RDMA参数选择

报告人：王凯

日期：2019年4月8日

在线版：

目录

1 背景问题

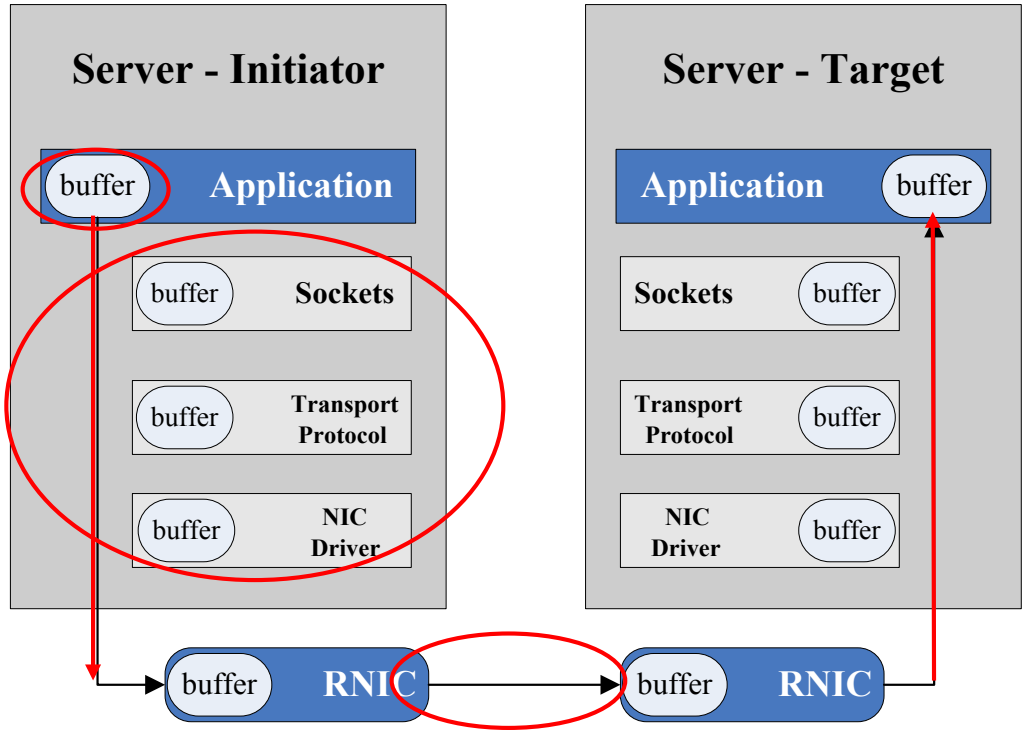
2 现有解决思路

3 本文解决思路及难点

4 实验结果

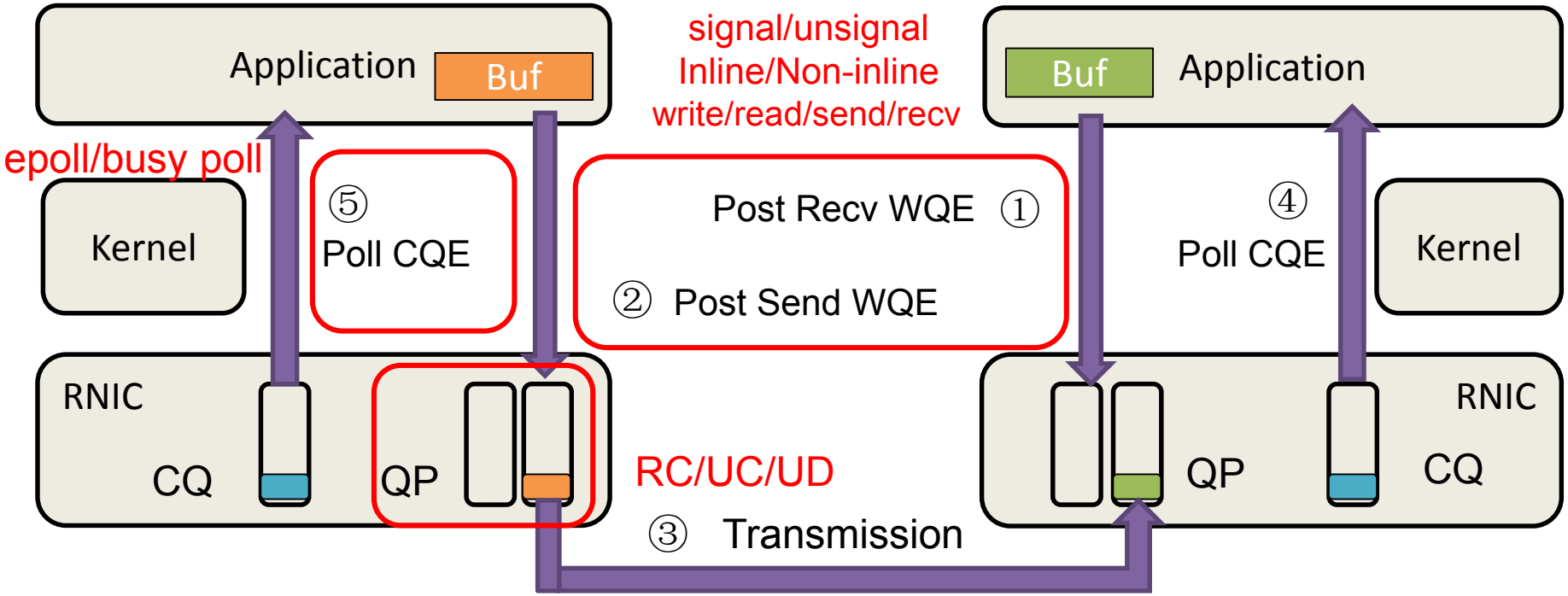
背景问题

- **R**emote
  - 远程服务器之间的数据交换
- **D**irect
  - 内核旁路技术
  - 协议栈下发到网卡
- **M**emory
  - 用户态应用虚拟内存
- **A**ccess
  - SEND/RECV，READ，WRITE等操作



背景问题

- RDMA技术将协议栈的控制功能和网络资源卸载至网卡，并对上层应用暴露多样化的硬件接口以定制化网络传输能力，是一个灰盒子。



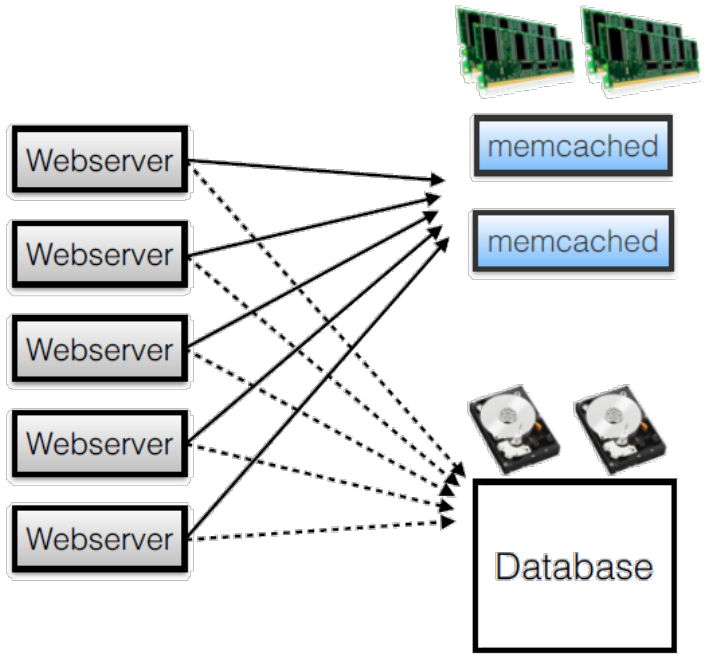
QP：Queue Pair（一组工作队列）　　WQE：Work Queue Element（工作队列中的元素）

CQ：Completion Queue（完成队列）　　CQE： Completion Queue Element（完成队列中的元素）

背景问题

- **参数选择难导致传输性能低**

signal/unsignal	是否产生完成通知
Inline/Non-inline	数据包是否和传输请求一起发送
write/read/send/recv	不同发送元语
epoll/busy poll	不同poll策略
RC/UC/UD	不同连接方式



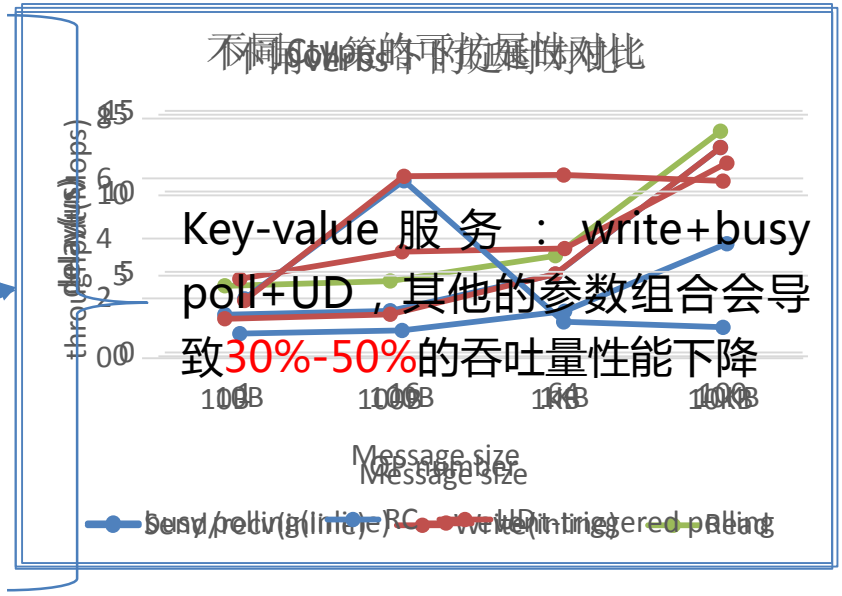
Key-value服务

数据包大小小于1KB；通信架构1->n

背景问题

- **参数选择难导致传输性能低**

signal/unsignal	是否产生完成通知
Inline/Non-inline	数据包是否和传输请求一起发送
write/read/send/recv	不同发送元语
epoll/busy poll	不同poll策略
RC/UC/UD	不同连接方式



目录

1 背景问题

2 现有解决思路

3 本文解决思路及难点

4 实验结果

现有解决思路

- Using RDMA efficiently for key-value services (SIGCOMM 2014)
  针对key-value服务利用write verb发送请求，并用UD传输模式下的send/recv元语返回数据。
- FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided (RDMA) Datagram RPCs (OSDI 2016)
  针对基于write/read实现的分布式事务处理系统存在的设计复杂、可扩展性差的问题，利用two-sided的send/recv元语解决。
- Efficient Distributed Memory Management with RDMA and Caching (VLDB 2018)
  针对分布式共享内存应用利用send/recv verb实现了控制信息的传输，同时利用write verb实现数据的传输。

- 存在问题
  - 当前的研究工作只针对特定的应用设计大量的对比实验验证在该应用特性下，如何选择参数能使应用性能得到最大的提升，而在数据中心多应用环境下，这种方式效率低，参数考虑不全面。

目录

1 背景问题

2 现有解决思路

3 本文解决思路及难点

4 实验结果

本文解决思路及难点

- 面向多应用的更为全面的动态参数选择机制
  - 对于输入的不同的应用特性，能动态地去选择更为全面的参数组合以最大化利用rdma特性，提高应用性能。
- 难点分析
  - 如何获得更为全面的应用特性与rdma参数的对应关系？
  - 为了不影响rdma的性能，动态选择过程如何更轻量化？

本文解决思路及难点

- 如何获得更为全面的应用特性与rdma参数的对应关系？
  - 不从上层应用出发去选择底层合适的RDMA参数，而是根据RDMA参数的原理机制获取对应的应用特性。
  - 确定RDMA传输过程中与应用性能相关的参数，分析参数底层实现原理获得对应的应用特性。
  - 针对每个参数，设计对比实验得到各个参数的取值和其所对应的应用特性的具体关系。

本文解决思路及难点

- 如何获得更为全面的应用特性与rdma参数的对应关系？

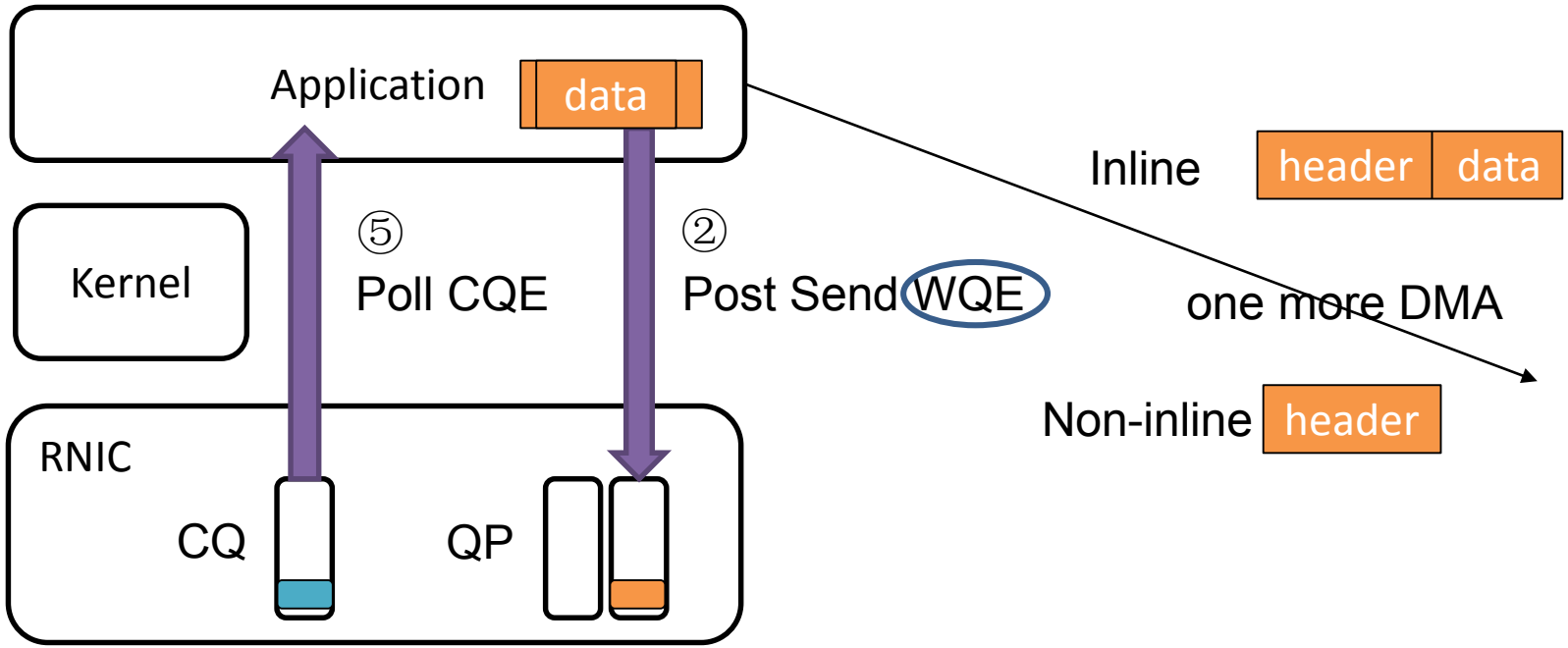
举例：

- Inline/Non-inline：inline模式可以使得数据包随着发送请求（WQE）一起发送给网卡，因此能减少一次数据DMA的过程，降低延时，但是只适合于一定数据包大小之内的数据包。

本文解决思路及难点

- 如何获得更为全面的应用特性与rdma参数的对应关系？

举例：



本文解决思路及难点

- 如何获得更为全面的应用特性与rdma参数的对应关系？

举例：

- Inline/Non-inline：inline模式可以使得数据包随着发送请求（WQE）一起发送给网卡，因此能减少一次数据DMA的过程，降低延时，但是只适合于一定数据包大小之内的数据包。

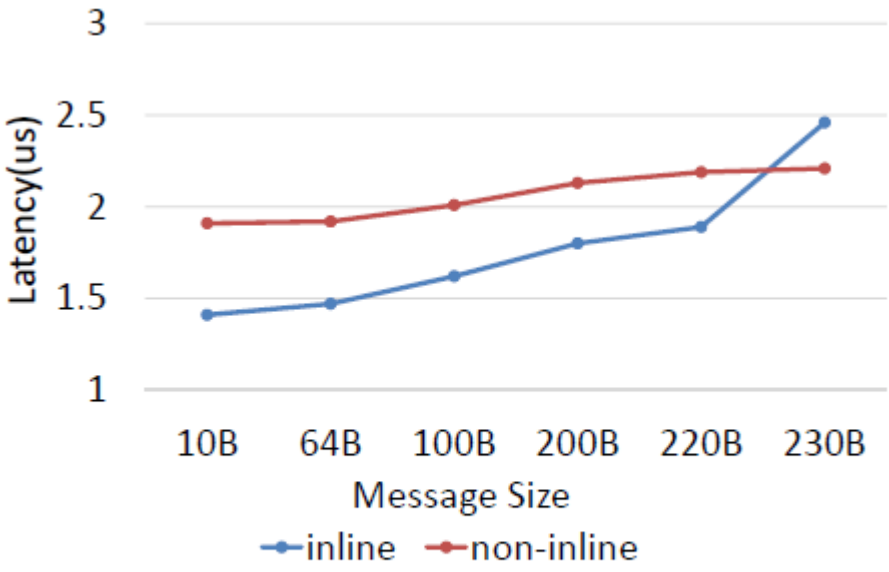
- RC/UD：RC需要在两个只能一对一通信的QP之间建立连接，而UD模式下，一个QP可以和多个QP通信；UD只支持最大MTU数据包大小的传输。

本文解决思路及难点

- 如何获得更为全面的应用特性与rdma参数的对应关系？

举例：

- 如右图所示，通过不同数据包的对比实验得到inline/non-inline 参 数与数据包大小这个应用特性的具体关系，比如本图220B以下的数据包更适合inline。

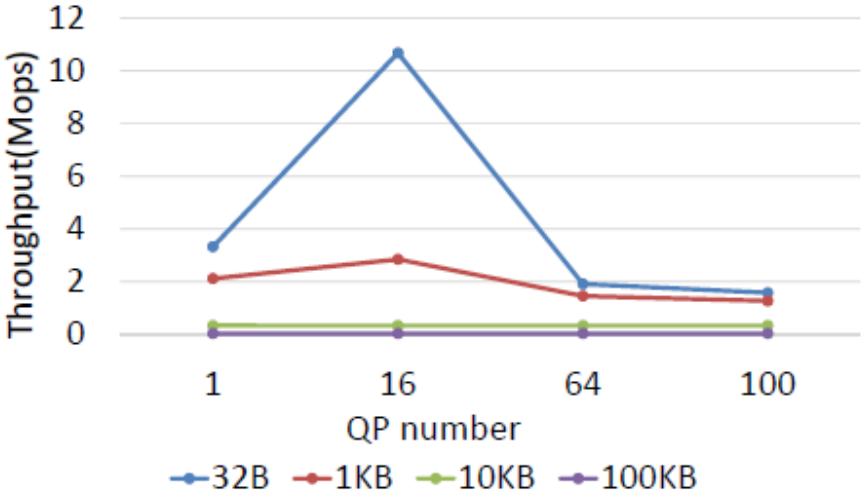


(a) Inline/non-inline

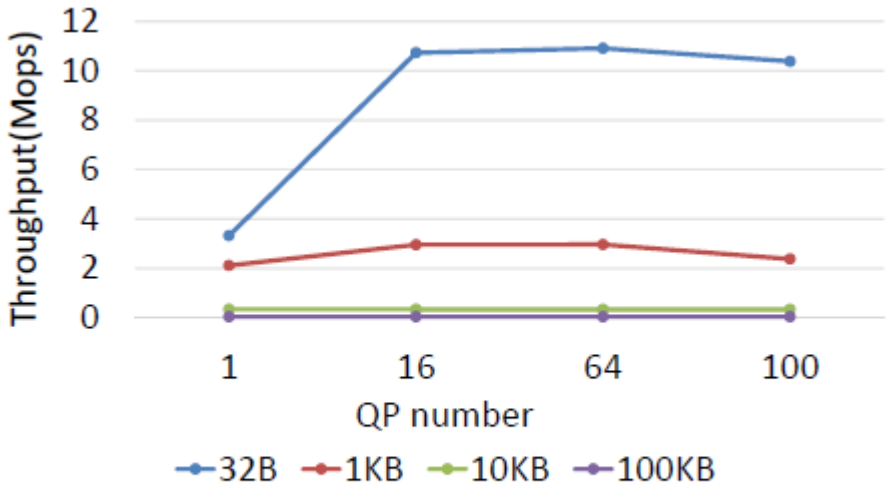
本文解决思路及难点

- 如何获得更为全面的应用特性与rdma参数的对应关系？

举例： 如下图所示，通过不同数据包在不同QP数量下的吞吐量对比，得到连接类型的参数选择和数据包大小以及应用通信框架规模的具体关系。



(c) Connection type RC

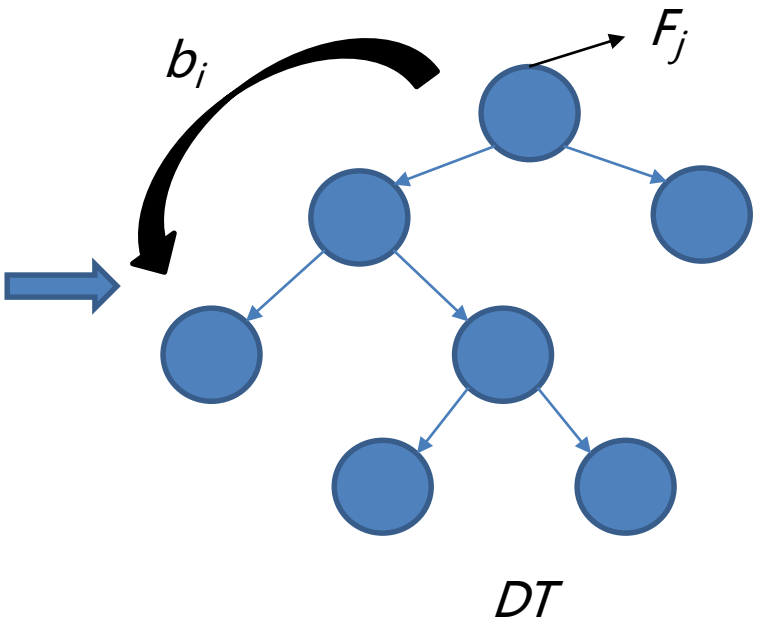


(d) Connection type UD

本文解决思路及难点

- 为了不影响rdma的性能，动态选择过程如何更轻量化？

Mtype	CPU_L	CPU_R	CPU_L vs CPU_R	CF	Msize	
	enough				>10KB	epoll
	enough				<10KB	bpoll
	lack					epoll
					<64B	inline
					>64B	Non-inline
control						signal
data						unsignal
				1->n	<4KB	UD+SEND
control	enough	lack		1->1		RC+SEND
control	enough	enough		1->1		WRITE
control	lack	enough		1->1		READ
control	lack	lack	<	1->1		RC+SEND
control	lack	lack	>	1->1		READ
control	enough	lack		1->n	>4KB	RC+SEND
control	enough	enough		1->n	>4KB	WRITE
control	lack	enough		1->n	>4KB	READ
control	lack	lack	<	1->n	>4KB	RC+SEND
control	lack	lack	>	1->n	>4KB	READ



本文解决思路及难点

- 为了不影响rdma的性能，动态选择过程如何更轻量化？
  - 9个判断条件：

1. Mtype==control?
2. CPUl==enough?
3. CPUr==enough?
4. CPUl<CPUr?
5. CF==1->1?
6. Msize<64B?
7. Msize<4KB?
8. Msize<100KB?
9. Msize<1MB?

  - 必要判断条件：1,2,5,6
  - 依赖关系：

5 is false —> 7

2 is true —> 8

2 is false or 1 is true —> 3

2 is false and 3 is false —> 4

2 is false and 3 is true and 1 is false —> 9

本文解决思路及难点

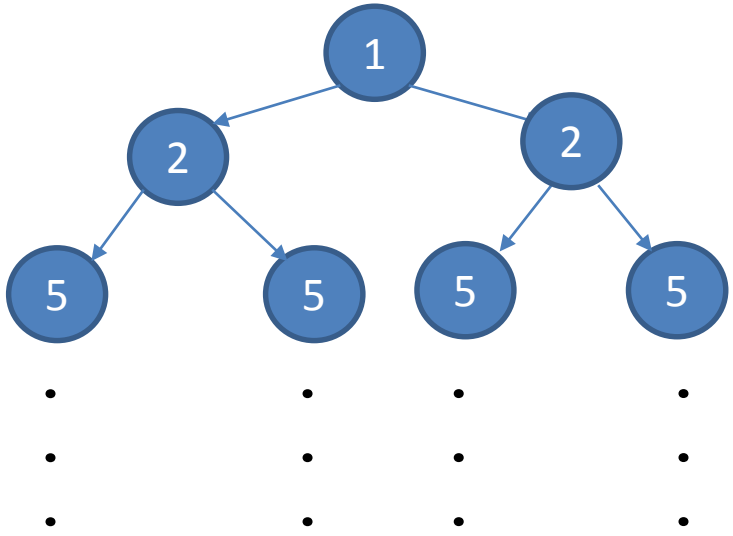
- 为了不影响rdma的性能，动态选择过程如何更轻量化？

Algorithm 1 DT Construction

Input: decision conditions set S_c, dependency relations set of conditions S_r

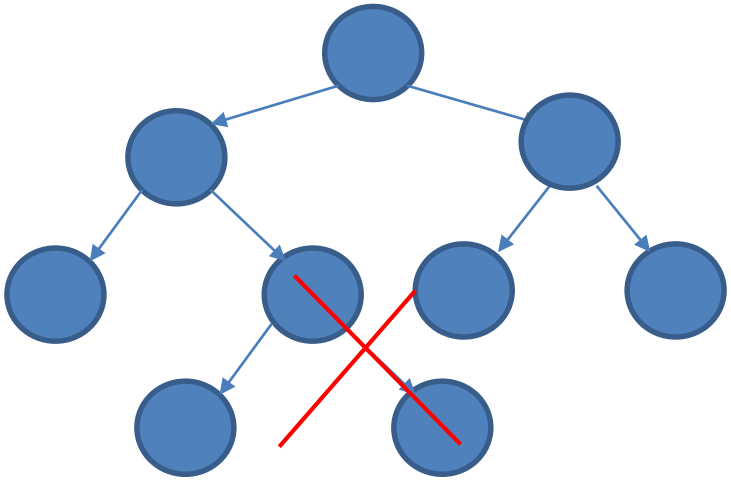
Output: a Decesion Tree DT

```
1:  Choose S_c^1, S_c^2, S_c^4 as the first three decision nodes out of
    order and initialize depth = 3, update S_pr
2:  for each branch in current depth do
3:      S_cc ← search S_r with S_pr
4:      if N_Msize in S_cc >= 3 then
5:          current decision node ← C_m
6:          S_cc -= C_m
7:      else
8:          current decision node ← C_r
9:          S_cc -= C_r
10:     end if
11: end for
```



本文解决思路及难点

- 为了不影响rdma的性能，动态选择过程如何更轻量化？



- 由于RDMA自身硬件限制，有些组合是不支持的，可以从树中剪枝掉：
  - ✓ read + inline
  - ✓ UD + write
  - ✓ UD + read

目录

1 背景问题

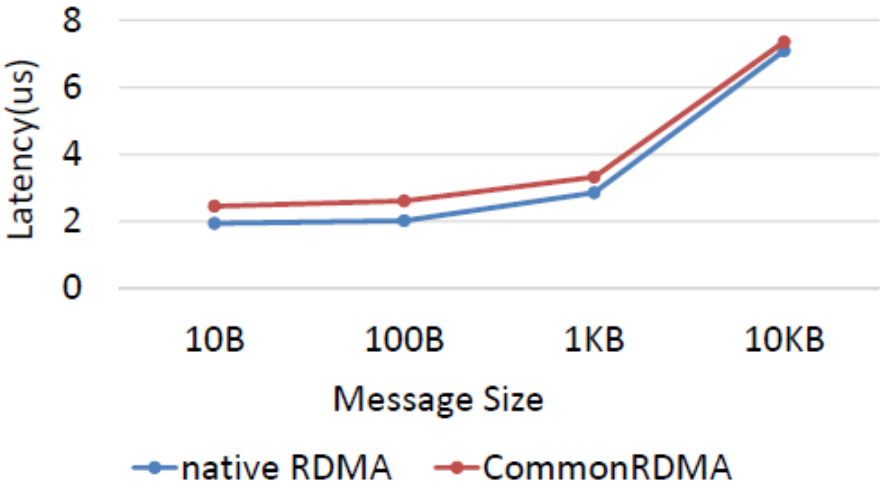
2 现有解决思路

3 本文解决思路及难点

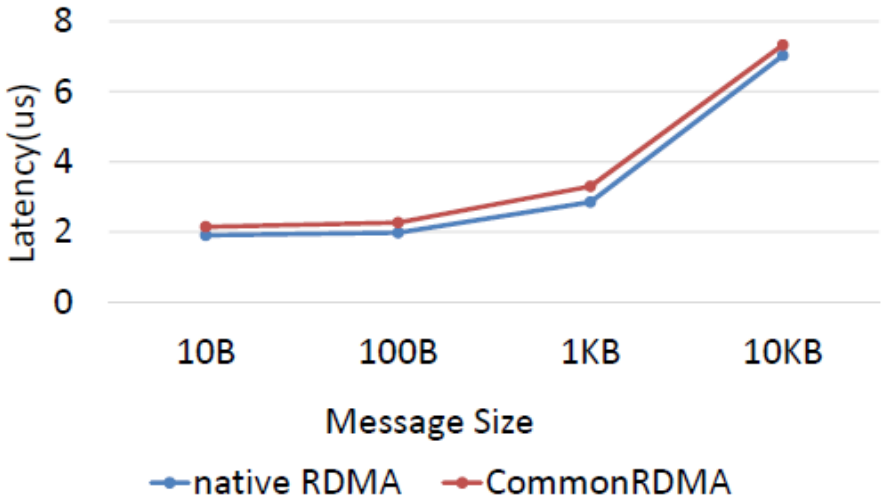
4 实验结果

实验结果

- 轻量化的动态参数选择验证
  - 相比于原生的RDMA，参数选择只需要很小的开销（<1us）。



(a) SEND/RECV latency



(b) WRITE latency

实验结果

- 参数选择对应用性能的提升验证
  - 相比于现有的key-value系统，本文的key-value系统性能更好。

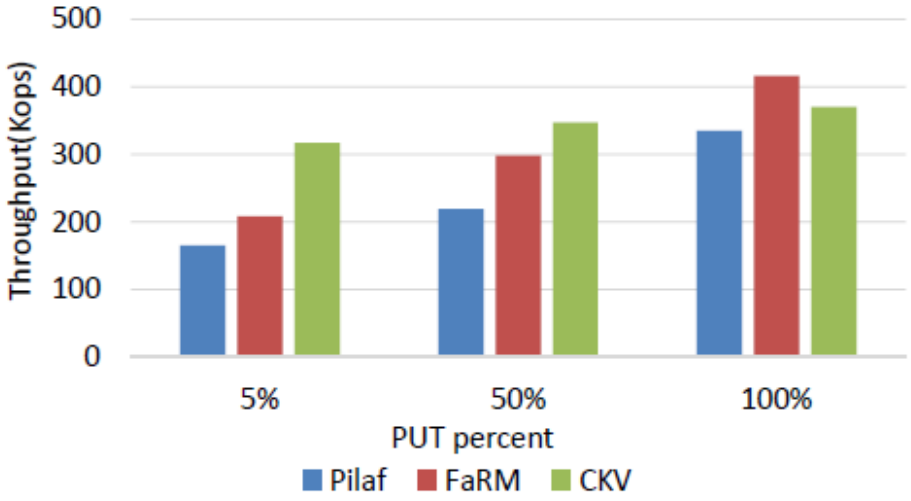


Figure 6: Throughput comparison for 48 byte key-value items

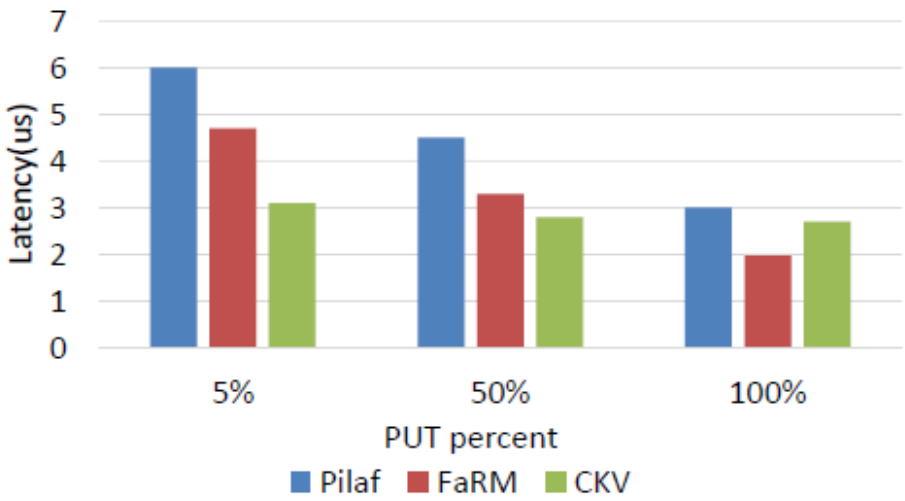


Figure 7: Latency comparison for 48 byte key-value items