Impact of Synchronization Topology on DML Performance

ABSTRACT

- 研究的主题：逻辑拓扑、物理拓扑和传输对DML参数同步的影响
- 研究的方向：从现有的物理拓扑和逻辑拓扑出发，分析对比他们的Global Synchronization Time
- 创新点：新的同步算法Hierarchical Parameter Synchronization

INTRODUCTION

参数的同步会收到三个方面的影响

- 逻辑拓扑
  - parameter server-based
  - mesh-based
  - ring-based
- 物理拓扑
  - Fat Tree
  - BCube
  - Torus
- 传输协议
  - TCP
  - RDMA

BACKGROUND

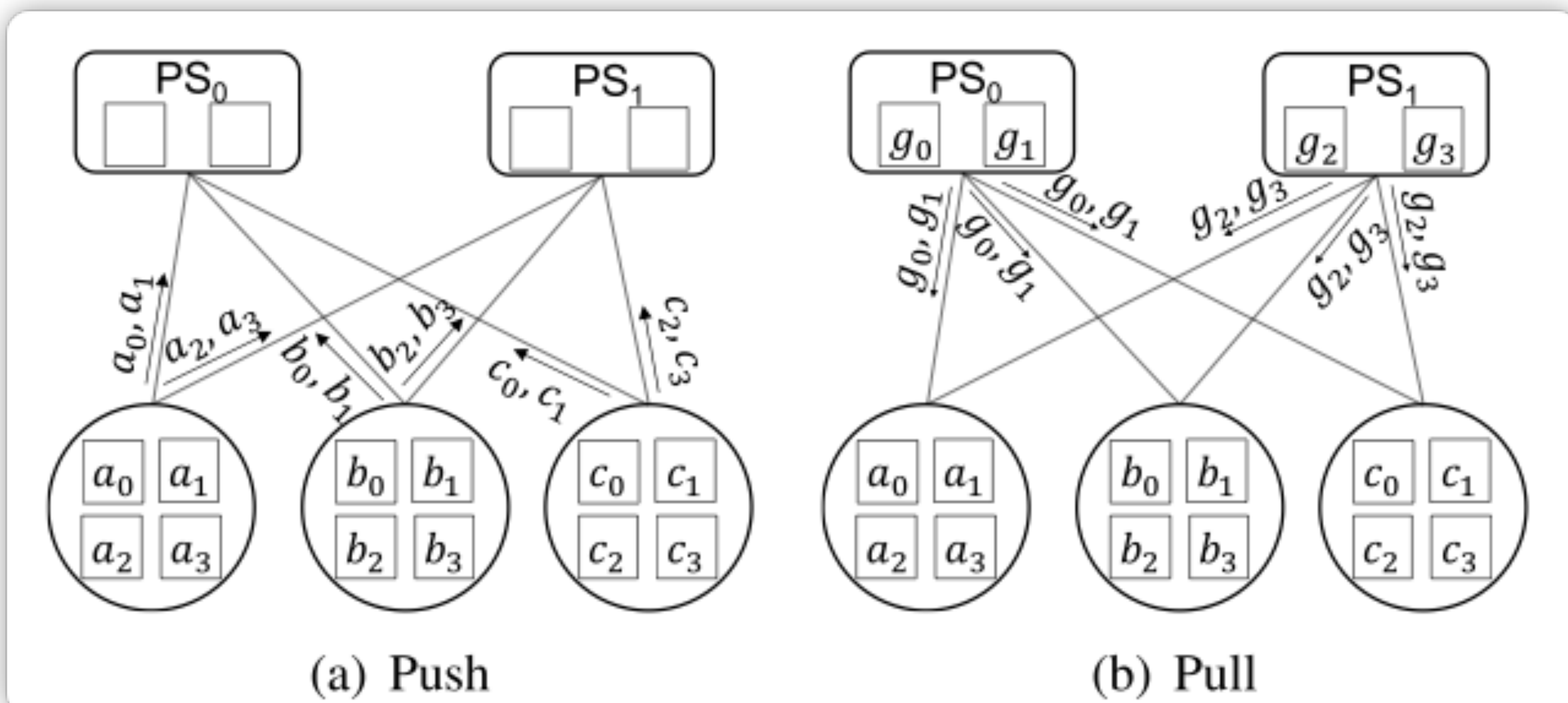
Logical Synchronization Topology

- PS-based Synchronization
- Mesh-based Synchronization
- Ring-based Synchronization

BACKGROUND

PS-based Synchronization

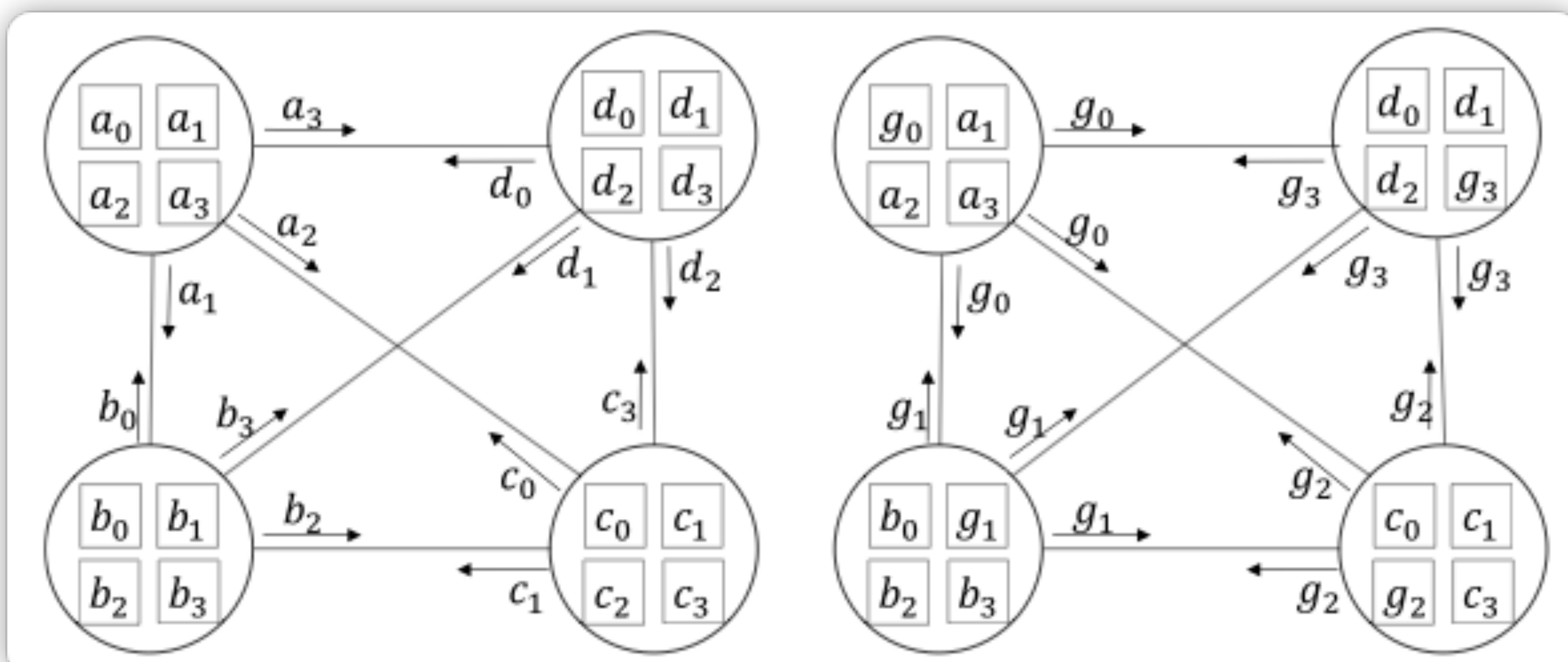
- 中心化同步算法
- Server分为Worker和Parameter Server
- Pull+Push



BACKGROUND

Mesh-based Synchronization

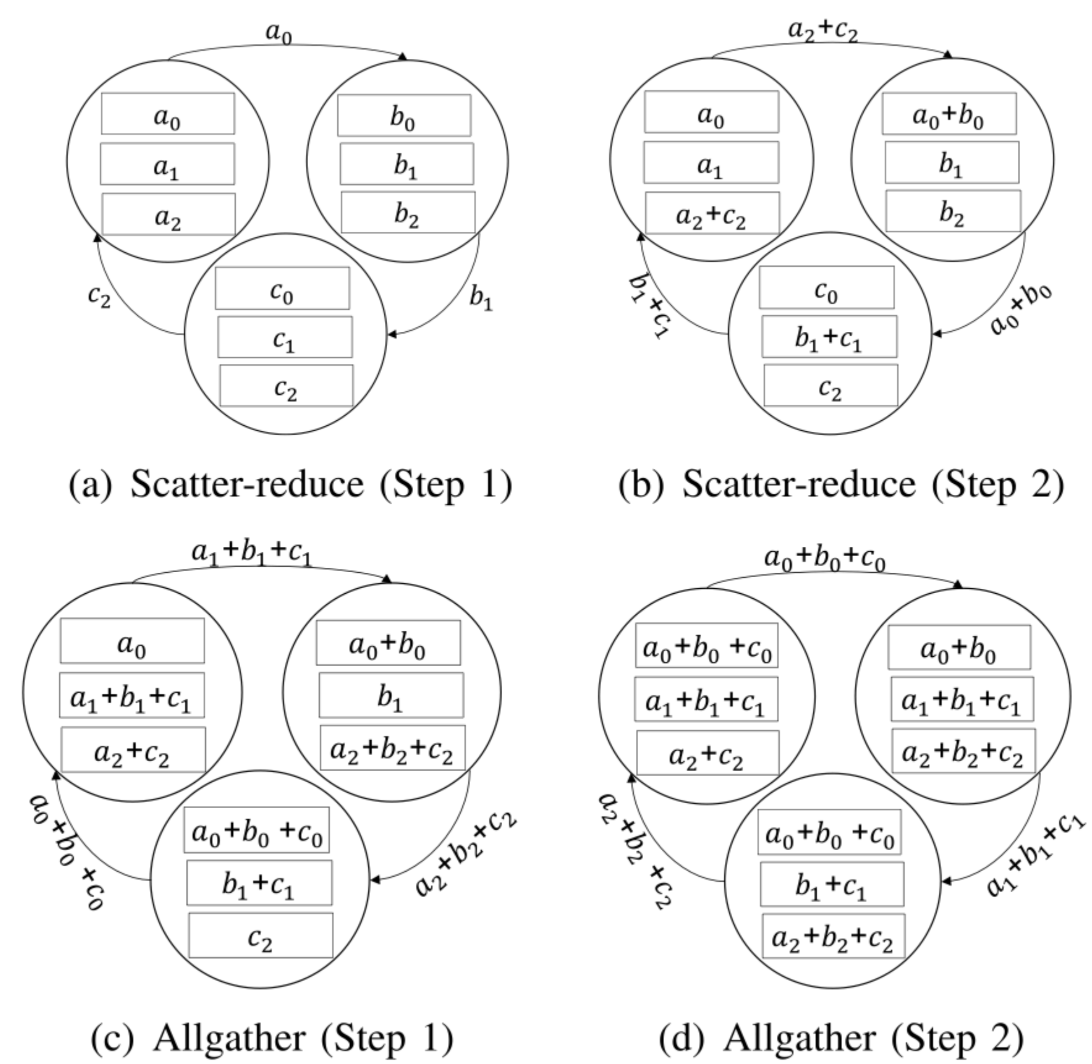
- 分散同步算法
- Server的功能是一致的
- Diffuse+Collect



BACKGROUND

Ring-based Synchronization

- 分散同步算法
- Scatter-reduce+Allgather
- N个Server，需要N-1次操作



(a) Scatter-reduce (Step 1)

(b) Scatter-reduce (Step 2)

(c) Allgather (Step 1)

(d) Allgather (Step 2)

BACKGROUND

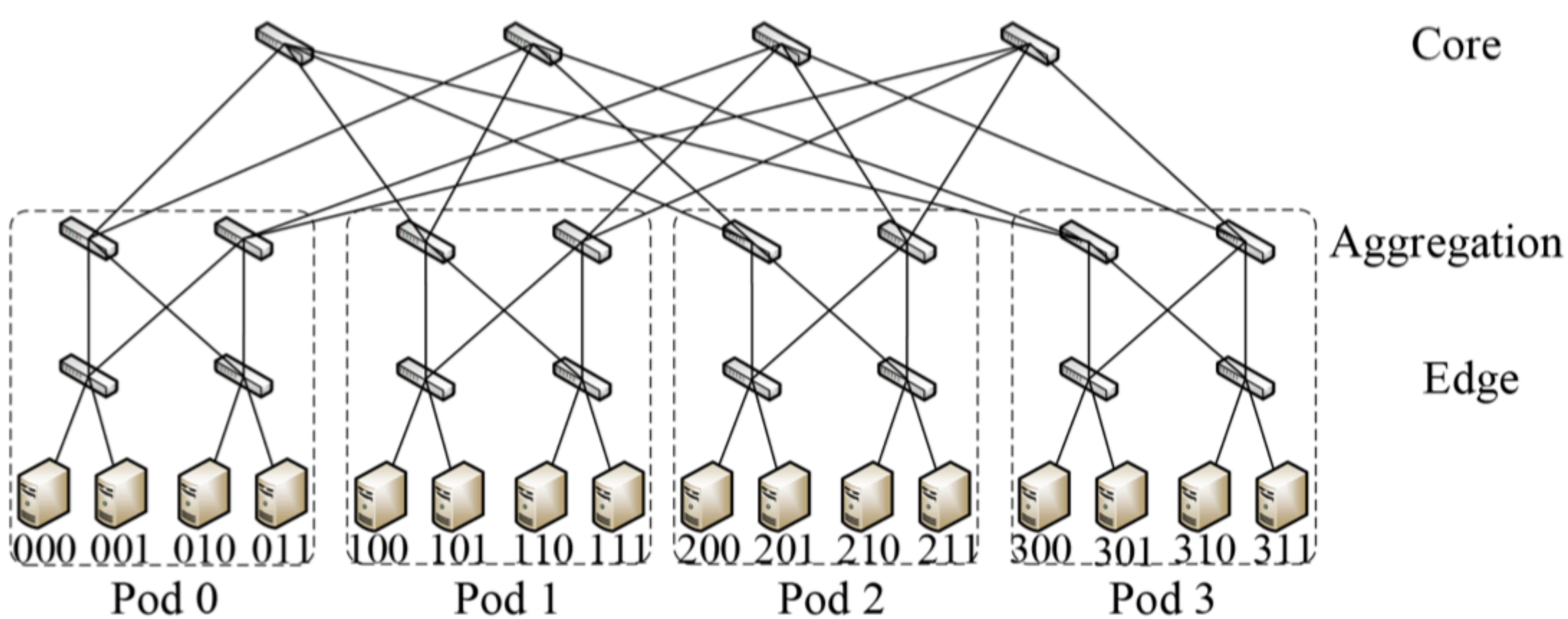
Physical Synchronization Topology

- Fat-Tree
- BCube
- Torus

BACKGROUND

Fat-Tree

- 以交换机为中心，分成了4层
- n端口Switch，$n^3/4$台Server
- 层与层之间的带宽是相同的

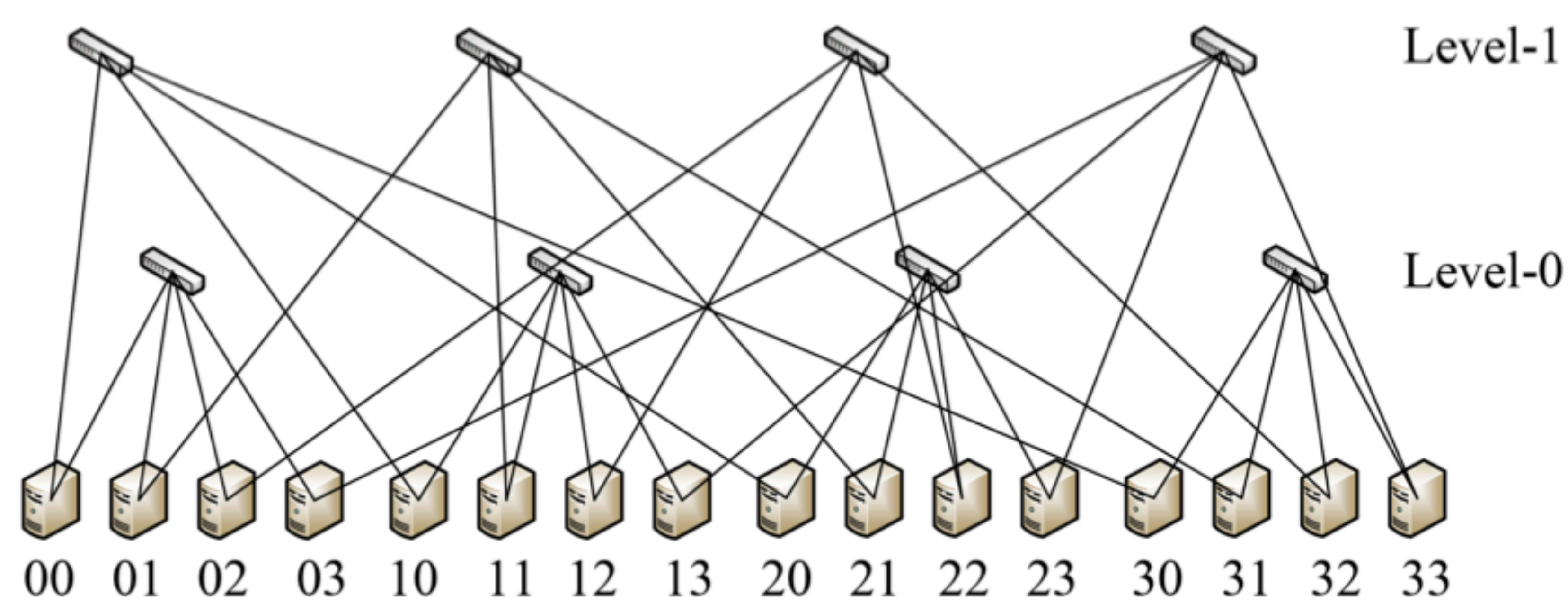


(a) Fat-Tree

BACKGROUND

BCube

- 以服务器为中心
- $BCube(n,k)$：分成k层，n^k台服务器，$k*n^{k-1}$台交换机
- 任意两台服务器之间有k条平行链路

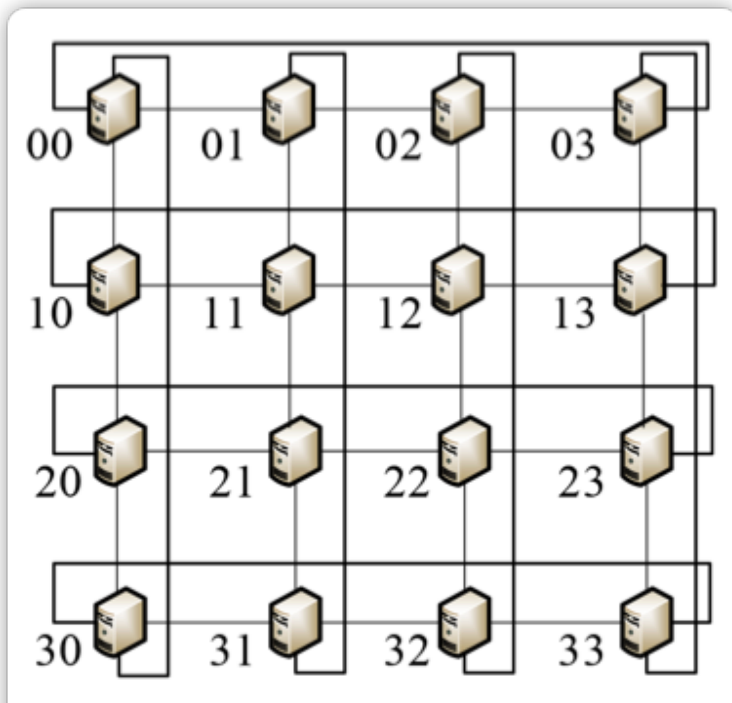


(b) BCube

BACKGROUND

Torus

- 以服务器为中心
- k-D Torus：可以互相连接



(c) 2-D Torus

BACKGROUND

Protocol

RDMA（Remote Direct Memory Access）

- 0拷贝：数据直接从网卡传输到应用层
- 内核旁路：在用户越过操作系统直接操作硬件
- DMA：直接内存访问，无需经过CPU

优点	缺点
延迟低	网络延迟要求高 网络的延迟会严重影响性能
高吞吐	

HiPS DESIGN

传统拓扑存在的问题

- MS拓扑中，同一链路的流量增大到一定规模时，会导致吞吐量的降低
- PS和MS的all-to-all传输模式可能会导致Pause Flow Control的传播
- RS中的one-to-one传输模式，通信的次数会与Server数量成正比

HiPS DESIGN

Algorithm 1 HiPS Algorithmic Framework

Input:
N: The total number of servers
n: The number of servers in a peer set
k: The number of hierarchical levels
p_array: The parameters on the server to synchronize by the process, i.e., $p\_array = [p_i^0, p_i^1, \ldots, p_i^{N-1}]$
$addr$: the address of a server, i.e., $[a_{k-1}, \ldots, a_i, \ldots, a_0]$

```
HiPS()
    allelic_array = p_array
    for i ← 0 to k − 1 do
        rank = addr[i]
        peer_array = GetPeers (i, addr)
        allelic_array =
        Sync1(rank, allelic_array, peer_array)
    end
    for i ← k − 1 to 0 do
        rank = addr[i]
        peer_array = GetPeers (i, addr)
        allelic_array =
        Sync2(rank, allelic_array, peer_array)
    end
    p_array = allelic_array
function GetPeers(level, addr)
    peer_array = []
    for i ← 0 to n − 1 do
        peer_addr = addr
        peer_addr[level] = i
        peer_array.append (peer_addr)
    end
    return peer_array
```

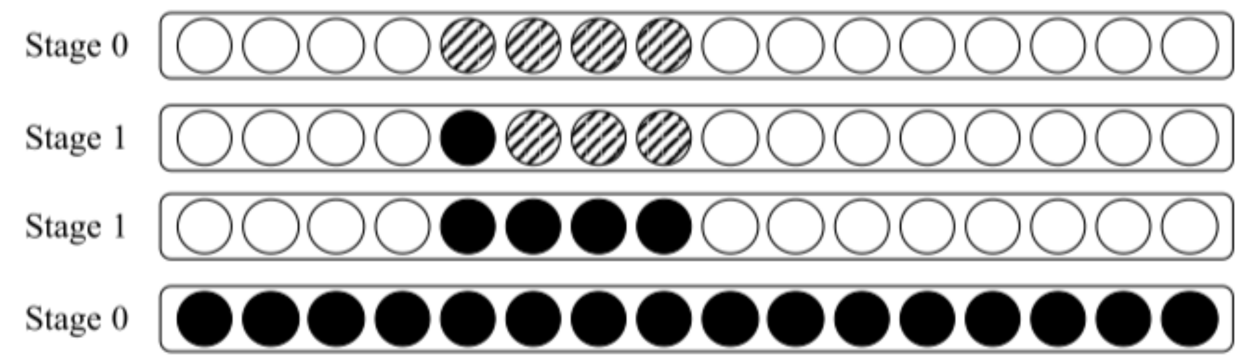
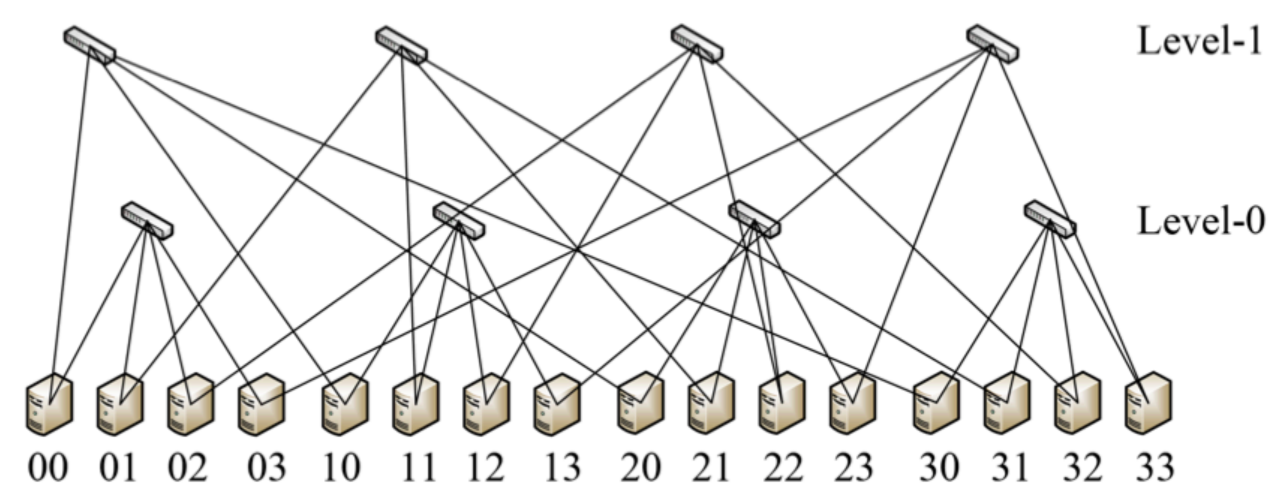


Fig. 5. Parameter states of server 10 in synchronization.



(b) BCube

ANALYSIS

Theoretical Analysis

Notation	Description
n	the number of ports of a switch in Fat-Tree or BCube, or the number of servers in a row (column) in Torus
k	the number of switch levels in BCube, or the number of dimensions in Torus
N	the total number of servers. For Fat-Tree, $N = n^3/4$. For BCube and Torus, $N = n^k$
α	latency, or startup time, of a communication
B	the bandwidth of the server NIC port or the switch port
P	the total size of all the gradients/parameters of a ML model

- Mesh-based
  - Fat-Tree：
    $$GST_{MF} = 2\left[\alpha + \frac{P/N}{B/(N-1)}\right] = \frac{2(N-1)P}{NB} + 2\alpha$$
  - BCube：
    $$GST_{MB} = \frac{2(\sqrt[k]{N}-1)P}{\sqrt[k]{N}B} + 2\alpha$$
  - Torus：
    $$GST_{MT} = \frac{(n+4)kP}{4B} + \frac{(n+4)k\alpha}{2}$$
- Ring-based

$$\begin{aligned} GST_{RF} &= 2(N-1)(\alpha + \frac{P/N}{B}) \\ &= \frac{2(N-1)P}{NB} + 2(N-1)\alpha \end{aligned}$$

$$GST_{RB} = \begin{cases} \frac{(N-1)P}{NB} + 2(N-1)\alpha, & k = 2 \text{ and } \\ & n \text{ is even} \\ \frac{2(N-1)P}{NB} + 2(N-1)\alpha, & k = 2 \text{ and } \\ & n \text{ is odd} \end{cases}$$

$$\begin{aligned} GST_{RT} &= 2(N-1)(\alpha + \frac{P/4/N}{B}) \\ &= \frac{(N-1)P}{2NB} + 2(N-1)\alpha \end{aligned}$$

ANALYSIS

Theoretical Analysis

Notation	Description
n	the number of ports of a switch in Fat-Tree or BCube, or the number of servers in a row (column) in Torus
k	the number of switch levels in BCube, or the number of dimensions in Torus
N	the total number of servers. For Fat-Tree, $N = n^3/4$. For BCube and Torus, $N = n^k$
α	latency, or startup time, of a communication
B	the bandwidth of the server NIC port or the switch port
P	the total size of all the gradients/parameters of a ML model

- HiPS
  - Fat-Tree：

$$\begin{aligned} GST_{HF} &= 2\left[3\alpha + \frac{(\frac{n}{2}-1)P}{\frac{n}{2}B} + \frac{(\frac{n}{2}-1)P}{(\frac{n}{2})^2B} + \frac{(n-1)P}{(\frac{n}{2})^2nB}\right] \\ &= \frac{2(N-1)P}{NB} + 6\alpha \end{aligned} \tag{8}$$

  - BCube：

$$GST_{HB} = 2\sum_{i=1}^{k}\left[\alpha + \frac{P/(kn^i)}{B/(n-1)}\right] = \frac{2(N-1)P}{kNB} + 2k\alpha$$

  - Torus：

$$GST_{HT} = \frac{(N-1)P}{kNB} + 2k(n-1)\alpha$$

ANALYSIS

Theoretical Analysis

Notation	Description
n	the number of ports of a switch in Fat-Tree or BCube, or the number of servers in a row (column) in Torus
k	the number of switch levels in BCube, or the number of dimensions in Torus
N	the total number of servers. For Fat-Tree, $N = n^3/4$. For BCube and Torus, $N = n^k$
α	latency, or startup time, of a communication
B	the bandwidth of the server NIC port or the switch port
P	the total size of all the gradients/parameters of a ML model

	Fat-Tree	BCube	Torus
MS	$\frac{2(N-1)P}{NB} + 2\alpha$	$\frac{2(\sqrt[k]{N}-1)P}{\sqrt[k]{N}B} + 2\alpha$	$\frac{(n+4)kP}{4B} + \frac{(n+4)k\alpha}{2}$, $n \bmod 4 = 0$
RS	$\frac{2(N-1)P}{NB} + 2(N-1)\alpha$	$\begin{cases} \frac{(N-1)P}{NB} + 2(N-1)\alpha, & k=2 \text{ and } n \text{ is even} \\ \frac{2(N-1)P}{NB} + 2(N-1)\alpha, & k=2 \text{ and } n \text{ is odd} \end{cases}$	$\frac{(N-1)P}{2NB} + 2(N-1)\alpha$, $k=2$
HiPS	$\frac{2(N-1)P}{NB} + 6\alpha$	$\frac{2(N-1)P}{kNB} + 2k\alpha$	$\frac{(N-1)P}{kNB} + 2k(n-1)\alpha$

- Fat-Tree中应用的所有逻辑拓扑都有相同的GST
- 相同的逻辑拓扑，$GST_{FatTree} > GST_{BCube} > GST_{Torus}$

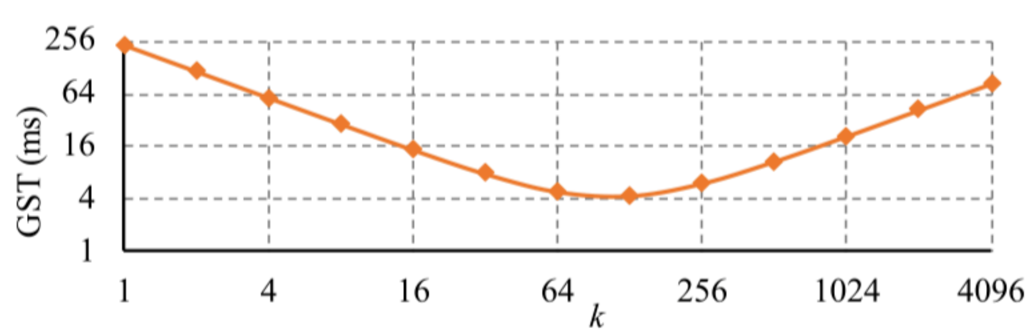
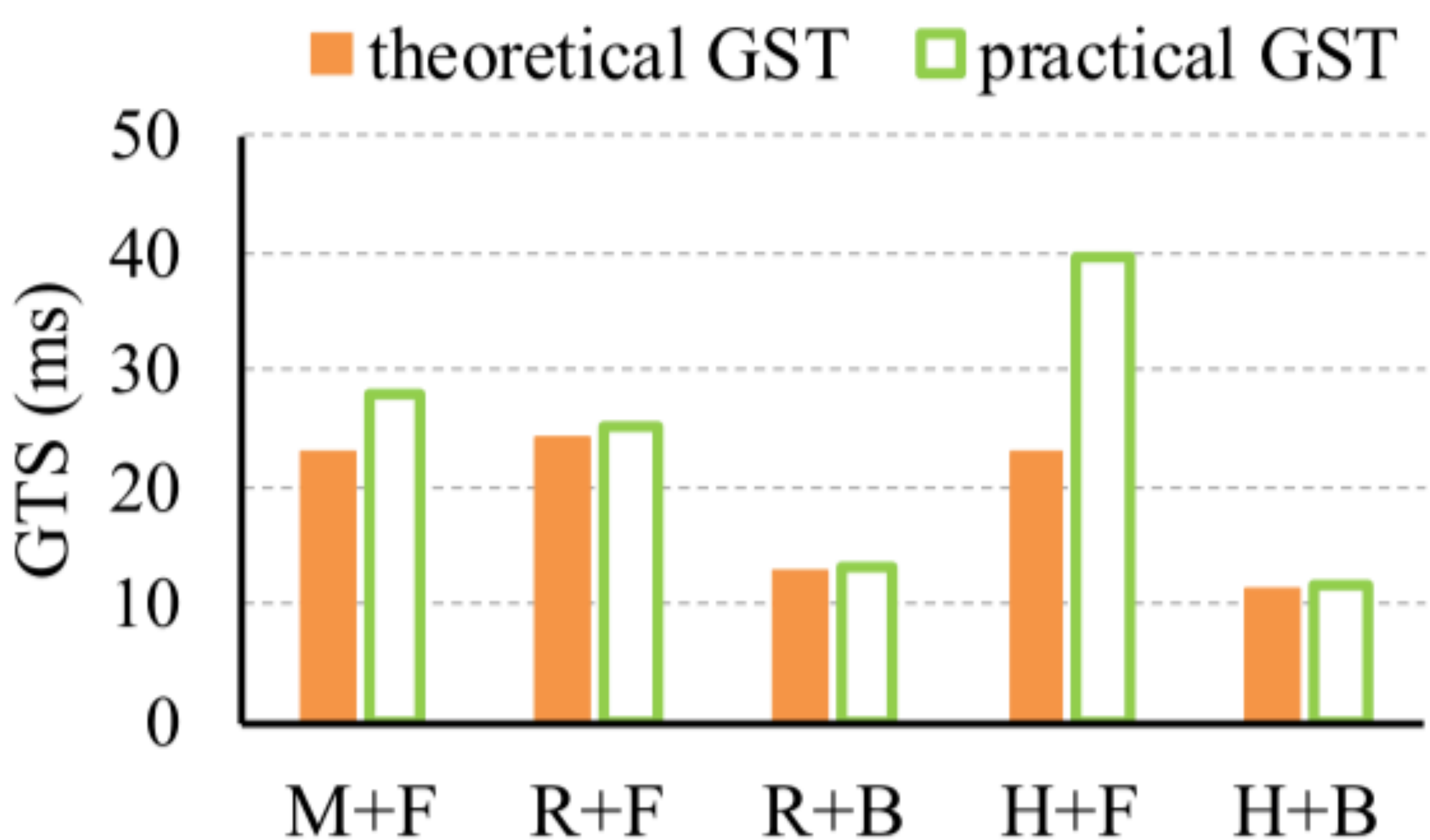


Fig. 6. Theoretical GSTs of HiPS+BCube with different k values, with $N = 128$, $P = 575\text{MB}$, $B = 40\text{Gbps}$, $\alpha = 10\mu\text{s}$.

EVALUATION

GST Comparison

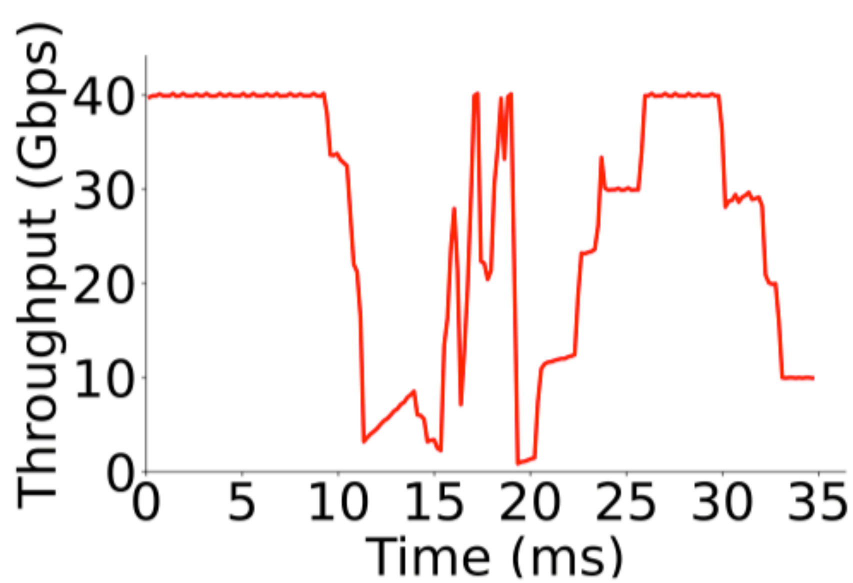
- 理论GST和实际GST对比



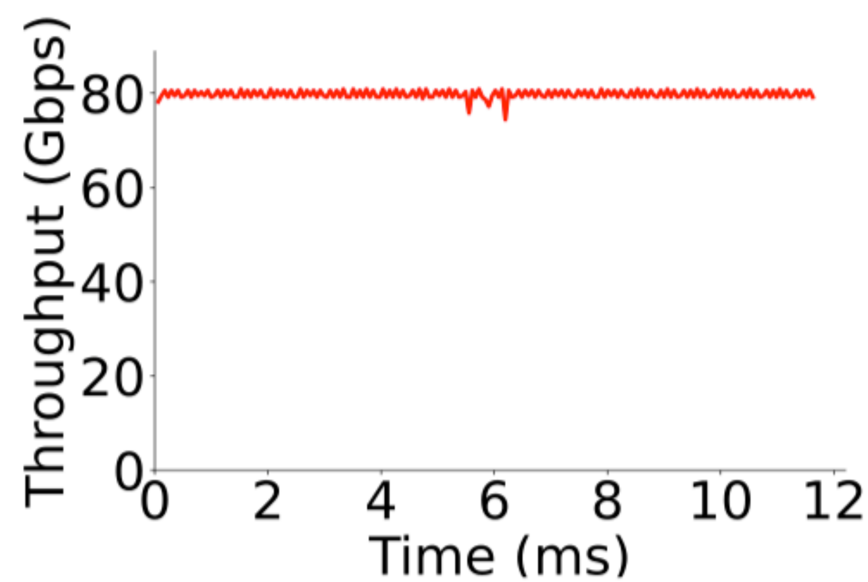
EVALUATION

GST Comparison

- 为了探究为什么理论GST和实际GST的差别，随机记录某一个Server的吞吐率。



(i) Total throughput of all flows received by one server in HiPS+Fat-Tree.

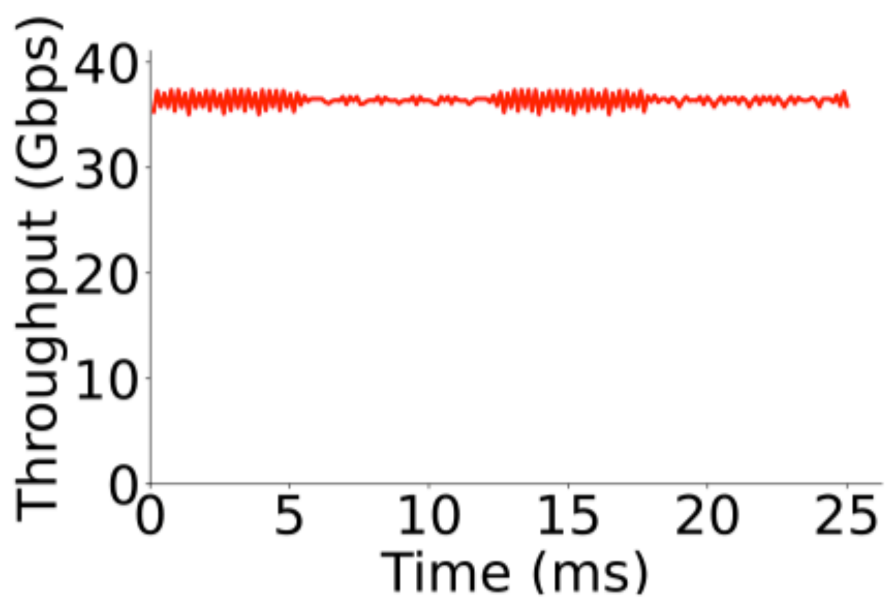


(j) Total throughput of all flows received by one server in HiPS+BCube.

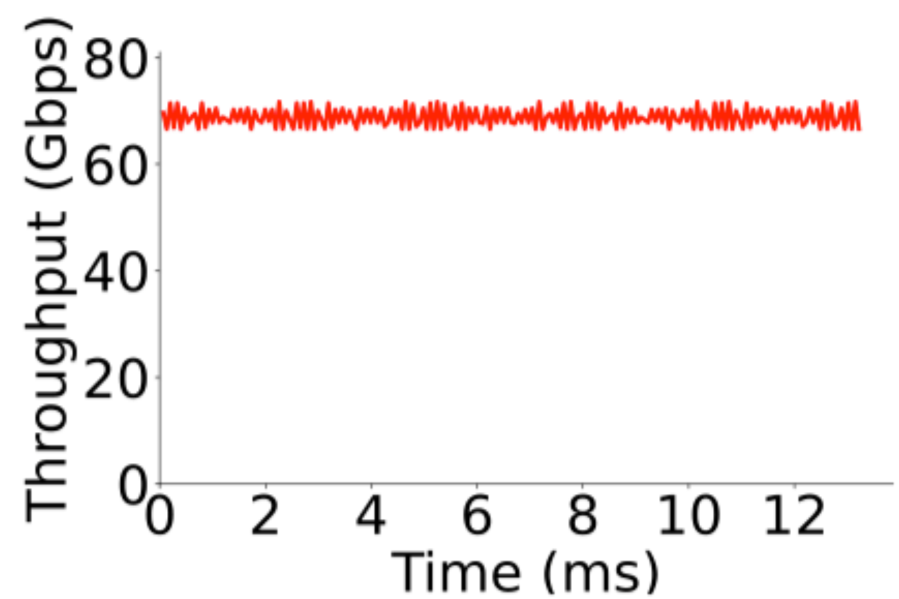
EVALUATION

GST Comparison

- 为了探究为什么理论GST和实际GST的差别，随机记录某一个Server的吞吐率。



(g) Total throughput of all flows received by one server in RS+Fat-Tree.



(h) Total throughput of all flows received by one server in RS+BCube.

EVALUATION

PFC Analysis

- 通过收集PFC pause frame，研究Fat-Tree和BCube对RDMA的支持。
- Mesh+Fat-Tree和HiPS+Fat-Tree均会产生PFC pause frame

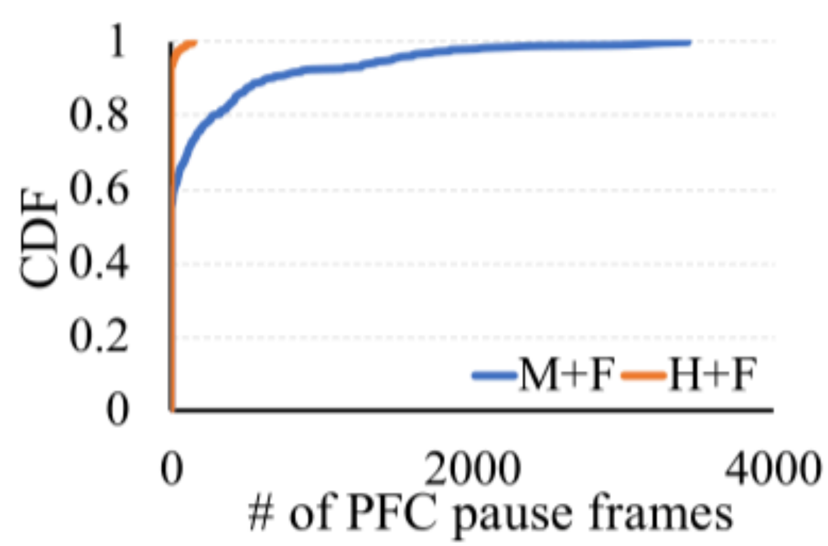
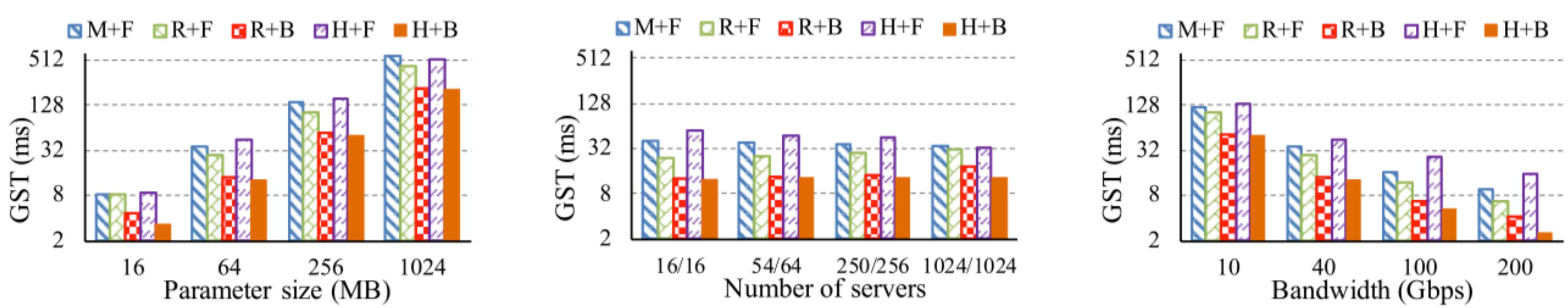


Fig. 10. CDF of the number of PFC pause frames received by different nodes.

EVALUATION

Scalability Analysis

- 随着Parameter size的增大，HiPS+BCube的GST是线性增长的，相比较于RS+BCube，HiPS能够在Parameter size比较小时也能达到很好的效果。
- 当增加Server数量时，HiPS+BCube的GST只增加了6.5%
- 带宽增加时，HiPS+BCube的GST的下降的最快。



(a) 250/256 servers and 40Gbps bandwidth.

(b) 64MB parameters and 40Gbps bandwidth.

(c) 64MB parameters and 250/256 servers.

CONCLUSION

- 物理拓扑对DML的影响：
  - Fat-Tree中Server只有一张网卡，无法支持同步算法的并行
  - ECMP会导致链路的流量负载不均衡，导致拥塞
  - BCube的拓扑比Fat-Tree更适合应用于RDMA协议
- 逻辑拓扑对DML的影响：
  - 大范围的参数同步，可能会造成链路的传输能力不够，导致交换机的吞吐量下降
  - 因此采用分层的参数同步算法，能够降低链路的要求，增加同步效率