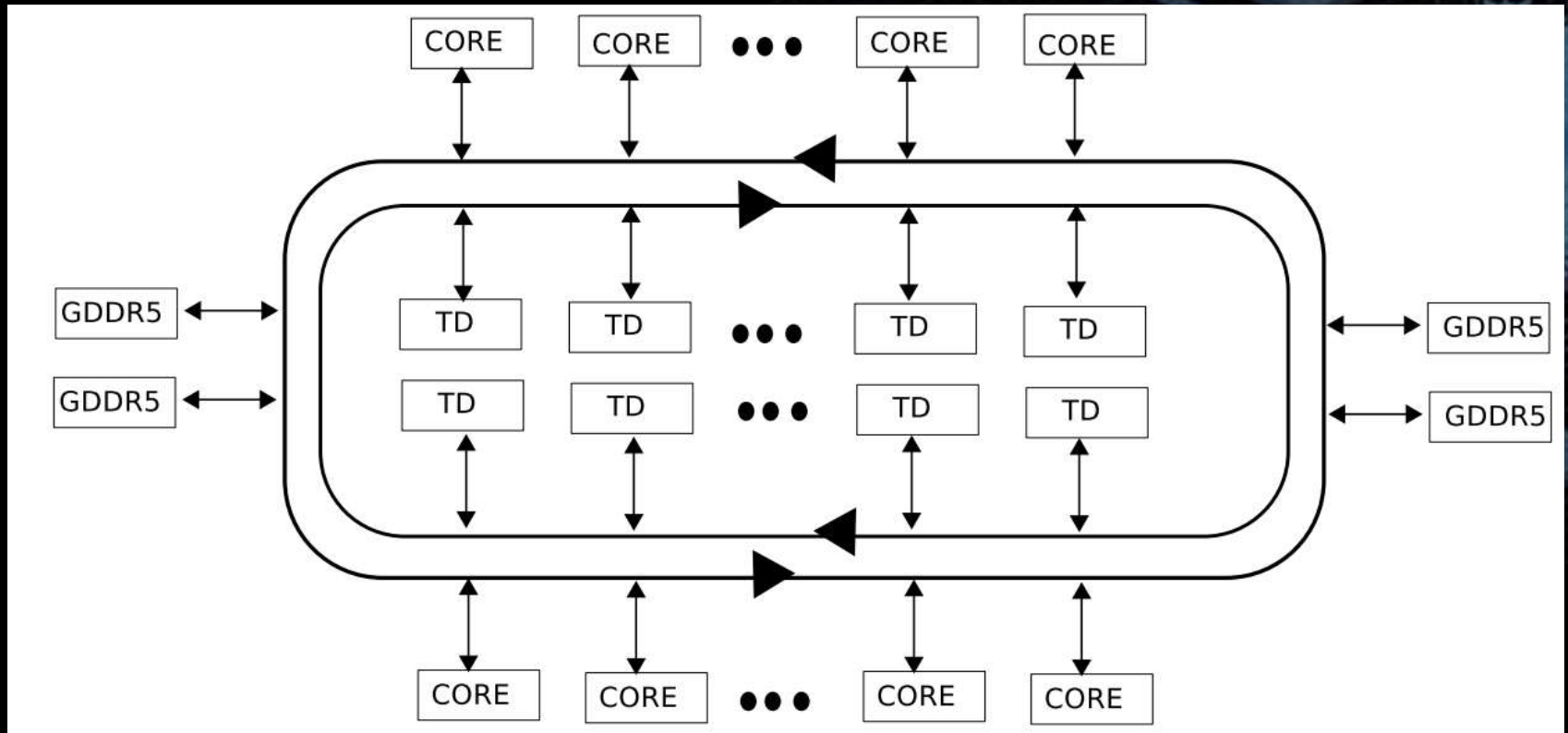


Figure 2: Architecture of the Intel Xeon Phi coprocessor

2.1.1 基于目录的cache一致性协议

Table 3: GOLS protocol states

		Number of owners	Is it Modified?
GOLS	Globally Owned Locally Shared	Several	Yes
GE /GM	Globally Exclusive /Modified	One	Yes or No (the core has M or E)
GS	Globally Shared	Several	No
GI	Globally Invalid	None	-

2.2 Xeon Phi协处理器模型参数化

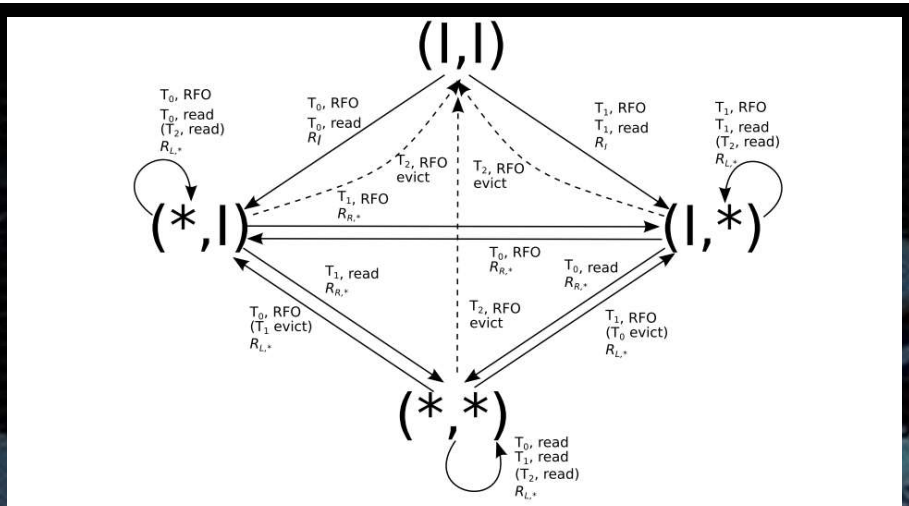


Figure 3: Graph of the simplified MESI transitions of a line within two cores

	Same core		Adjacent cores		Middle distance		Largest distance	
	avg	stdev	avg	stdev	avg	stdev	avg	stdev
M	8.6	0.2	241.2	21.7	234.7	25.6	240.1	10.4
E	8.6	0.2	227.4	20.6	235.8	25.5	237.4	27.7
S	8.7	0.9	232.0	10.2	233.4	35.0	233.4	22.5
I	277.7	34.0	274.3	25.2	278.8	34.4	284.5	29.6

(a) Results of the BenchIT [18] latency benchmark in nanoseconds

Label	Cost
$R_{L,M}$	8.6
$R_{L,E}$	8.6
$R_{L,S}$	8.7
$R_{R,M}$	234.7
$R_{R,E}$	235.8
$R_{R,S}$	233.4
R_I	277.7

(b) Model parameters (nanoseconds)

Label	Cost
$R_{L,*} = R_L$	8.6
$R_{R,*} = R_R$	235.8
R_I	277.7

(c) Simplified parameters (nanoseconds)

Table 2: Parametrizing and simplifying the model: (a) shows latency results and standard deviations for different distances, (b) extracts the relevant model parameters (cf. Fig. 1), (c) shows the simplified model parameters (cf. Fig. 3).

3.Communication Models

- The Single-line Ping-Pong Model
  - 模型用于单个cache行的ping pong 基准测试，其中发送缓冲区与一个不同的接收缓冲区通信。
  - 基准测试涉及两个内存位置和两行，而不是像前一个模型中那样只有一个位置。
- The Multi-line Ping-Pong Model
  - 把单行ping pong扩展到多行，然后研究从多个线程访问单行时竞争的影响。
  - 这将产生一个完整的cache一致性系统中的通信模型，它涵盖了所有的场景。

3.1 The Single-line Ping-Pong Model

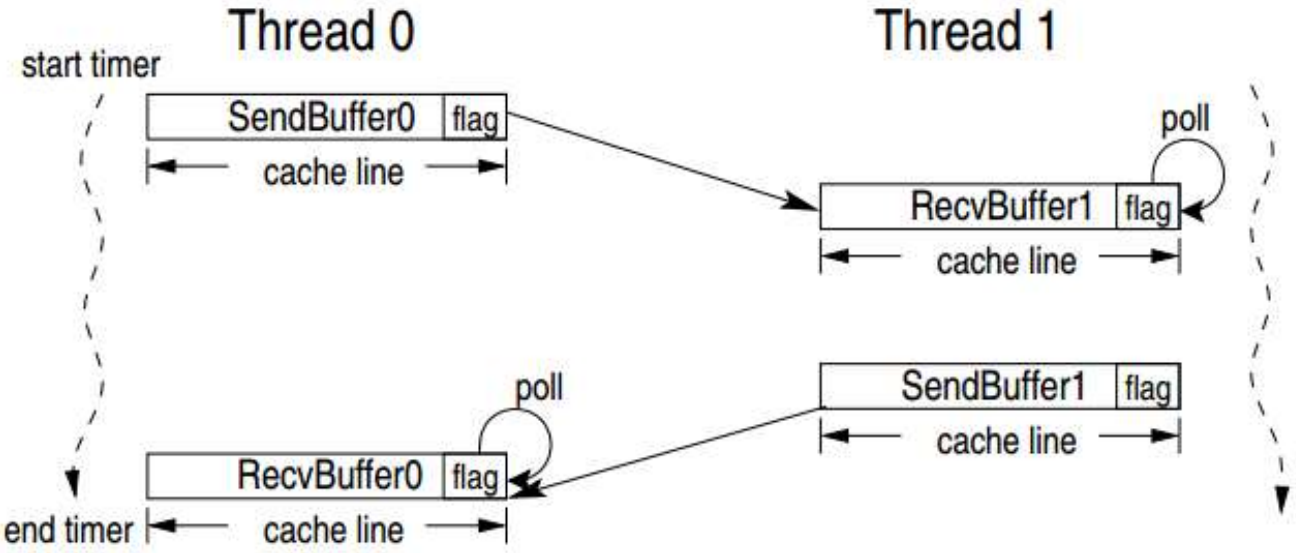
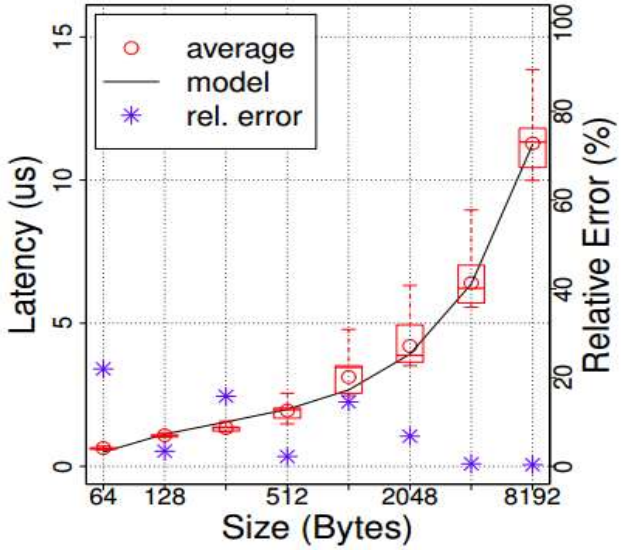
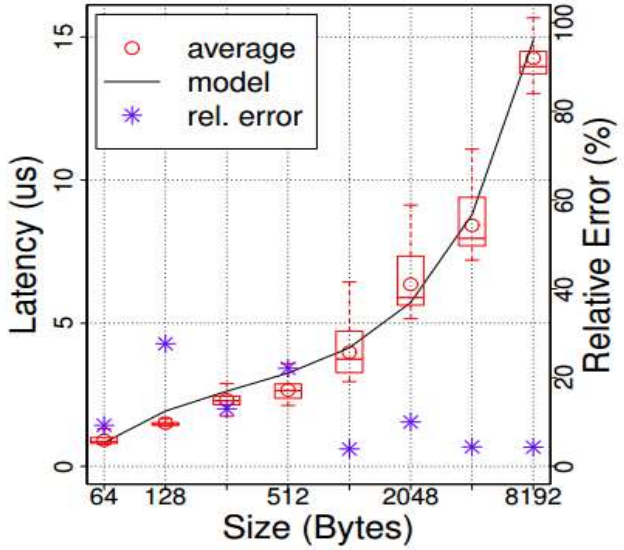


Figure 4: Ping-Pong test between two threads using four buffers.

3.2 The Multi-line Ping-Pong Model



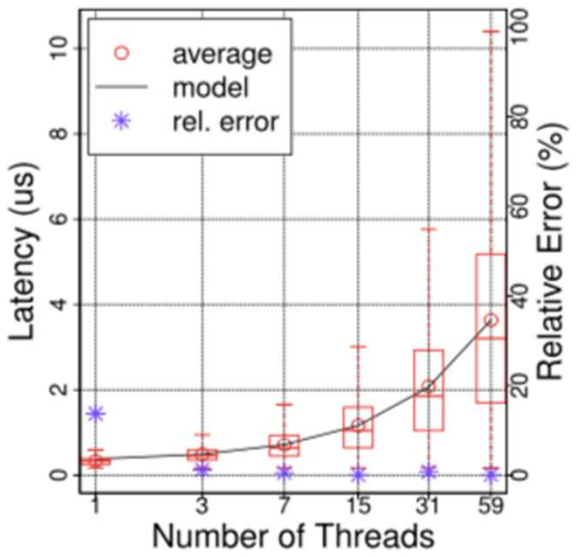
(a) Sender and receiver buffers in Exclusive state. The parameters of the model (in nanoseconds) are $76.0 \cdot N + 1521.0 - \frac{1096.0}{N}$



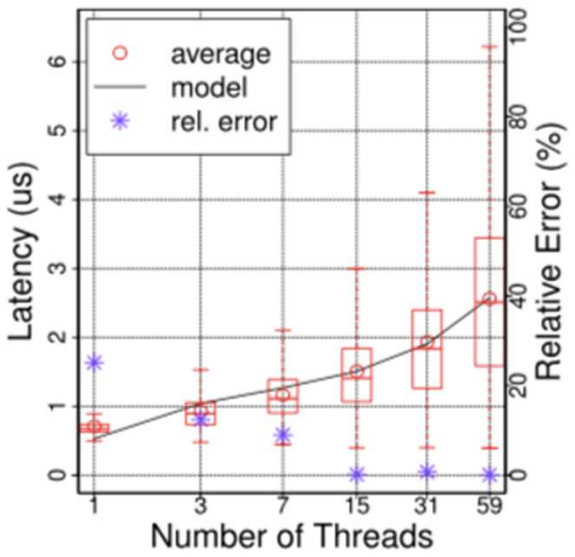
(b) Sender and receiver buffers in Invalid state. The parameters of the model (in nanoseconds) are $94.9 \cdot N + 2750.0 - \frac{2017.5}{N}$.

Figure 5: Latency and performance model for a multi-line ping-pong

3.3 DTD Contention Model



(a) Sender and receiver buffers in Exclusive state. The parameters of the model (in nanoseconds) are $320.5 + 56.2 \cdot n_{th}$.



(b) Sender and receiver buffers in Invalid state. The parameters of the model (in nanoseconds) are $23.4 \cdot n_{th} + 1202.0 - \frac{695.8}{n_{th}}$.

Figure 6: Contention in the access to the same line

4. 模型中通信算法的设计

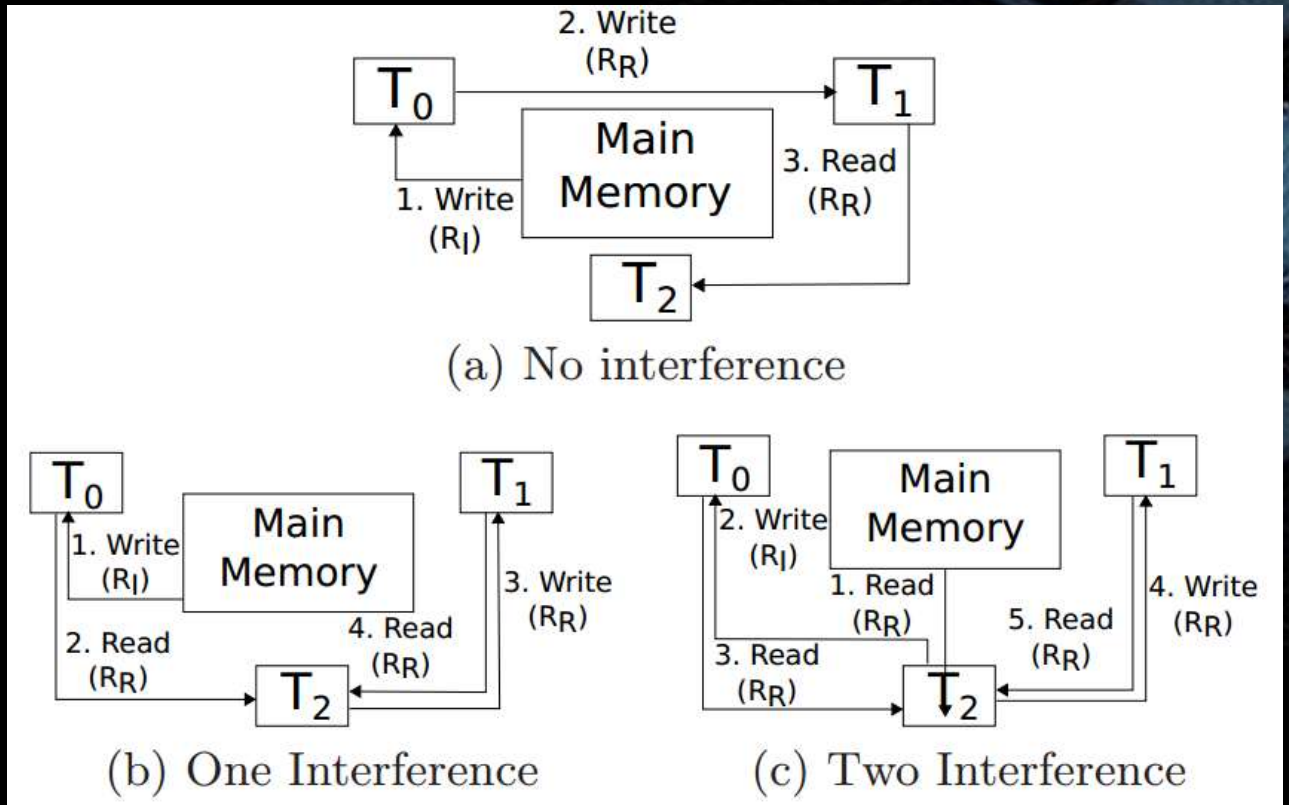


Figure 7: Interference in the access to a cache line by two writers (T_0 and T_1) and a reader that is polling the line waiting for writes.

4.1 Fast Message Broadcasting

- 广播操作包括从一个名为root的线程向其他线程发送一条消息。
- 我们将用树来讨论，因为它们是广播算法中比较常见的通信模式。
- 在广播操作中，发送方与多个接收方通信，接收方驱动的方法允许同时复制，因此，尽管有额外的确认，但仍可以为更多的线程提供更好的负载切换。

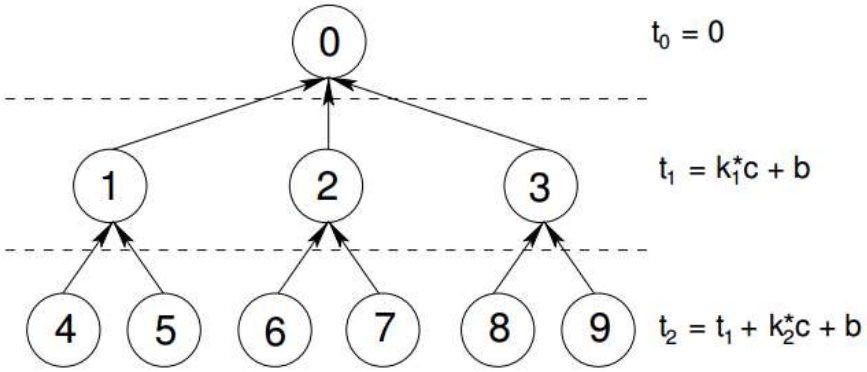


Figure 8: Tree for an 10-threads broadcast assuming d = 2, $k_1 = 3$, $k_2 = 2$.

4.2 Barrier Synchronization

- 栅栏同步涉及到每个线程都知道其他线程已经到达同步点。
- 我们将其建模为传播障碍，因为它已被证明是单端口LogP系统的最佳算法，但在我们的min-max模型中优化了参数。
- 在我们的共享内存的场景中,假设每个线程拥有一个通知行,每个“发送”操作包括设置一个标志和等待接收者承认他们读过这个标志;并且，“接收”是将相应标志的读取通知发送方。

4.3 Small Reduction

- Reduction操作是应用于从所有线程收集的数据中。
- root从多个线程接收，因此，操作与广播非常不同。前者是让所有这些线程都写入一个公共位置。然后,每个线程必须检查:
- （1）接收缓冲区是否准备好,（2）读取缓冲区,（3）将reduction操作用于缓冲区私有数据,（4）将结果写入数据缓冲区,（5）通知已完成。如果多个线程正在访问同一个缓冲区，则必须以原子方式执行步骤2到5，从而进行序列化。为了避免序列化，root有几个缓冲区，每个子代都在其中写入数据。然后根读取它们并执行操作。

5. 评价

- 设计的算法在Intel Xeon Phi协处理器5110P上进行了评估，60核1052 MHz，主机为Intel Xeon E5-2670砂桥，8核2.60 Ghz。Intel MIC软件栈是MPSS Gold update 2.1.4346-16，其中包含Intel Composer XE 2013.0.079、Intel Compiler v.13.0和In- tel MPI v.4.1.0.024。使用的基准是EPCC OpenMP基准3.0和Intel MPI基准3.2。

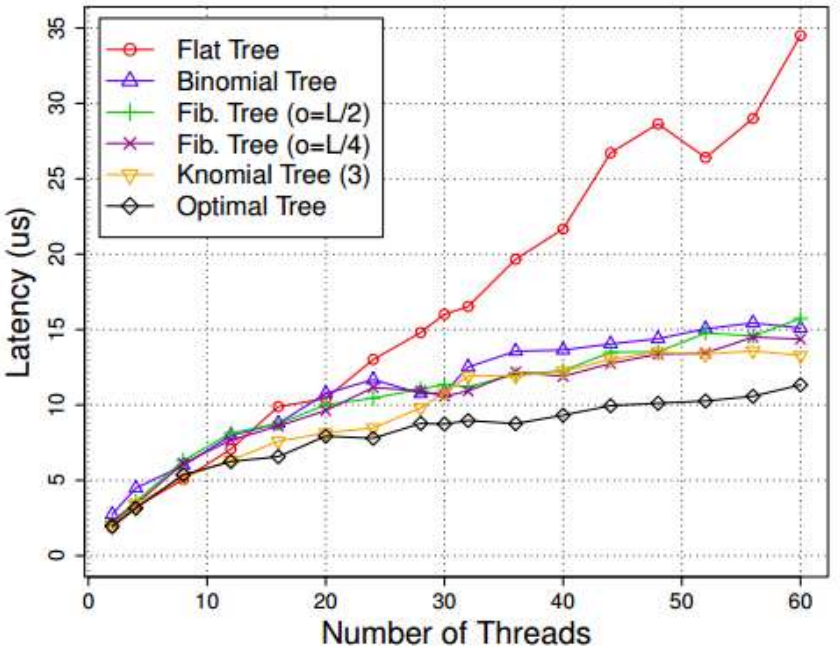


Figure 9: Small Broadcast performance comparing our optimal algorithm with widely used broadcast trees.

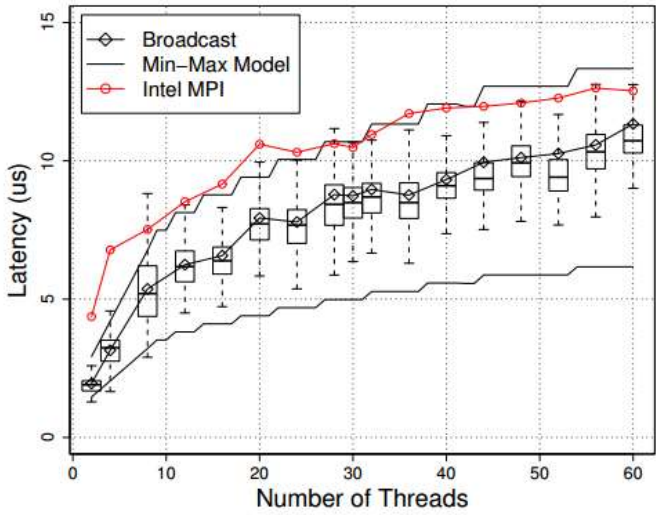


Figure 10: Small Broadcast performance compared to the model and the Intel MPI implementation.

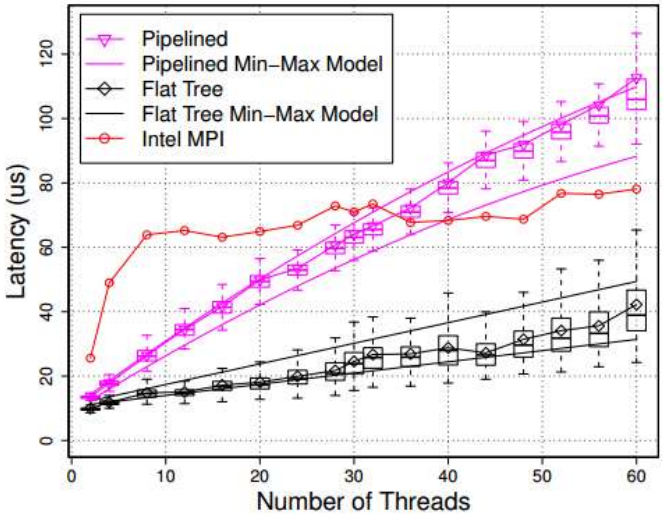


Figure 11: Large Broadcast (8 kb) algorithms compared to the model and the Intel MPI implementation.

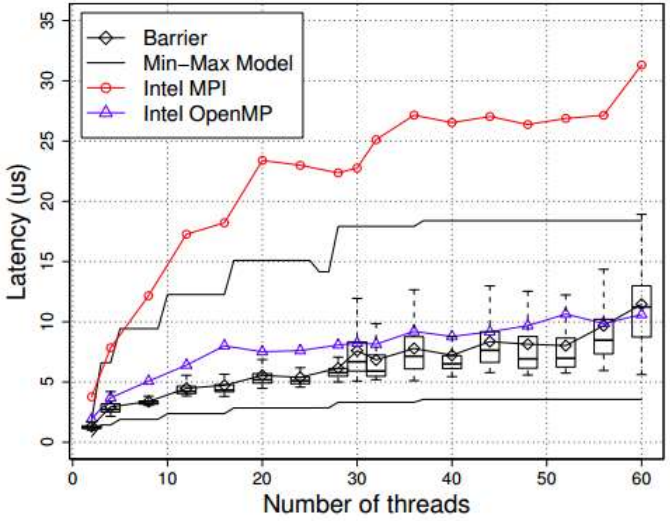


Figure 12: Barrier Synchronization results compared to the model and the Intel OpenMP and MPI implementations.

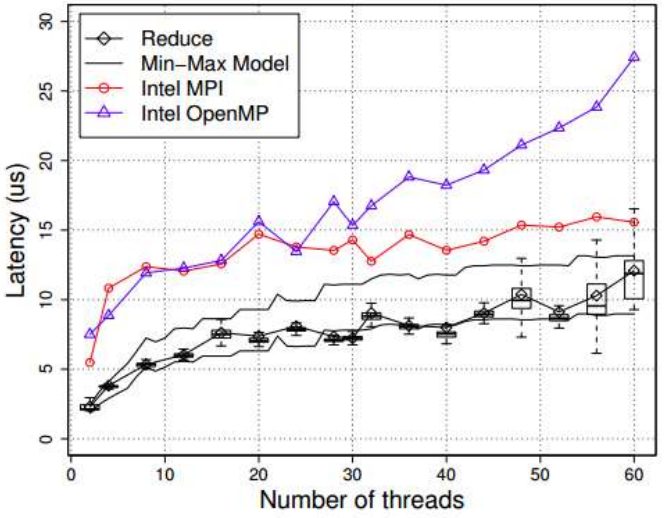


Figure 13: Small Reduction performance compared to the model and the Intel OpenMP and MPI implementations

总结

- 对于小数据，轮询阶段的通知系统和线程造成的干扰对性能的影响要大于实际的数据传输。为了对这些影响进行建模，我们不得不使用极大极小模型，这使得算法开发相当复杂。
- 然而，我们的模型允许算法设计人员从架构和cache一致性协议中抽象出来，并在纯分析的基础上设计算法。我们展示了我们的模型可以组合成一个强大的框架，用于调优和开发并行算法。
- 我们的模型描述了在cache一致性架构上设计并行算法的可行方法，可以相当用于优化和参数化已知的算法。
- 此外，我们还展示了如何开发新的和最优的算法，演示了如何使用它们来指导算法的设计和开发。
- 在分析模型的帮助下，我们开发的算法在几乎所有配置条件下都比英特尔手工调优的MPI和OpenMP库的性能有所提高，最大提高4.3倍。该方法也适用于其它体系结构和算法。