分布式机器学习网络

报告人：王帅

清华大学

2021年8月21日

报告提纲

一、研究背景与相关工作

二、层次化参数同步算法

三、物理网络互联拓扑优化

四、流优先级调度全网优化

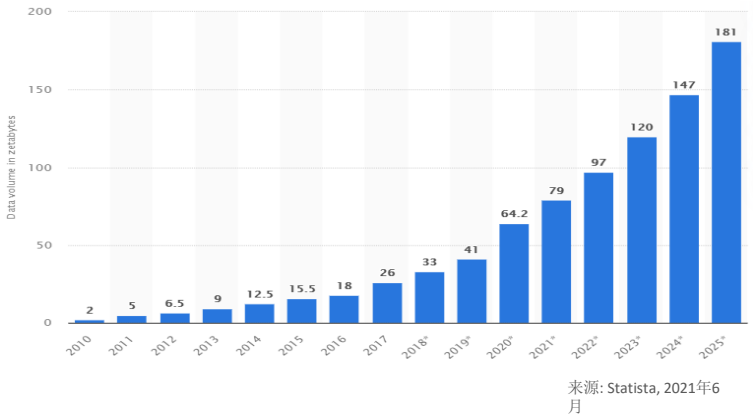
大规模机器学习任务的算力需求

海量数据

深度模型

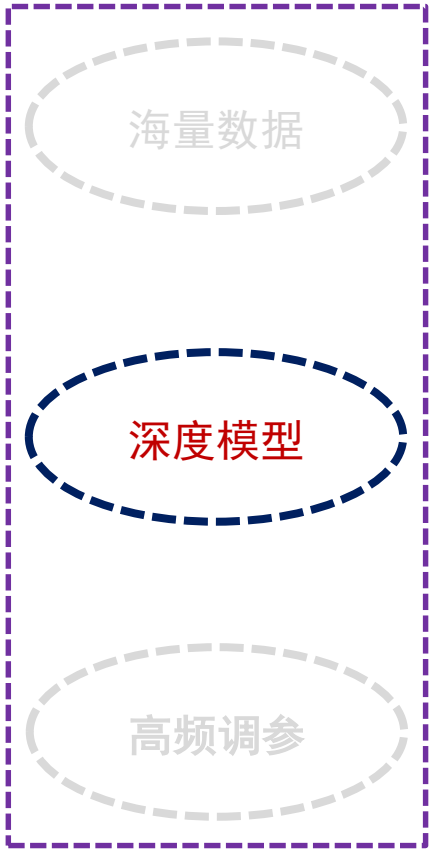
高频调参

- 互联网时代的海量训练数据是突破人工智能瓶颈的重要原因
  - ✓ ImageNet：千万张图片
  - ✓ 推荐系统：千万级条目
  - ✓ 语言模型：TB级词库



来源: Statista, 2021年6月

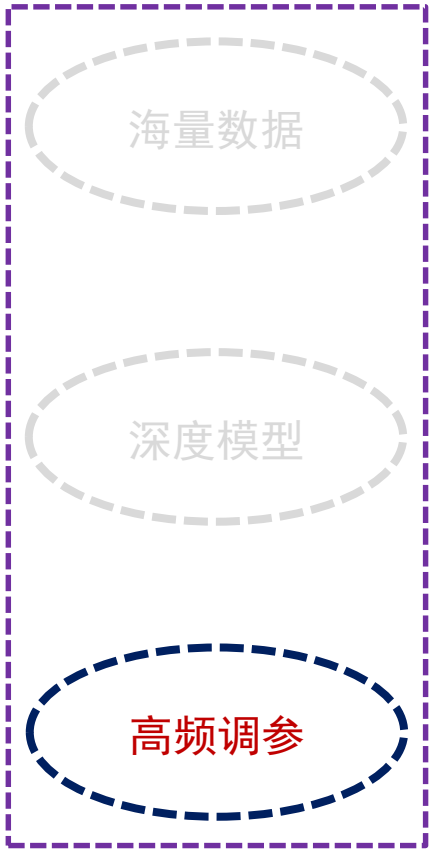
大规模机器学习任务的算力需求



- 深度学习的模型规模越来越大
  - ✓ 视觉模型 - ResNet，百万级参数
  - ✓ 语言模型 - Bert，亿级参数
  - ✓ 多模态模型 - 悟道2.0，万亿级参数

模型	发布时间	参数量	预训练数据量
GPT	2018 年 6 月	1.17 亿	约 5GB
GPT-2	2019 年 2 月	15 亿	40GB
GPT-3	2020 年 5 月	1750 亿	45TB

大规模机器学习任务的算力需求



- 机器学习算法的效果高度依赖于超参数的设置，需要不断进行尝试、调优
  - Learning rate
  - Batch size
  - SSP步长

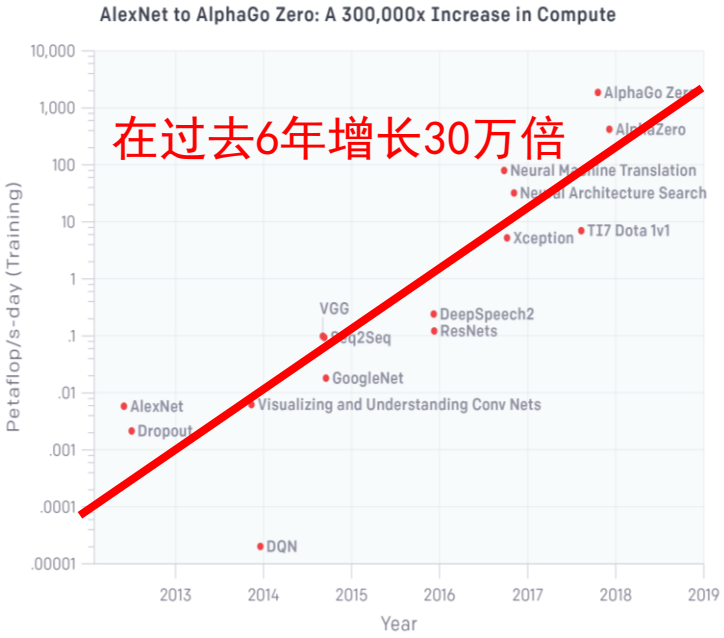
Resnet	Style transfer	Tacotron	BiDAF
Inceptionv1	YOLO	WaveRNN	GPT
ELMo	Densenet	**Open AI GPT**	VoxelNet
Mobilenet	FCN	Inceptionv3	**Mask R-CNN**
Nasnet	**BERT**	VGGish	DeepConvLSTM
VGG	MT-DNN	Xception	Resnet50
PSPNet	Inceptionv2	Transformer	Inceptionv4
ULMFit	SSD Lite	LAS	**Deep Speech**
SSD mobilenet	Deeplabv3	Seq2seq	GPT2
Squeezenet	WaveGlow	RNN-T	**Wavenet**

source: WWDC 2019

分布式机器学习

分布式机器学习成为大幅提升算力的重要途径

- AI算力需求在过去6年增长30万倍，平均每三个半月翻一番
- 单节点无法满足AI算力要求，分布式AI成为AI计算的常态



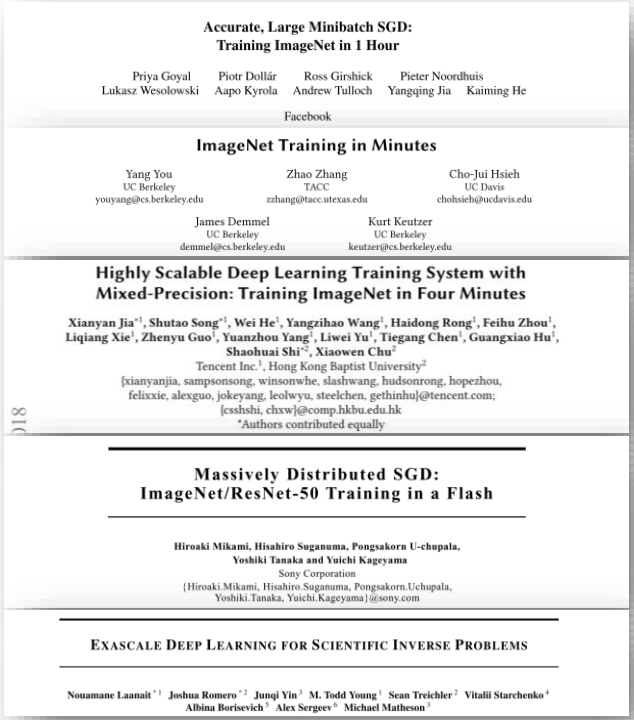
source: openai.com

模型	单 P100 GPU训练时间 (小时)
Transformer	31
Mini Go	73
Mask-R-CNN	83
ResNet-50 v1.5	147

source: mlperf.org

业界公开的分布式机器学习系统

- ImageNet训练
  - 2017.06, Facebook: 1 小时
    - 256 块 P100 GPU
  - 2017.09, UC Berkeley: 24 分钟
    - 512 个 Intel KNL处理器
  - 2018.07, Tencent: 6.6 分钟
    - 2048 块 P40 GPU
  - 2018.10, Sony: 224 秒钟
    - 2176 块 V100 GPU
  - 2019.4, Fujitsu: 72秒
    - 2048块 V100 GPU
  - 2019.8, Huawei: 59.8秒
    - 2014块 Ascend （昇腾）910处理器
- 科学逆问题求解
  - 2019.09, ORNL
    - Summit超级计算机(4,600 节点，27,600 块 V100 GPU)
- 预训练模型
  - 2021.05,智源研究院
    - 神威太湖之光超级计算机



Accurate, Large Minibatch SGD:
Training ImageNet in 1 Hour

Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook

ImageNet Training in Minutes

Yang You
UC Berkeley
youyang@cs.berkeley.edu

Zhao Zhang
TACC
zzhang@tacc.utexas.edu

Cho-Jui Hsieh
UC Davis
chohsieh@ucdavis.edu

James Demmel
UC Berkeley
demmel@cs.berkeley.edu

Kurt Keutzer
UC Berkeley
keutzer@cs.berkeley.edu

Highly Scalable Deep Learning Training System with
Mixed-Precision: Training ImageNet in Four Minutes

Xianyan Jia*[1], Shutao Song*[1], Wei He[1], Yangzihao Wang[1], Haidong Rong[1], Feihu Zhou[1],
Liqiang Xie[1], Zhenyu Guo[1], Yuanzhou Yang[1], Liwei Yu[1], Tiegang Chen[1], Guangxiao Hu[1],
Shaohuai Shi*[2], Xiaowen Chu[2]
Tencent Inc.[1], Hong Kong Baptist University[2]
{xianyanjia, sampsonsong, winsonwhe, slashwang, hudsonrong, hopezhou,
felixxie, alexguo, jokeyang, leolwyu, steelchen, gethinhu}@tencent.com;
{csshshi, chxw}@comp.hkbu.edu.hk
*Authors contributed equally

018

Massively Distributed SGD:
ImageNet/ResNet-50 Training in a Flash

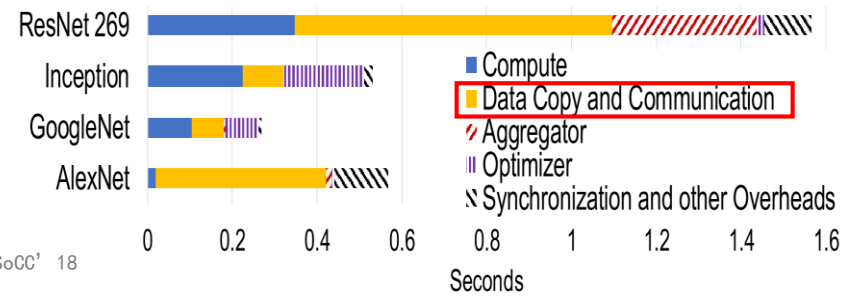
Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala,
Yoshiki Tanaka and Yuichi Kageyama
Sony Corporation
{Hiroaki.Mikami, Hisahiro.Suganuma, Pongsakorn.Uchupala,
Yoshiki.Tanaka, Yuichi.Kageyama}@sony.com

EXASCALE DEEP LEARNING FOR SCIENTIFIC INVERSE PROBLEMS

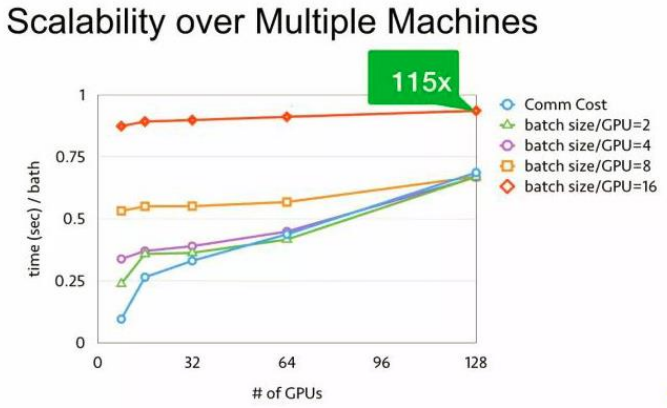
Nouamane Laanait*[1] Joshua Romero*[2] Junqi Yin[3] M. Todd Young[1] Sean Treichler[2] Vitalii Starchenko[4]
Albina Borisevich[5] Alex Sergeev[6] Michael Matheson[3]

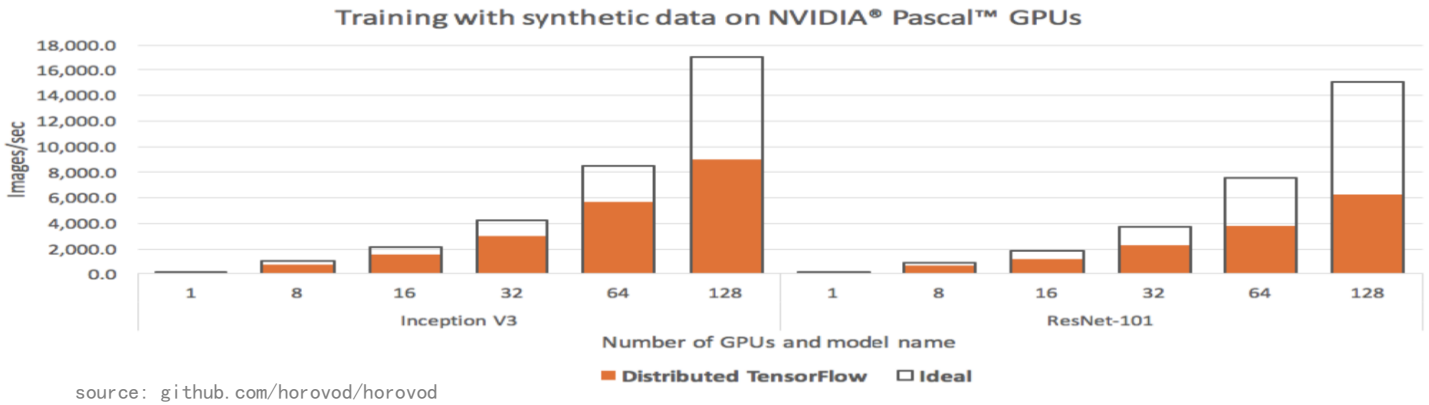
分布式机器学习系统的网络通信开销

- 数据并行方式下，节点间需频繁同步模型参数
- 随着分布式机器学习系统规模的不断扩大，网络通信开销逐渐成为限制系统性能的一大瓶颈



source: SoCC' 18





source: github.com/horovod/horovod

分布式机器学习系统的网络通信开销

$$t=(T_M/n+ t_U)*R$$

# of iterations till convergence: R

Time for training the mini-batch M in a single server: T_M

# of workers in DML: n

Time for parameter update in each iteration: t_U

- n越大（分布式机器学习的规模越大）
  - 计算时间越短
  - 网络通信开销的比重越大
  - 悖论
    - 网络开销大->设置较大的T_M/n
    - 较大的T_M ->影响机器学习模型的训练准确度

分布式机器学习的网络优化方法

- 并行同步策略优化
  - 模型并行，数据并行，ASP, SSP, BSP
  - Stanza [arXiv 2018]：将卷积层和全连接层在不同组计算节点处理
  - GPipe [arXiv 2018]：进一步划分“微批量”，迭代内流水线并行
  - PipeDream [SOSP 2019]: 迭代间1F1B流水线并行，避免GPipe下的气泡问题
  - FlexFlow [SysML 2019]: 通过模拟器与搜索自行选择最优的策略
  - FELA [ICDE 2020]: 采用灵活并行度和基于token机制的弹性工作负载调度
  - DAPPLE [PPoPP 2021]: 结合“微批量”和1F1B流水并行，压缩迭代内气泡
- 参数同步路径优化
  - HiPS [SIGCOMM 2018 WKSHP]: 层次化参数同步
  - BML [NeurIPS 2018]: 软硬件co-design的参数同步算法
  - FlexTree : 基于更精确的计算/通信开销模型对参数同步的逻辑拓扑调优

分布式机器学习的网络优化方法

- 计算通信重叠优化
  - TicTac [SysML 2019]：发送端使用多线程严格控制参数发送顺序
  - P3 [SysML 2019]：使用切片和优先级发送队列，在发送端实现抢占式调度
  - ByteScheduler [SOSP 2019]: 通过自己的通讯库接管整个调度流程，消除global barrier
  - Geryon [INFOCOM 2020]: 基于流优先级机制对参数传输顺序进行全网规模调度，增大计算和通信的重叠
  - CEFS [CoNEXT 2020]: 针对异构训练集群，结合参数优先级和工作者优先级，采用流优先级调度，最大化优先级调度的性能收益
- 网络传输性能优化
  - RDMA / GDR : 远程直接数据存取，从根源上提高网络的效能
  - Horovod [arXiv 2018]: Ring Allreduce基础上通过Tensor Fusion统一处理小张量
  - MG-WFBP [INFOCOM 2019] :优化Tensor Fusion的粒度
  - Poseidon [ATC 2017]: 无需等待的反向传播和混合通信方式

分布式机器学习的网络优化方法

- 物理网络拓扑优化
  - Fat-Tree [SIGCOMM 2008]
  - Bcube [SIGCOMM 2009]
  - Torus [Kluwer Academic 2002]
- 网络辅助计算优化
  - DAIET [HotNets 2017]: 在交换机上进行参数聚合
  - SwitchML [NSDI 2021]: 将参数服务器功能卸载到可编程交换机
  - ATP [NSDI 2021]: 可编程交换机对流经的参数执行“尽力而为”的汇总

团队发表论文情况

- Songtao Wang, Dan Li, Yang Cheng, et. al, "A Scalable, High-performance and Fault-tolerant Network Architecture for Distributed Machine Learning", IEEE/ACM Transactions on Networking, 2020.
- Yang Cheng, Dan Li, Zhiyuan Guo, et. Al, "Accelerating End-to-End Deep Learning Workflows with Codesign of Data Preprocessing and Scheduling", IEEE TPDS-SS-AI, 2020.
- Shuai Wang, Dan Li, Jiansong Zhang, et. al, “CEFS: Compute-Efficient Flow Scheduling for Iterative Synchronous Applications”, CoNext 2020, Barcelona, Spain.
- Shuai Wang, Dan Li and Jinkun Geng, “Geryon: Accelerating Distributed CNN Training by Network-Level Flow Scheduling”, INFOCOM 2020, Beijing, China.
- Jinkun Geng, Dan Li and Shuai Wang, “FELA: Incorporating Flexible Parallelism and Elastic Tuning to Accelerate Large-Scale DML”, ICDE 2020, Dallas, Texas.
- Shuai Wang, Dan Li, Jinkun Geng, et. al, "Impact of Network Topology on the Performance of DML: Theoretical Analysis and Practical Factors", INFOCOM 2019, Paris, France.
- Jinkun Geng, Dan Li and Shuai Wang, "Rima: An RDMA-Accelerated Model-Parallelized Solution to Large-Scale Matrix Factorization", ICDE 2019, Macao, China.
- Yang Cheng, Dan Li, Zhiyuan Guo, et. al. "DLBooster: Boosting End-to-End Deep Learning Workflows with Offloading Data Preprocessing Pipelines", ICPP 2019, Kyoto, Japan.
- SongtaoWang, Dan Li, Yang Cheng, et. al, "BML: A High-performance, Low-cost Gradient Synchronization Algorithm for DML Training", NeurIPS 2018, Montreal, Canada.
- Jinkun Geng, Dan Li, Yang Cheng, et. al, "HiPS: Hierarchical Parameter Synchronization in Large-Scale Distributed Machine Learning", SIGCOMM WKSHPS, Budapest, Hungary, 2018.
- Jinkun Geng, Dan Li and Shuai Wang, “Accelerating Distributed Machine Learning by Smart Parameter Serve”, APNet 2019, Beijing, China
- Jinkun Geng, Dan Li and Shuai Wang, “ElasticPipe: An Effcient and Dynamic Model-Parallel Solution to DNN Training”, ScienceCloud 2019, Phoenix, AZ, USA
- Jinkun Geng, Dan Li and Shuai Wang, “Horizontal or Vertical? A Hybrid Approach to Large-Scale Distributed Machine Learning”, CCIW 2019, Phoenix, AZ, USA

报告提纲

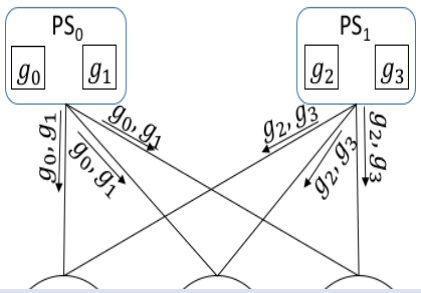
一、研究背景与相关工作

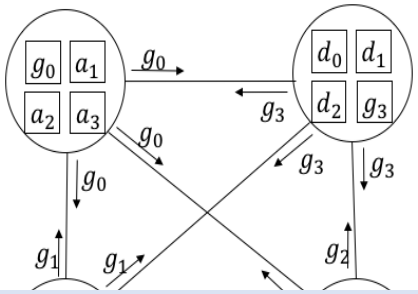
二、层次化参数同步算法

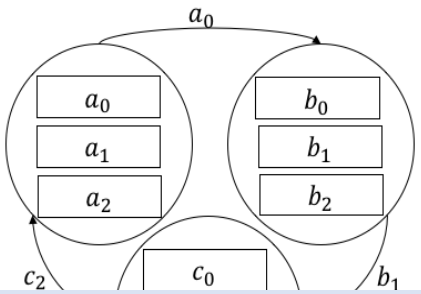
三、物理网络互联拓扑优化

四、流优先级调度全网优化

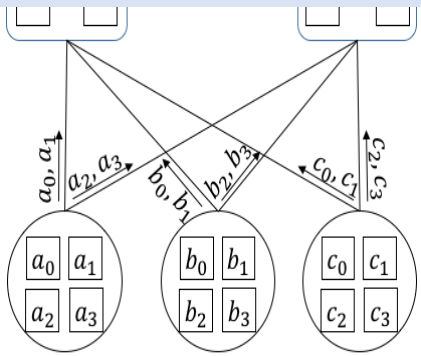
现有参数同步算法

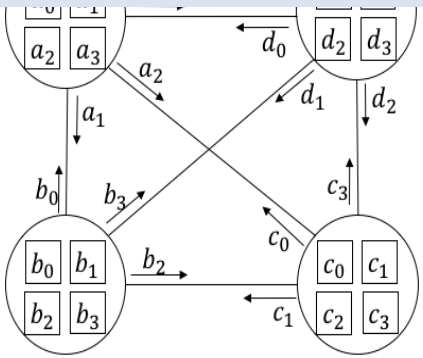


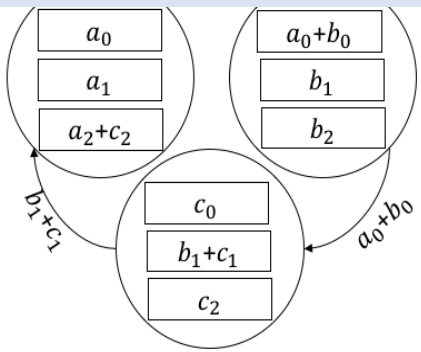




现有参数同步算法是扁平化的、拓扑无感知的，
而物理网络拓扑往往具有“层次化”的结构特点







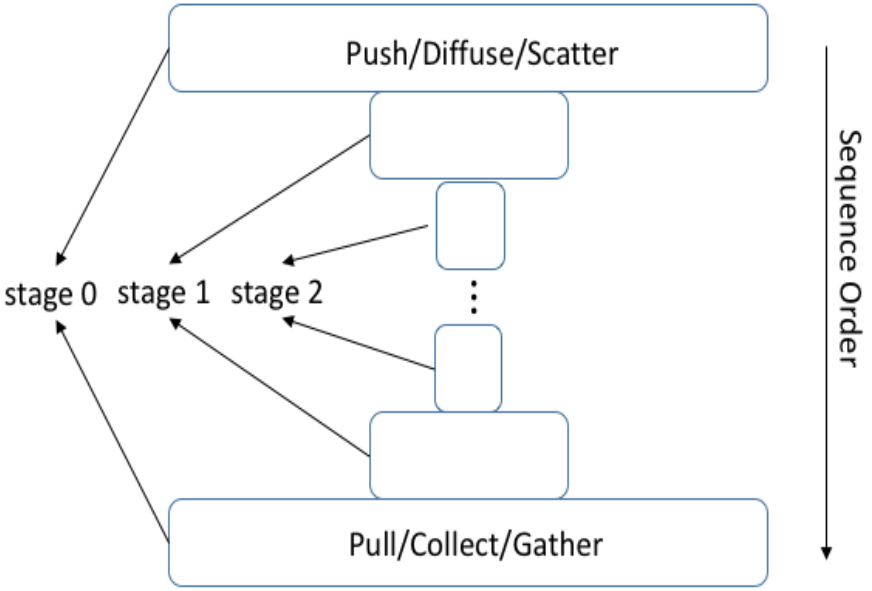
PS-based
(Pull + Push)

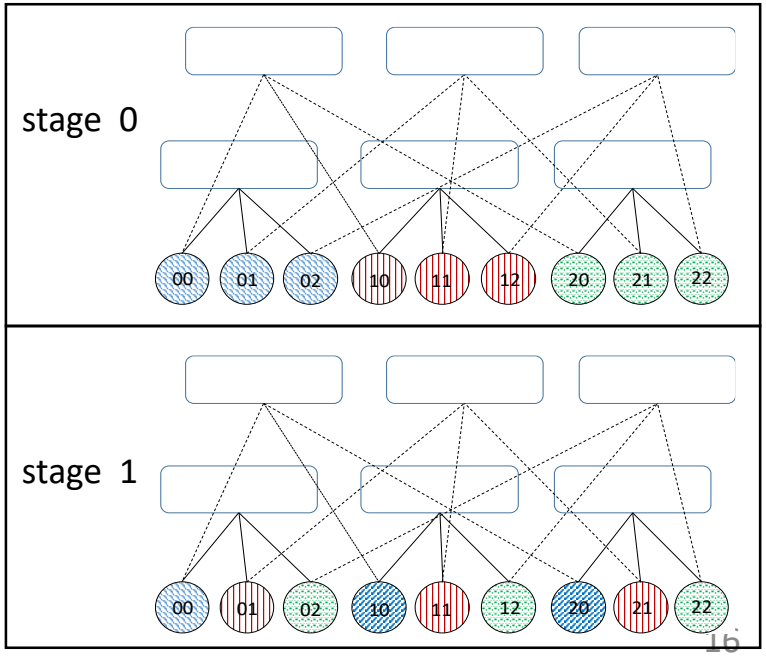
Mesh-based
(Diffuse + Collect)

Ring-based
(Scatter + Gather)

层次化参数同步算法HiPS

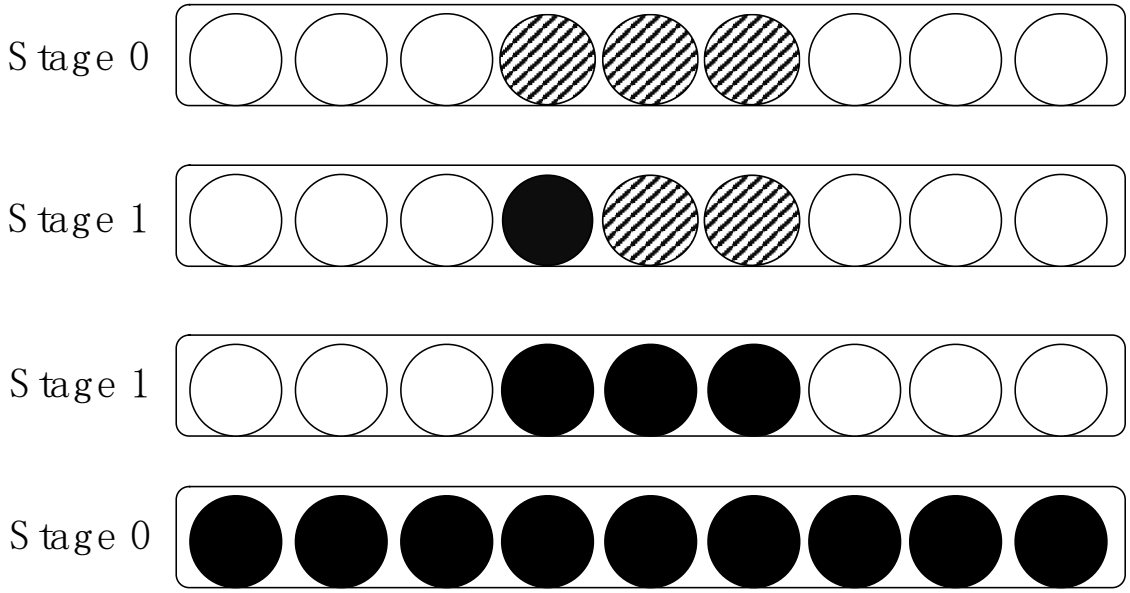
- HiPS结合底层网络拓扑的层次化特点，提出分层参数同步的方法
  - 分层同步、层内聚合，减少通信数据量
  - 层内支持多种扁平化同步算法





层次化参数同步算法HiPS

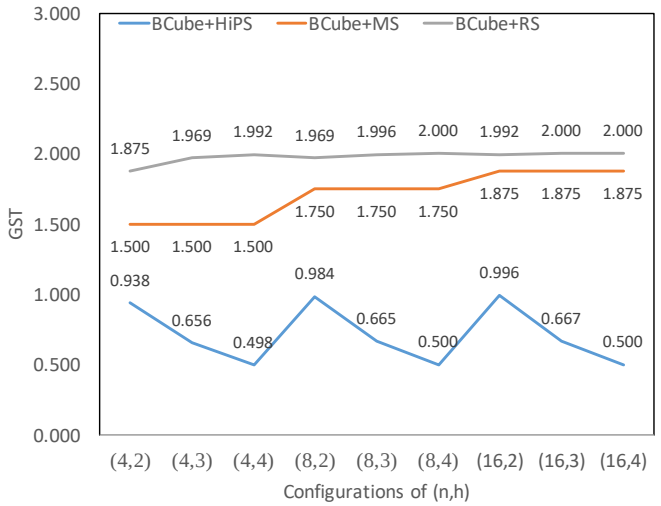
- Server <01>的参数同步过程：
  - stage 0同步时，同步所有梯度
  - 当stage 0结束时，其上有3个部分梯度和
  - stage 1同步时，仅同步部分梯度和
  - 当stage 1结束时，其上有1个完整梯度和

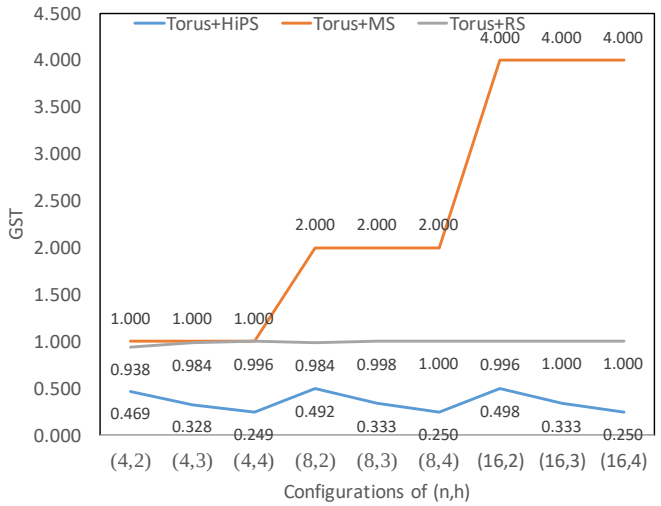


层次化参数同步算法HiPS

- 理论全局参数同步时间GST
  - 相比扁平化参数同步算法，HiPS的参数同步时间可降低h倍

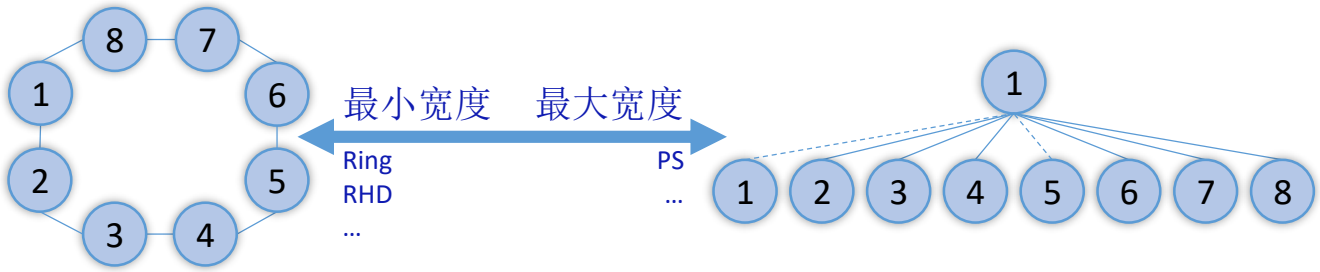
	BCube	Torus
HiPS	$2\frac{n^h-1}{n^h}\frac{W}{hB}$	$\frac{n^h-1}{n^h}\frac{W}{hB}$
PSS/MS	$2\frac{n-1}{n}\frac{W}{B}$	$\begin{cases}\frac{(n^2-1)}{4n}\frac{W}{B}, & n \text{ is odd}\\ \frac{n}{4}\frac{W}{B}, & n \text{ is even}\end{cases}$
RS	$2\frac{n^h-1}{n^h}\frac{W}{B}$	$\frac{n^h-1}{n^h}\frac{W}{B}$





灵活参数同步算法FlexTree

- 现有同步算法宽度不够灵活，存在如下问题：



✓ 一打一，带宽专享
x 内存读写次数多
x 串行启动

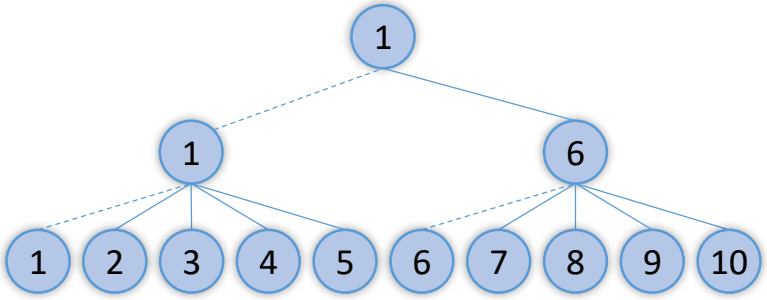
✓ 内存读写次数少
x 并行启动
x 多打一，带宽竞争

- 结合Full-Mesh和树状拓扑特点，提出宽度灵活的参数同步方法FlexTree。
  - 更灵活：宽度灵活，兼顾计算开销和通信开销
  - 更高效：利用Cost Model选择更优宽度

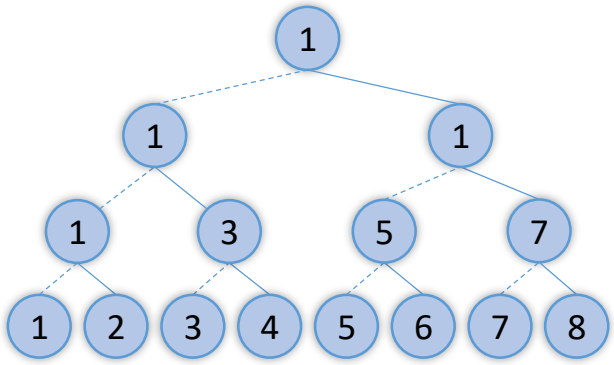
FlexTree

兼顾多种节点数：针对不同节点数完成不同宽度树的建立。例：

- 实验常用节点数：8：一叉树（Ring Allreduce）、二叉树（Tree Allreduce第一层树宽为2，第二层树宽为4）、四叉树（Tree Allreduce第一层树宽为4，第二层树宽为2）、八叉树（Parameter Server）
- 奇数节点数：9：一叉树（Ring Allreduce）、三叉树（Tree Allreduce树宽为3）、九叉树（Parameter Server）
- 偶数节点数：10：一叉树（Ring Allreduce）、二叉树（Tree Allreduce第一层树宽为2，第二层树宽为5）、五叉树（Tree Allreduce第一层树宽为5，第二层树宽为2）、十叉数（Parameter Server）



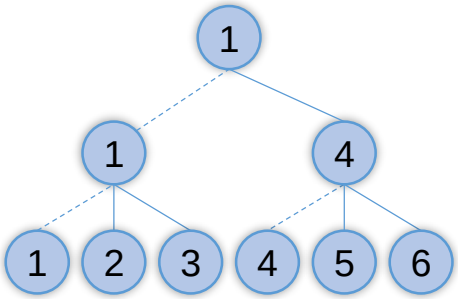
一层宽度为2、第二层宽度为5



各层宽度均为2

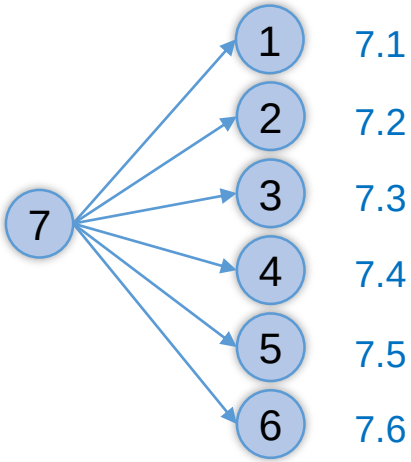
FlexTree

- 非常见节点数，如$N = n + 1$：
  - reduce scatter
    - n个节点内部使用任意一种FlexTree
    - 与此同时，+1节点将数据分为n份，分别发给前n个节点。
  - Allgather
    - 待前n节点汇总完成后，反向执行reduce scatter过程



进行任意一种FlexTree

与此同时

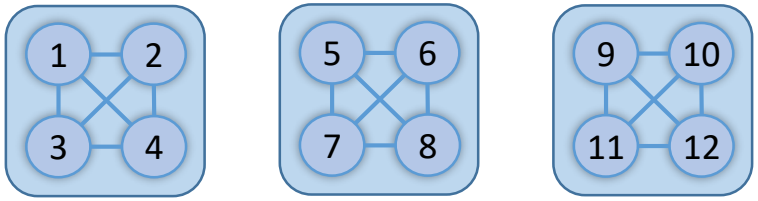


发送数据给前若干节点

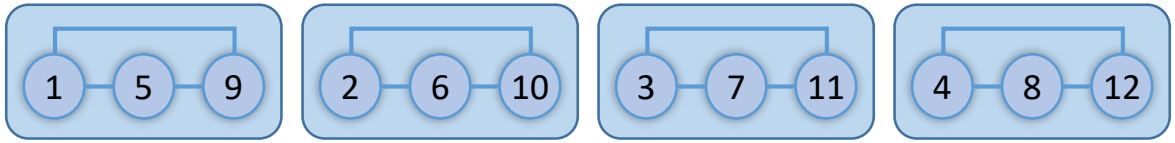
FlexTree

同步步骤：以12个节点构成下层宽度4、上层宽度3的树为例

1.每四个节点为一组，组内进行full mesh通信，随后聚合。



2.每三个节点为一组，组内进行full mesh通信，随后聚合。
此时已经完成Gather过程。



3.沿用此前的三节点组，组内进行full mesh通信。

4.沿用此前的四节点组，组内进行full mesh通信。同步完成。

FlexTree

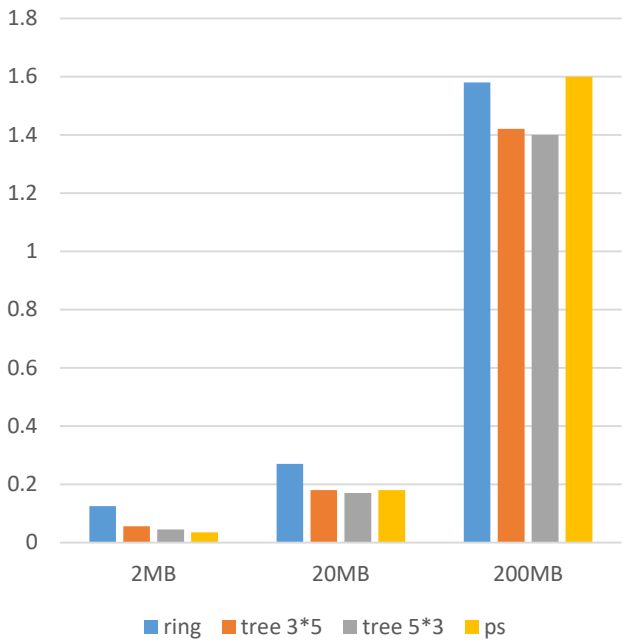
- Cost Model
  - α：延迟系数，与通信步骤数共同作用。
  - β：带宽系数，与数据量、节点数共同作用。
  - γ：计算开销系数，与数据量、节点数共同作用。
  - δ：内存读写系数，与数据量、节点数共同作用。
  - ε：多对一阻塞系数，与宽度、数据量共同作用。

$$cost = 2\boldsymbol{\alpha} + \frac{2(N-1)S}{N}\boldsymbol{\beta} + \frac{(N-1)S}{N}\boldsymbol{\gamma} + \frac{(N+1)S}{N}\boldsymbol{\delta} + Sw\boldsymbol{\varepsilon}$$

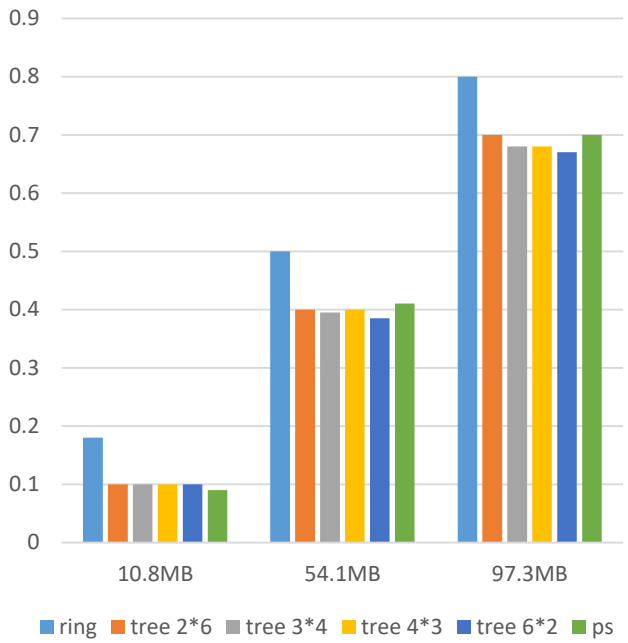
N:节点数；S:总数据量；w:树宽。这是一个PS开销计算的样例。

FlexTree

实验测试：不同树宽下全局参数同步时间对比。FlexTree在部分条件下的性能优于ring和ps。



15节点



12节点

横轴：单节点上的数据量；纵轴：同步时间（秒，越低越好）

报告提纲

一、研究背景与相关工作

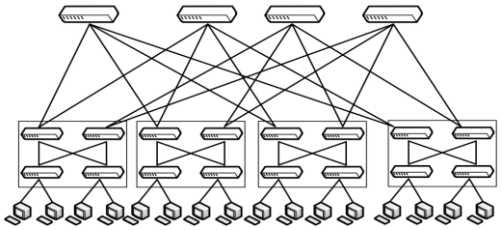
二、层次化参数同步算法

三、物理网络互联拓扑优化

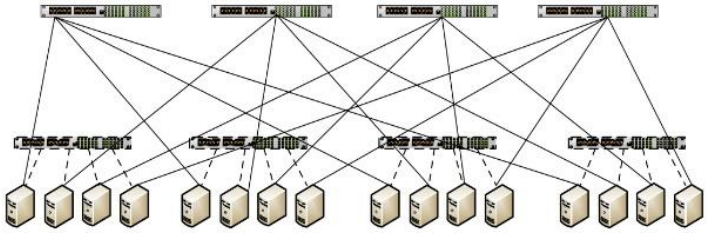
四、流优先级调度全网优化

物理网络拓扑优化

- Fat-Tree逐渐成为现代数据中心中常用的网络拓扑
  - Fat-Tree是为数据中心中无法预测的流量模式而设计的无阻塞网络拓扑
  - 分布式机器学习产生的流量模式是可预测的、规则的

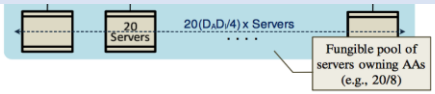


Fat-Tree

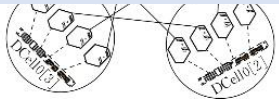


BCube

借助分布式机器学习流量模式的“可预测性”，为其适配底层物理网络拓扑，以降低参数同步时间



VL2

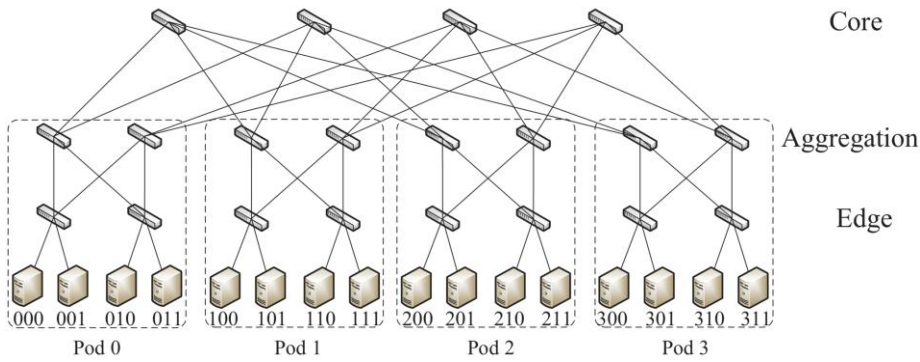


DCell

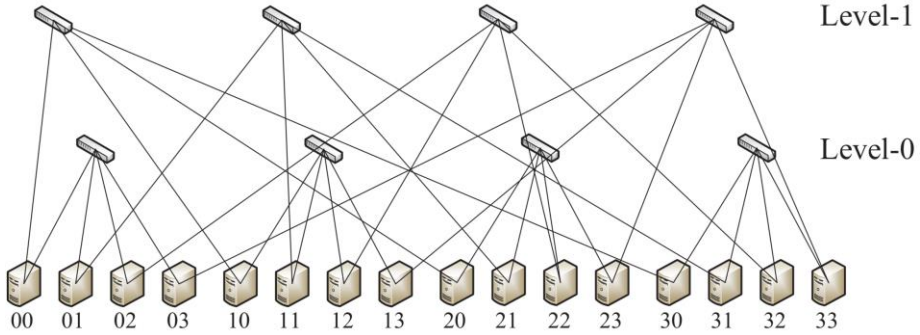
FiConn

物理网络拓扑优化

- Fat-Tree
  - 以交换机为中心的代表拓扑
  - 服务器上只有1块网卡
  - 只有交换机参与数据包转发
  - PFC暂停帧易大范围蔓延
  - ECMP 实现流量负载均衡



- BCube
  - 以服务器为中心的代表拓扑
  - 服务器上有k (k>1)块网卡
  - 服务器也要参与数据包转发
  - 服务器可以吸收PFC暂停帧
  - 应用可以控制传输路径，实现流量负载均衡



- 同等网络规模BCube和Fat-Tree的交换机数量比为$\frac{k}{5}$

物理网络拓扑优化

- 理论全局参数同步时间GST
  - Fat-Tree的GST与算法无关
  - 结合层次化参数同步算法，BCube的GST比Fat-Tree小很多

	Fat-Tree	BCube
mesh-based synchronization	$\frac{2(N-1)}{N}\frac{P}{B}$	$\frac{2(\sqrt[k]{N}-1)}{\sqrt[k]{N}}\frac{P}{B}$
ring-based synchronization	$\frac{2(N-1)}{N}\frac{P}{B}$	$\begin{cases}\frac{N-1}{N}\frac{P}{B}, & n \text{ and } k \text{ are even}\\ \frac{2(N-1)}{N}\frac{P}{B}, & \text{otherwise}\end{cases}$
HiPS	$\frac{2(N-1)}{N}\frac{P}{B}$	$\frac{2(N-1)}{kN}\frac{P}{B}$

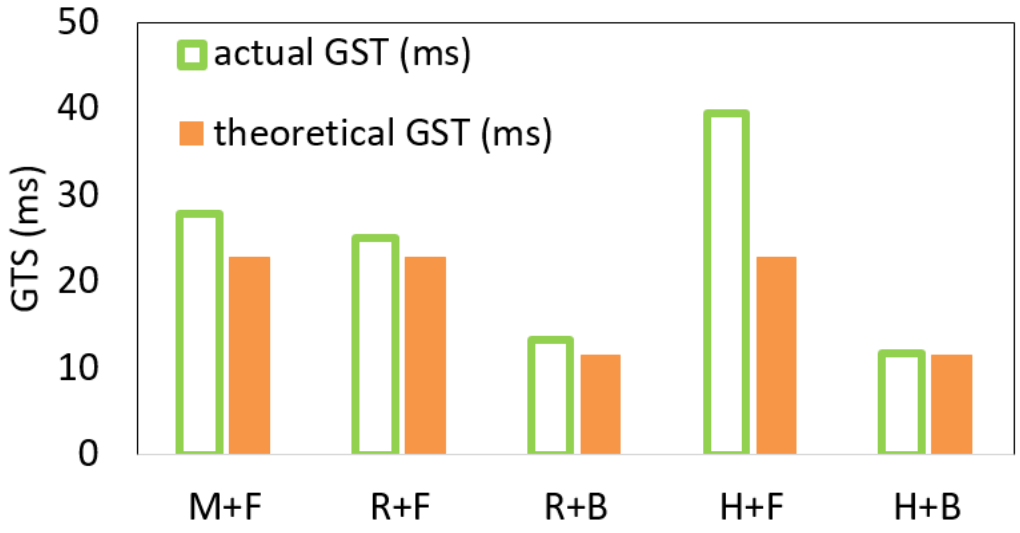
BCube中，多网卡配置提高了服务器的进出口带宽，为参数同步并行化提供了更好的支持

物理网络拓扑优化

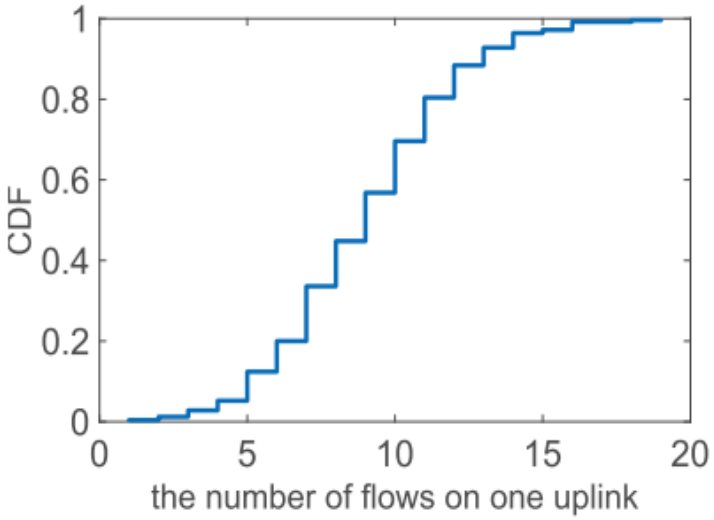
- NS-3仿真参数同步过程
  - M+F : Mesh-based synchronization in Fat-Tree
  - R+F : Ring-based synchronization in Fat-Tree
  - R+B : Ring-based synchronization in BCube
  - H+F : HiPS in Fat-Tree
  - H+B : HiPS in BCube
- 仿真配置
  - 使用RoCE v2传输协议, 使能PFC和DCQCN
  - 带宽为40Gbps, 传播时延为1μs
  - 使用ECMP做流量负载均衡
  - Fat-Tree: 125台10-port交换机连接250台服务器
  - BCube: 32台16-port交换机连接256台服务器 (2层)
  - 同步参数量设为57.5 MBytes (VGG-19参数量的1/10)

物理网络拓扑优化

- ECMP哈希碰撞进一步降低了Fat-Tree的整体网络利用率，增大参数同步时间，而BCube的源路由机制带来了更好的负载均衡性能



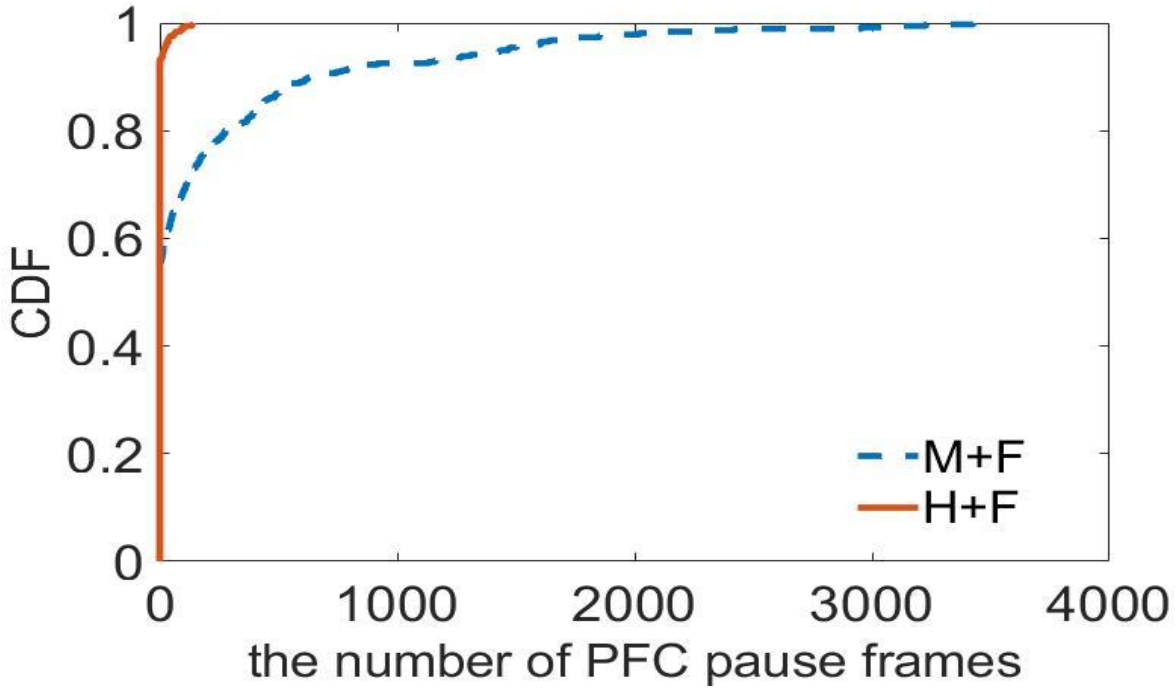
仿真结果和理论计算结果对比



H+F下agg-core链路流数量分布

物理网络拓扑优化

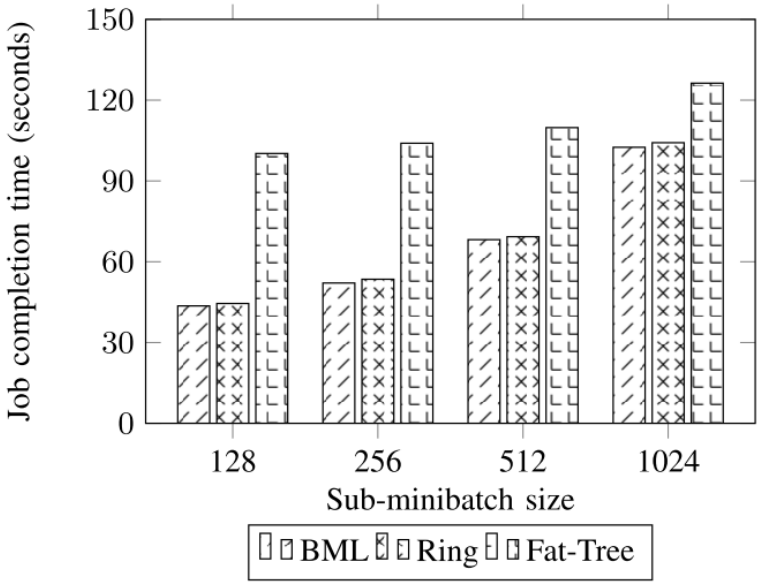
- Fat-Tree中，交换机的有限缓存容易造成RDMA PFC暂停帧的大范围蔓延，而BCube中服务器的内存可以有效吸收暂停帧，防止其进一步蔓延



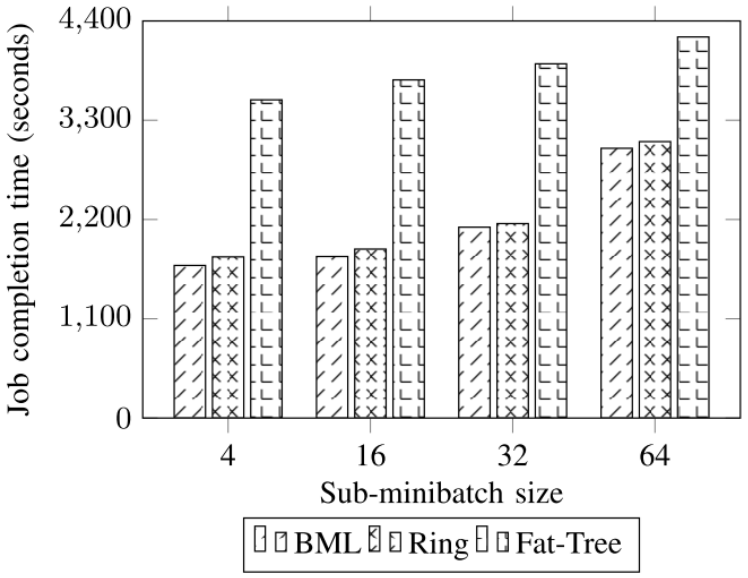
各节点接收PFC暂停帧数量的分布

端到端训练性能

- 基于BCube拓扑实现了分层参数同步算法BML
  - 相比Ring和Fat-Tree，BCube可提升端到端训练吞吐



LeNet-5模型



VGG-19模型

报告提纲

一、研究背景与相关工作

二、层次化参数同步算法

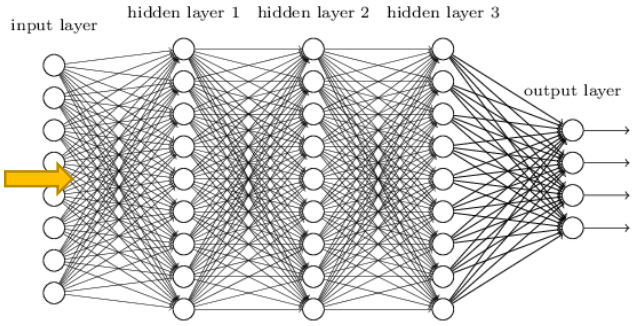
三、物理网络互联拓扑优化

四、流优先级调度全网优化

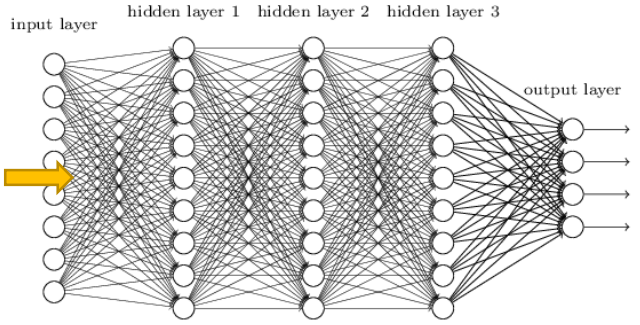
参数优先级问题

- 模型的训练过程是计算和通信相互依赖的：前向传播时，计算依赖于通信；反向传播时，通信依赖于计算
- 根据模型计算和通信的依赖关系，对参数传输数据流进行有效调度，合理利用带宽资源，重叠通信和计算，隐藏通信开销

调度前

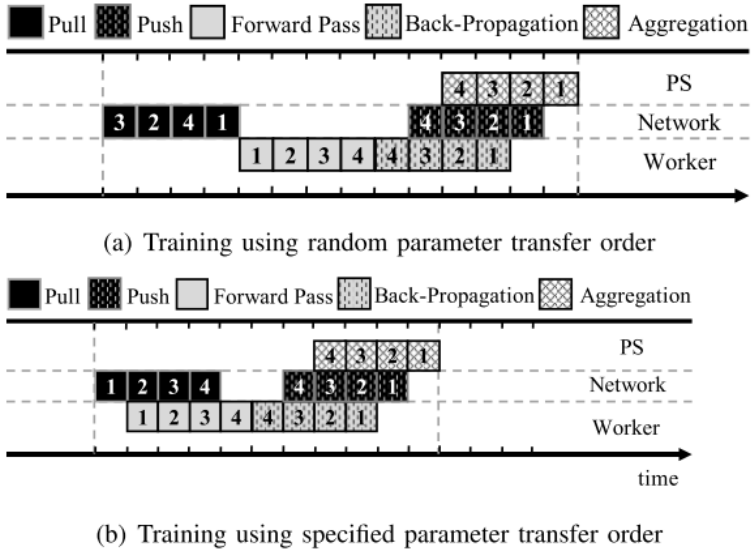


调度后



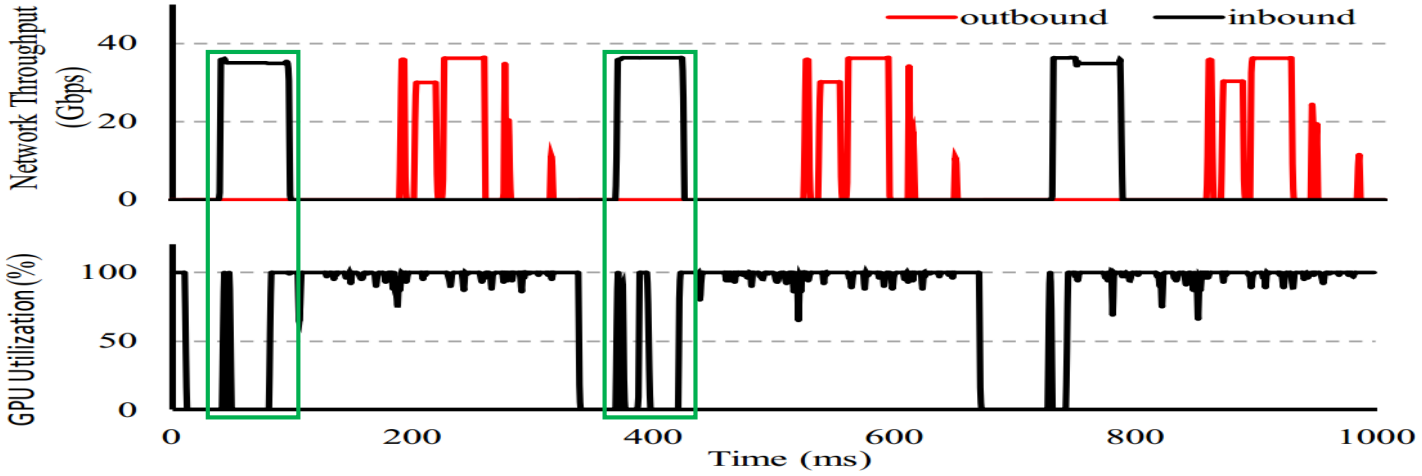
参数优先级问题

- 参数传输顺序具有随机性
- PS和worker间使用单channel通信
  - 后发送的参数需要经历更长的排队时延
  - 相应的参数未就位时，计算资源不得不闲置



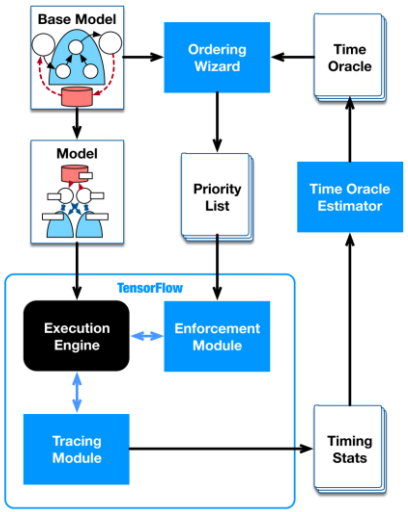
(a) Training using random parameter transfer order

(b) Training using specified parameter transfer order



端侧调度方法及不足

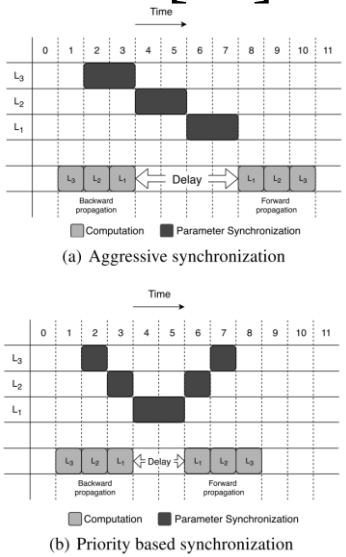
TicTac[40]



方法：使用多线程严格控制发送顺序

不足：非抢占式调度浪费网络资源和多线程开销

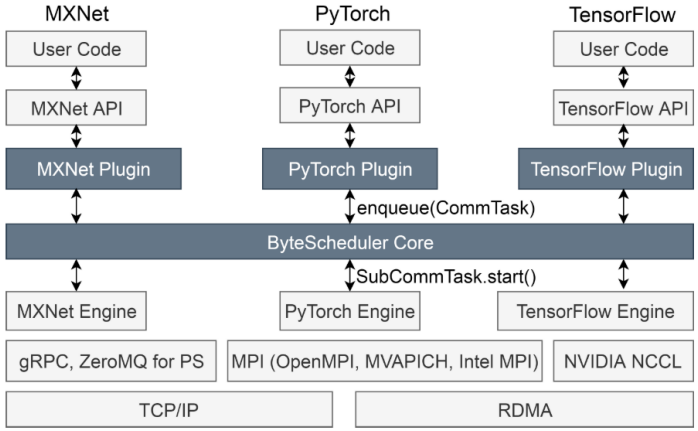
P3[41]



方法：使用切片和优先级发送队列实现抢占式调度

不足：阻塞式通信效率低和切片粒度不易调节

ByteScheduler[42]

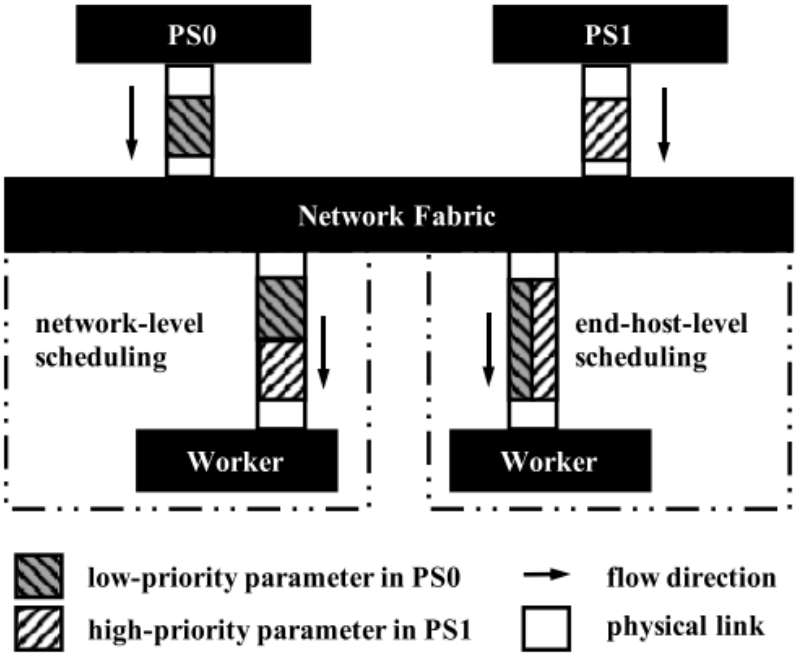


方法：统一调度框架和自适应调节切片和发送窗口

不足：端侧调度难以协调多个节点

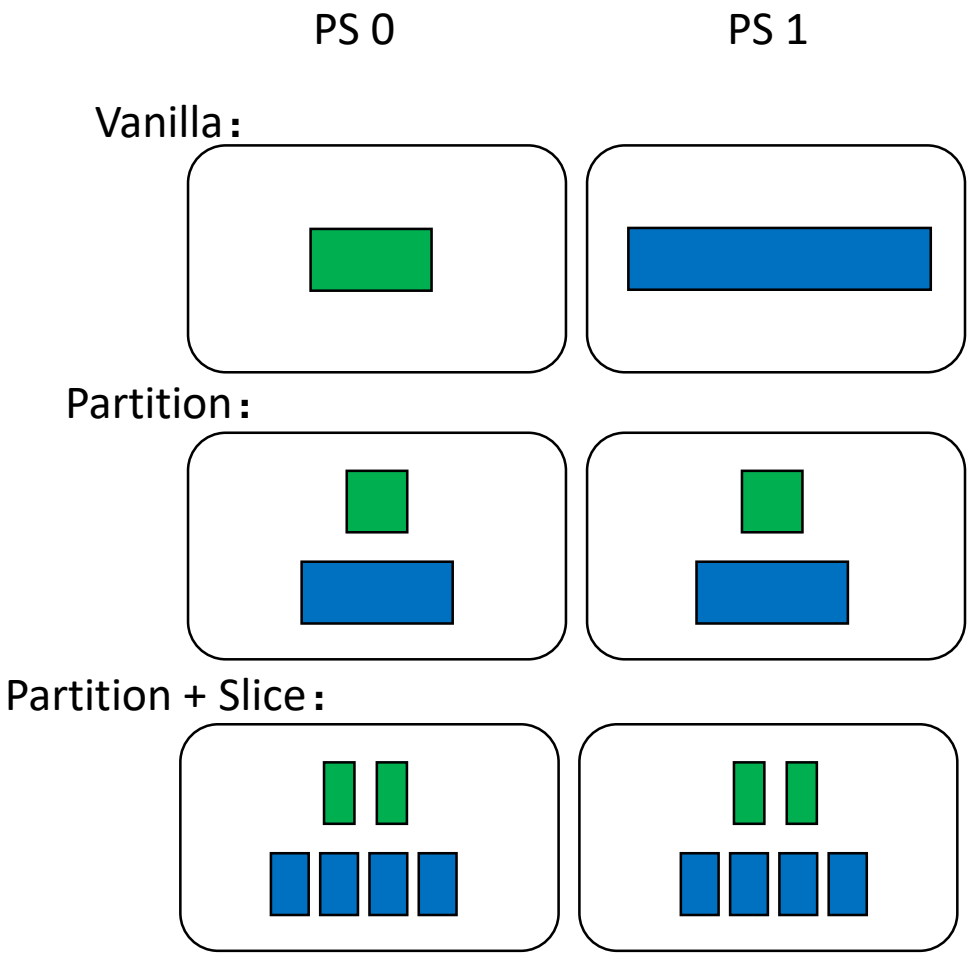
流优先级调度全网优化

- 基于流优先级调度的参数传输优化
  - 不同的参数使用不同的流传输
  - 承载紧急程度高的参数的流赋予高优先级
  - 网络配置优先级流调度策略
- 优点
  - 保证紧急参数的及时传输，实现计算和通信的重叠
  - 利用网络流调度实现数据包级别的抢占式调度
  - 可协调不同PS上的参数传输顺序



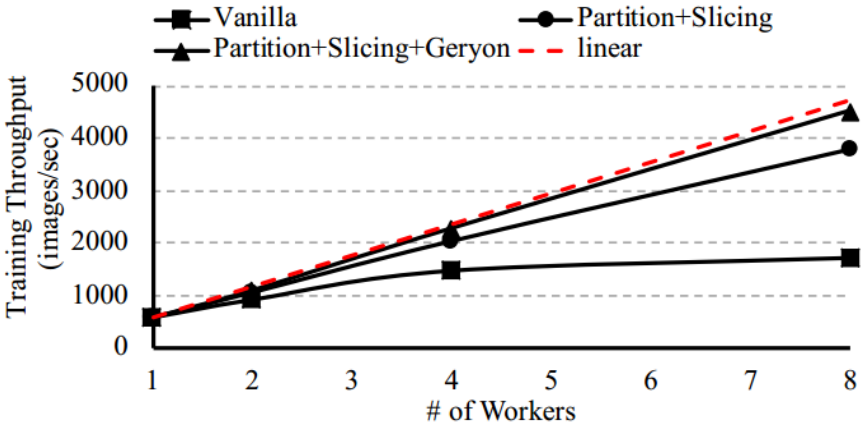
流优先级调度全网优化

- tensor partition (划分)
  - 多个PS间达到计算的负载均衡和通信的负载均衡
- tensor slice (切片)
  - 通过更细粒度的切片使得PS上计算和通信可以重叠，并且推送梯度和拉取参数也可以重叠，从而利用全双工带宽

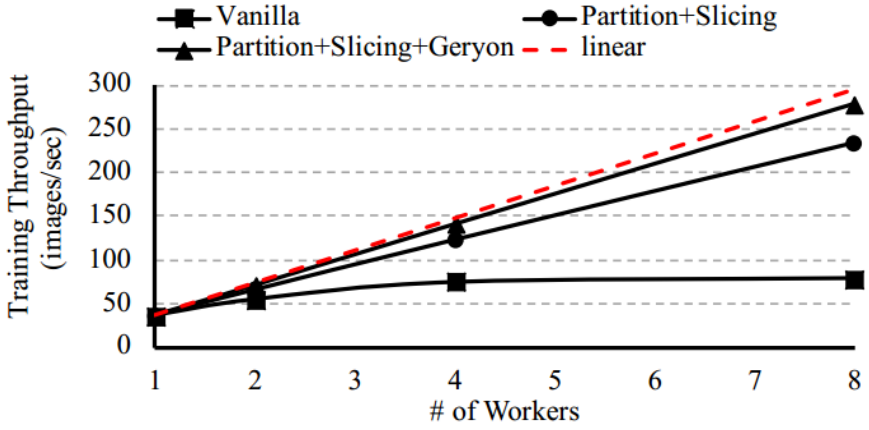


流优先级调度全网优化

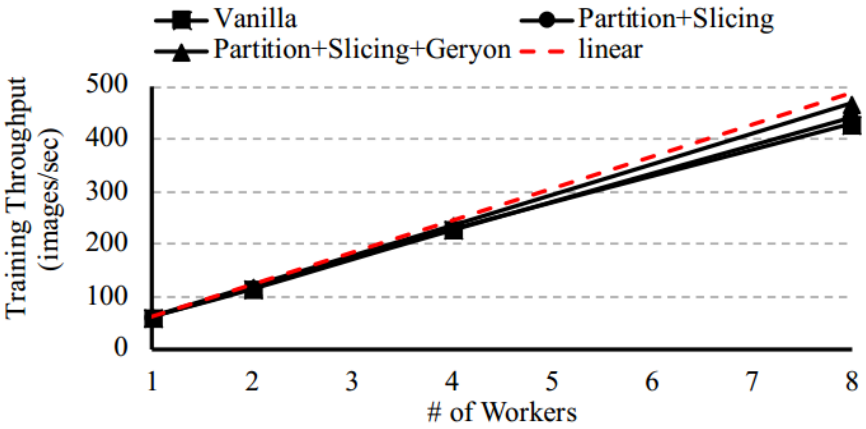
- 10GbE RDMA下的性能对比



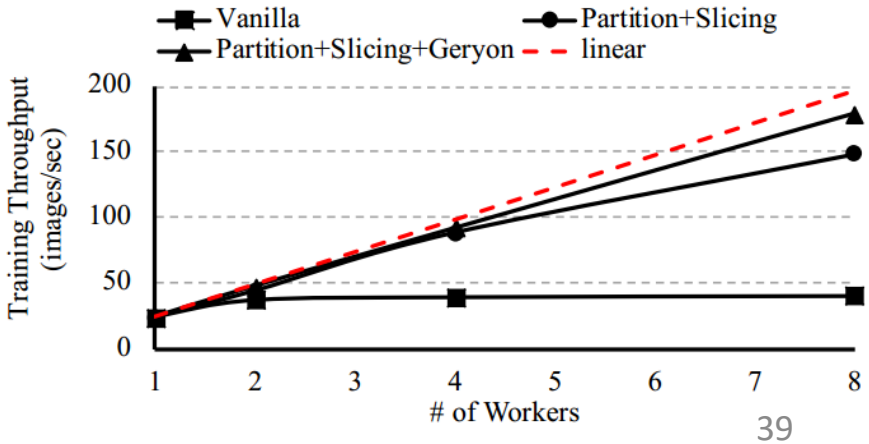
(a) AlexNet



(b) VGG-19



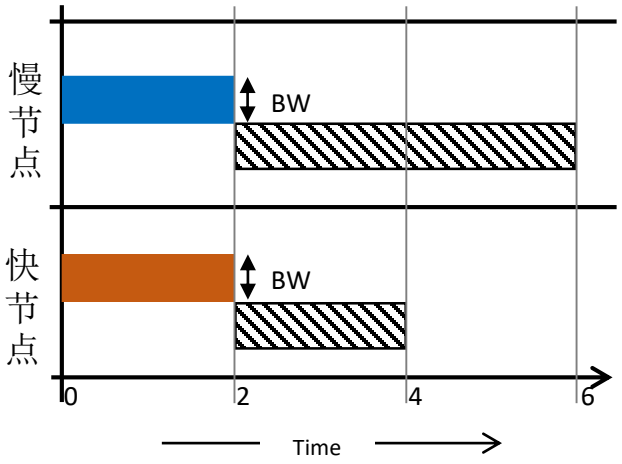
(c) ResNet-50



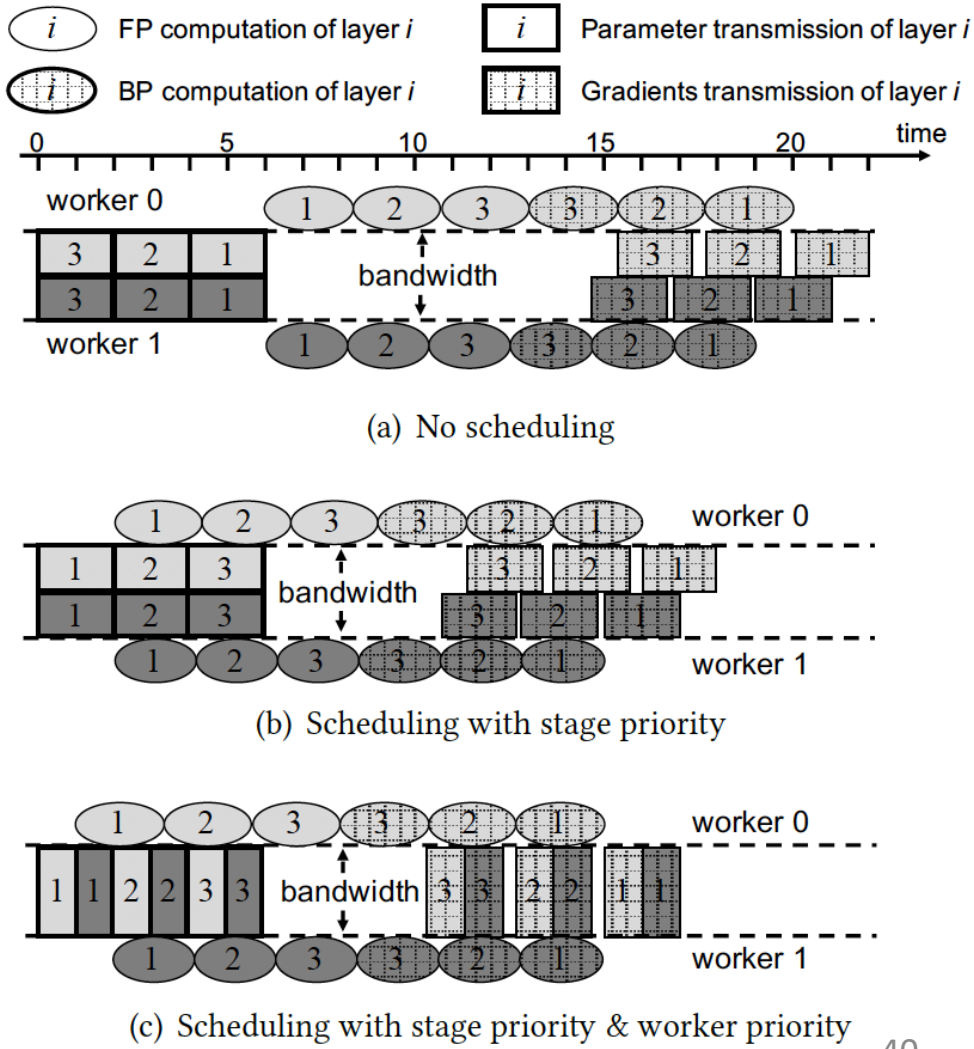
(d) YOLO

worker优先级调度

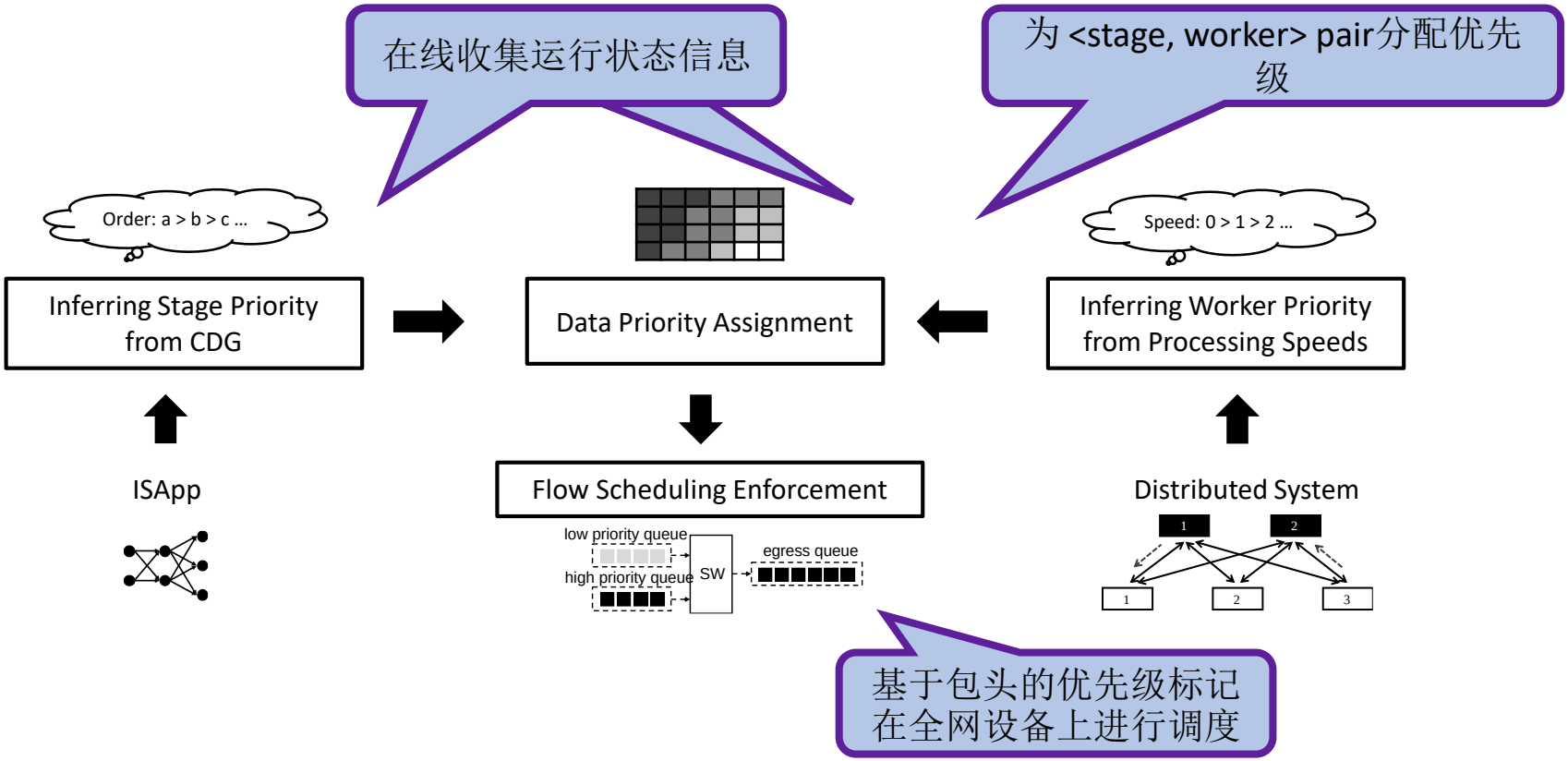
- 基于worker priority的流调度
  - ✓ 对多个worker，优先发送给处理速度慢的worker，使其更早地开始计算，避免拖慢整体速度



公平共享带宽



CEFS 架构


在线收集运行状态信息
为 <stage, worker> pair分配优先级
Order: a > b > c ...
Speed: 0 > 1 > 2 ...
Inferring Stage Priority from CDG
Data Priority Assignment
Inferring Worker Priority from Processing Speeds
ISApp
Flow Scheduling Enforcement
Distributed System
low priority queue
high priority queue
SW
egress queue
1
2
1
2
3
基于包头的优先级标记
在全网设备上进行调度

worker优先级调度效果

- 相比于TensorFlow
  - ✓ 只基于参数优先级进行调度的训练吞吐提升为5%~15%
  - ✓ 引入worker优先级调度的训练吞吐提升为12%~23%

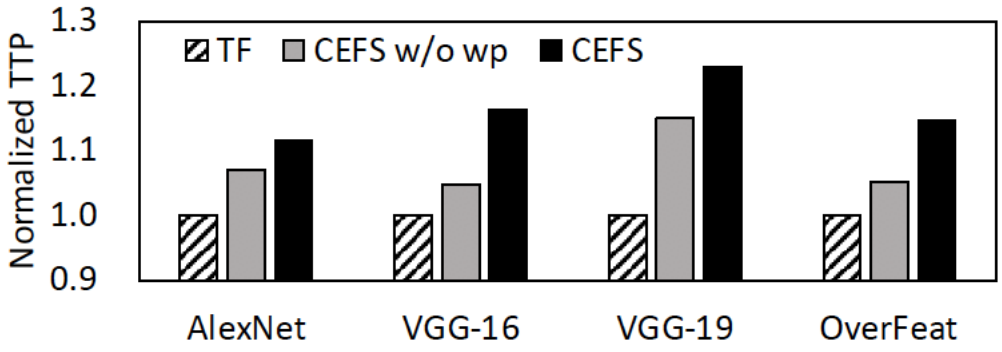
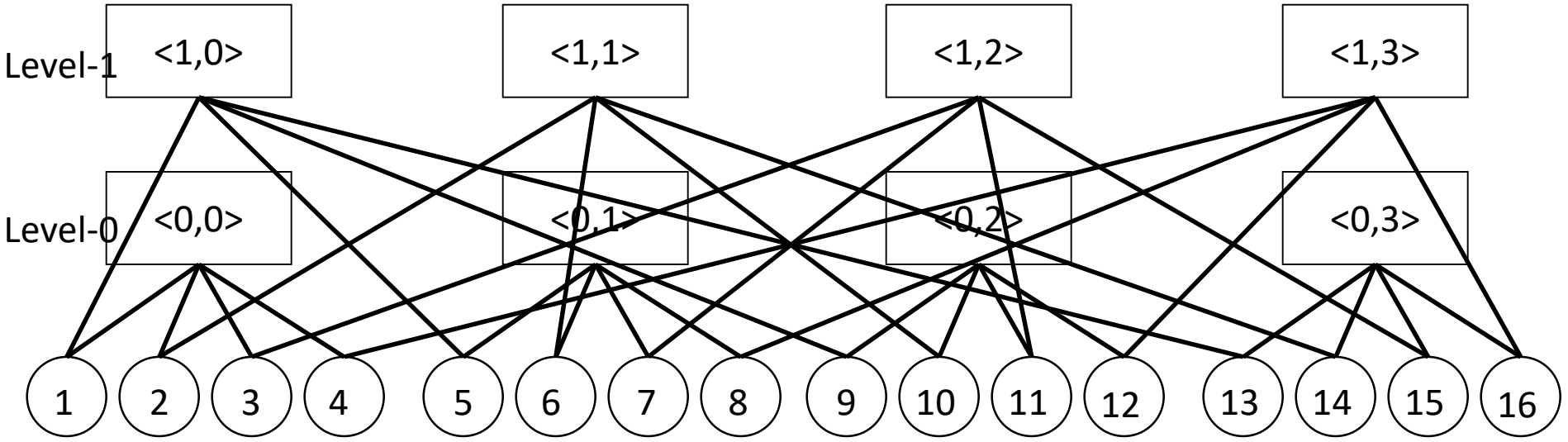


Figure 14: Breakdown of performance improvement between stage priority and worker priority at 25Gbps bandwidth.

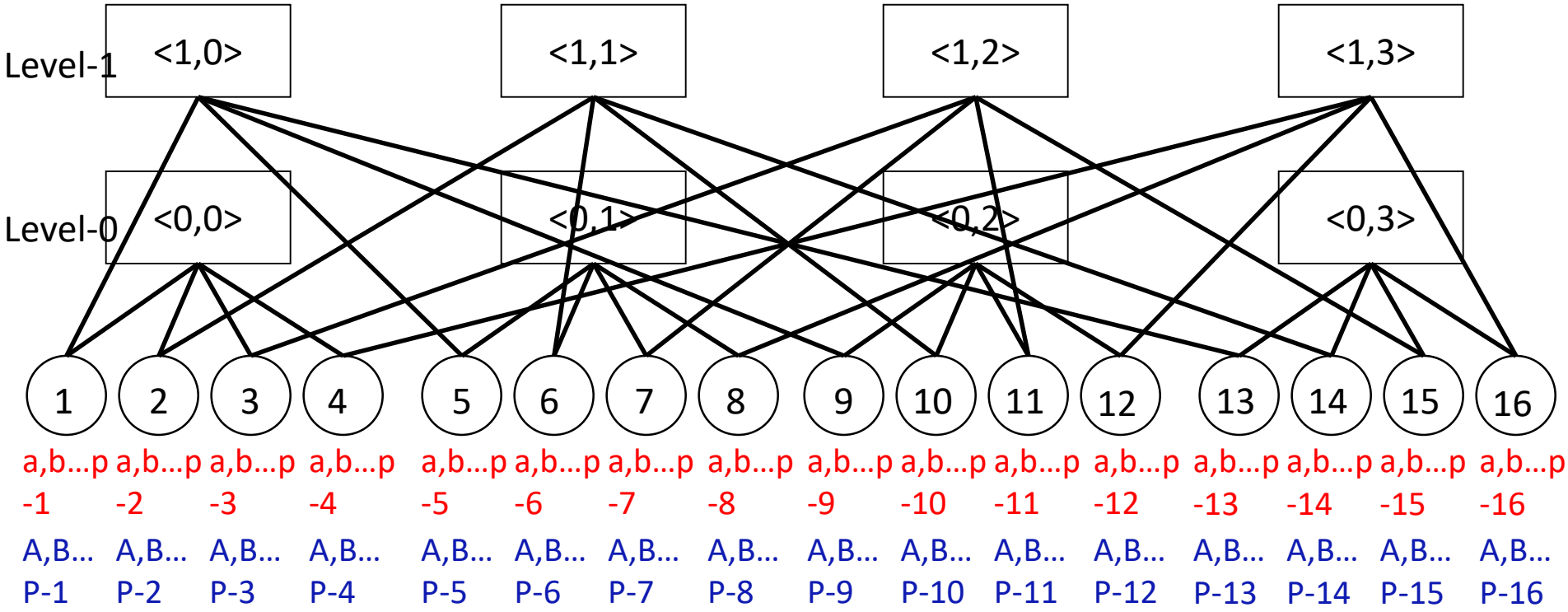
谢谢大家！

BML: High Performance

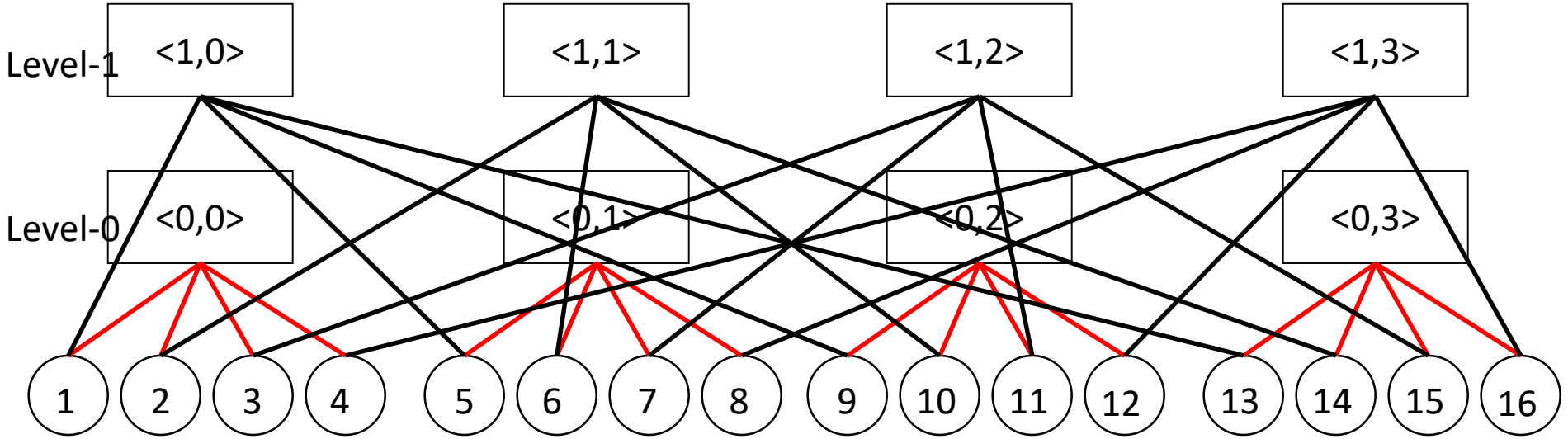


- Divide the parameter into 32 sets and use a,b,…,p,A,B,…,P to denote
- It takes T/32 time to transmit a parameter set in the link

BML Performance: t=0

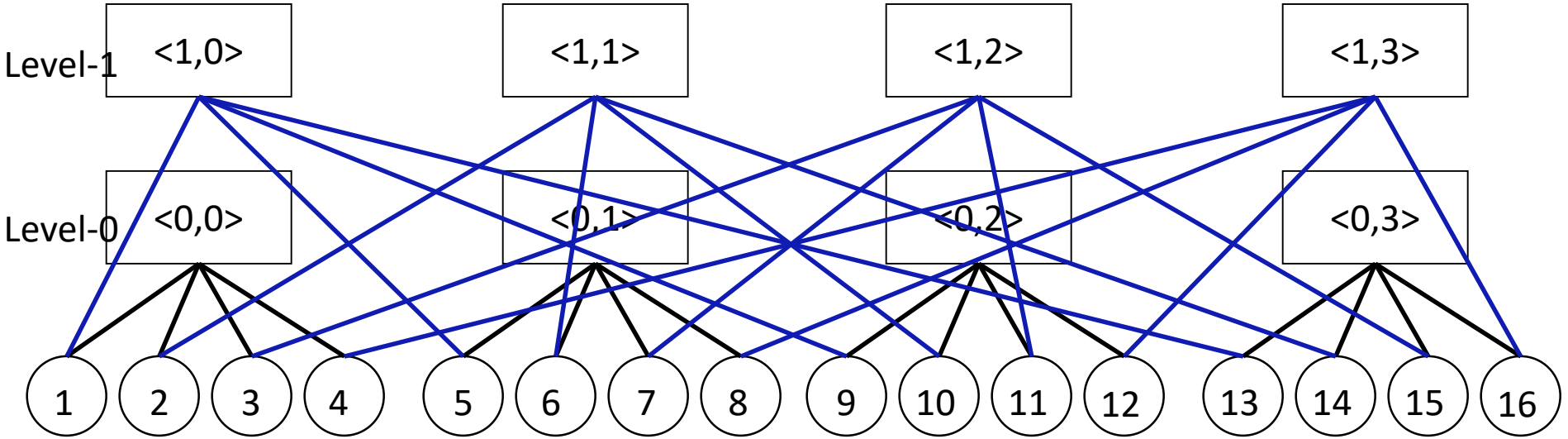


BML Performance: t=12*T/32



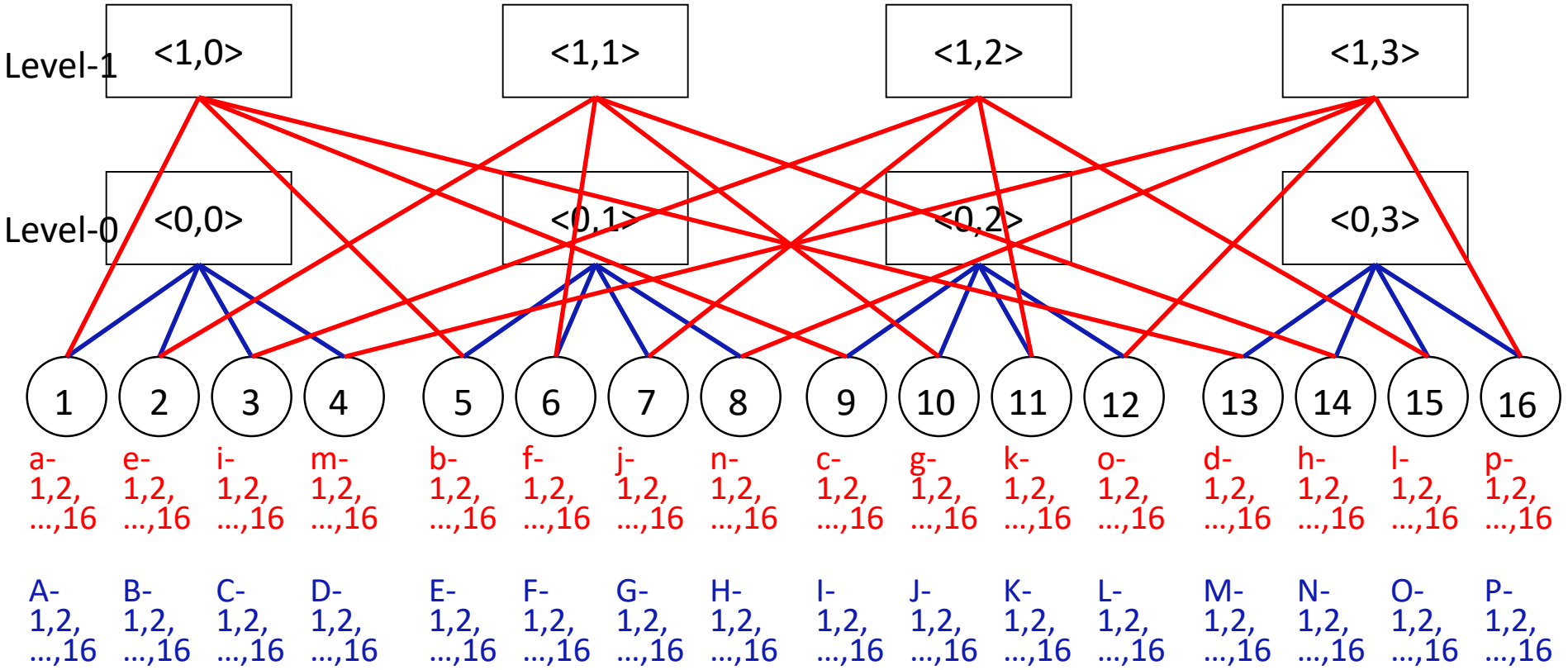
a-1,2,3,4	e-1,2,3,4	i-1,2,3,4	m-1,2,3,4	a-5,6,7,8	e-5,6,7,8	i-5,6,7,8	m-5,6,7,8	a-9,10,11,12	e-9,10,11,12	i-9,10,11,12	m-9,10,11,12	a-13,14,15,16	e-13,14,15,16	i-13,14,15,16	m-13,14,15,16
b-1,2,3,4	f-1,2,3,4	j-1,2,3,4	n-1,2,3,4	b-5,6,7,8	f-5,6,7,8	j-5,6,7,8	n-5,6,7,8	b-9,10,11,12	f-9,10,11,12	j-9,10,11,12	n-9,10,11,12	b-13,14,15,16	f-13,14,15,16	j-13,14,15,16	n-13,14,15,16
c-1,2,3,4	g-1,2,3,4	k-1,2,3,4	o-1,2,3,4	c-5,6,7,8	g-5,6,7,8	k-5,6,7,8	o-5,6,7,8	c-9,10,11,12	g-9,10,11,12	k-9,10,11,12	o-9,10,11,12	c-13,14,15,16	g-13,14,15,16	k-13,14,15,16	o-13,14,15,16
d-1,2,3,4	h-1,2,3,4	l-1,2,3,4	p-1,2,3,4	d-5,6,7,8	h-5,6,7,8	l-5,6,7,8	p-5,6,7,8	d-9,10,11,12	h-9,10,11,12	l-9,10,11,12	p-9,10,11,12	d-13,14,15,16	h-13,14,15,16	l-13,14,15,16	p-13,14,15,16

BML Performance: t=12*T/32

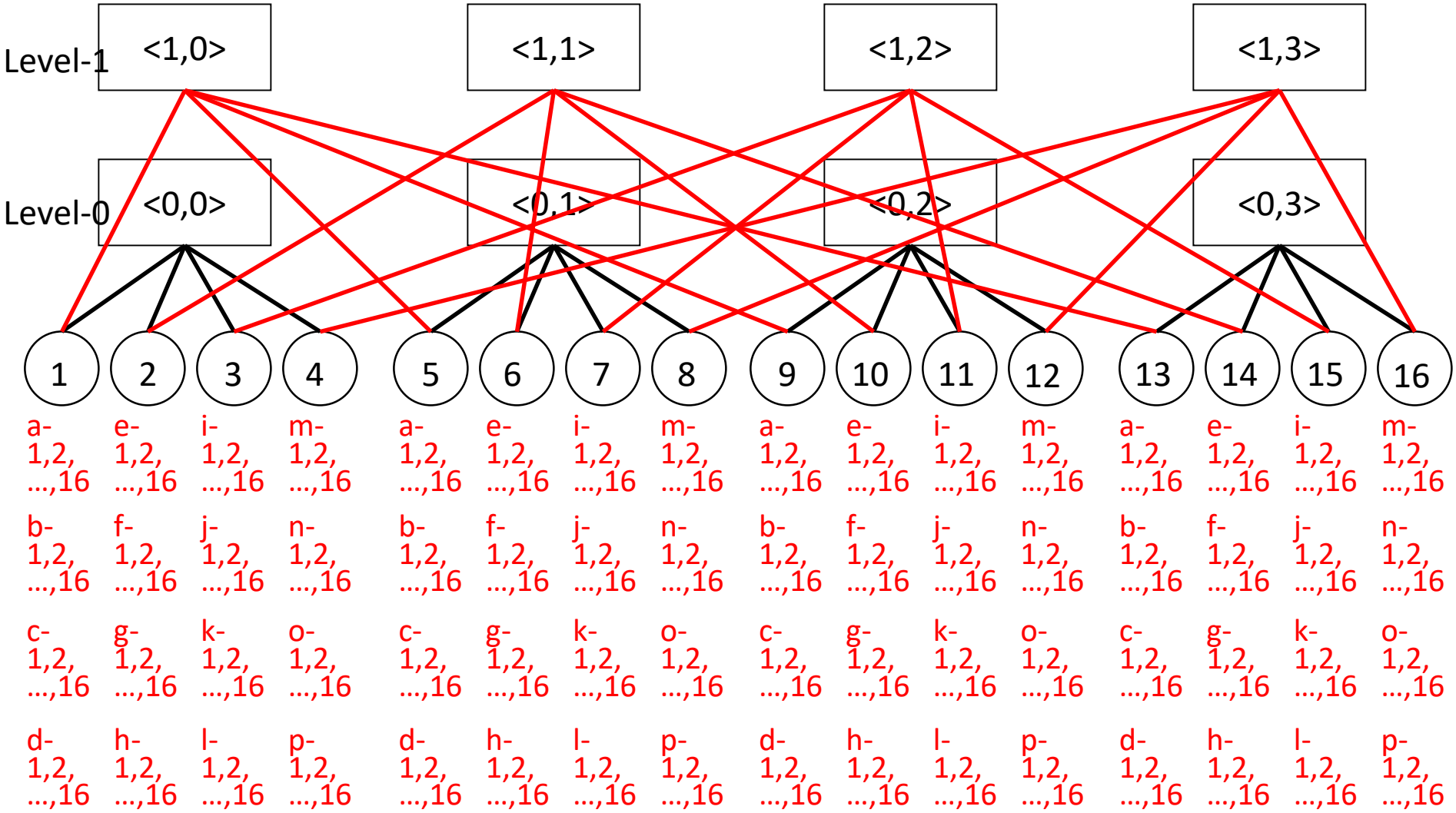


A-1,5,9,13	A-2,6,10,14	A-3,7,11,15	A-4,8,12,16	E-1,5,9,13	E-2,6,10,14	E-3,7,11,15	E-4,8,12,16	I-1,5,9,13	I-2,6,10,14	I-3,7,11,15	I-4,8,12,16	M-1,5,9,13	M-2,6,10,14	M-3,7,11,15	M-4,8,12,16
B-1,5,9,13	B-2,6,10,14	B-3,7,11,15	B-4,8,12,16	F-1,5,9,13	F-2,6,10,14	F-3,7,11,15	F-4,8,12,16	J-1,5,9,13	J-2,6,10,14	J-3,7,11,15	J-4,8,12,16	N-1,5,9,13	N-2,6,10,14	N-3,7,11,15	N-4,8,12,16
C-1,5,9,13	C-2,6,10,14	C-3,7,11,15	C-4,8,12,16	G-1,5,9,13	G-2,6,10,14	G-3,7,11,15	G-4,8,12,16	K-1,5,9,13	K-2,6,10,14	K-3,7,11,15	K-4,8,12,16	O-1,5,9,13	O-2,6,10,14	O-3,7,11,15	O-4,8,12,16
D-1,5,9,13	D-2,6,10,14	D-3,7,11,15	D-4,8,12,16	H-1,5,9,13	H-2,6,10,14	H-3,7,11,15	H-4,8,12,16	L-1,5,9,13	L-2,6,10,14	L-3,7,11,15	L-4,8,12,16	P-1,5,9,13	P-2,6,10,14	P-3,7,11,15	P-4,8,12,16

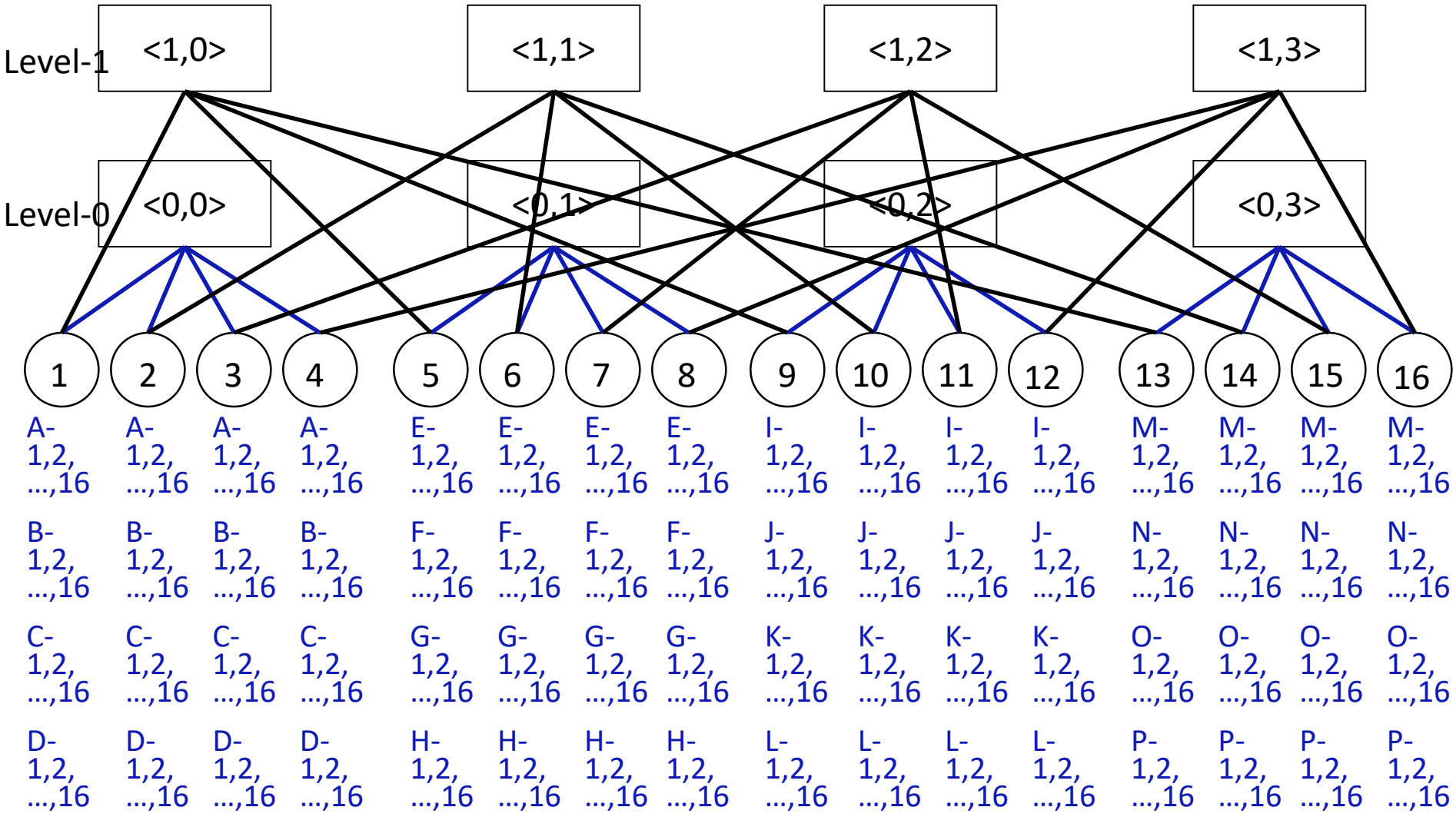
BML Performance: t=15*T/32



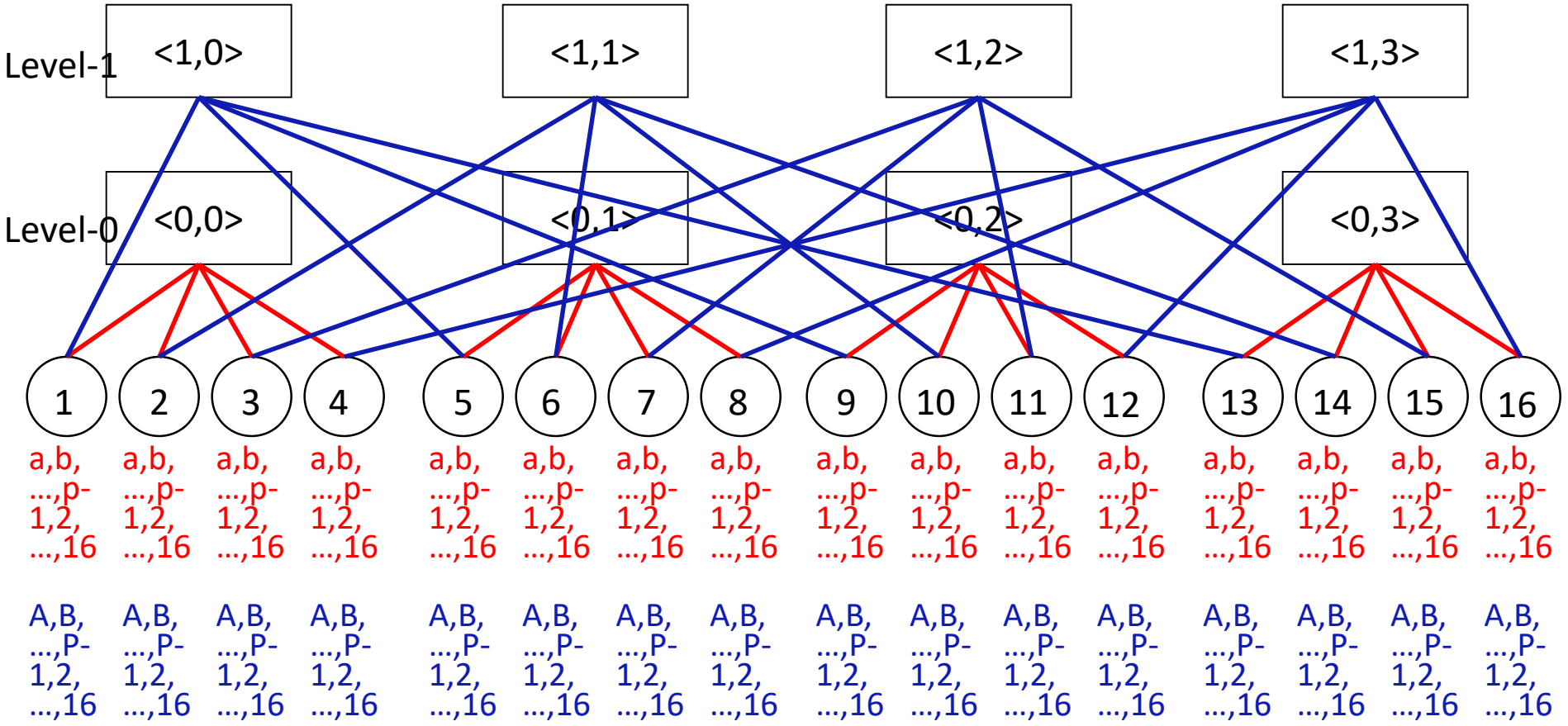
BML Performance: t=18*T/32



BML Performance: t=18*T/32



BML Performance: t=30*T/32



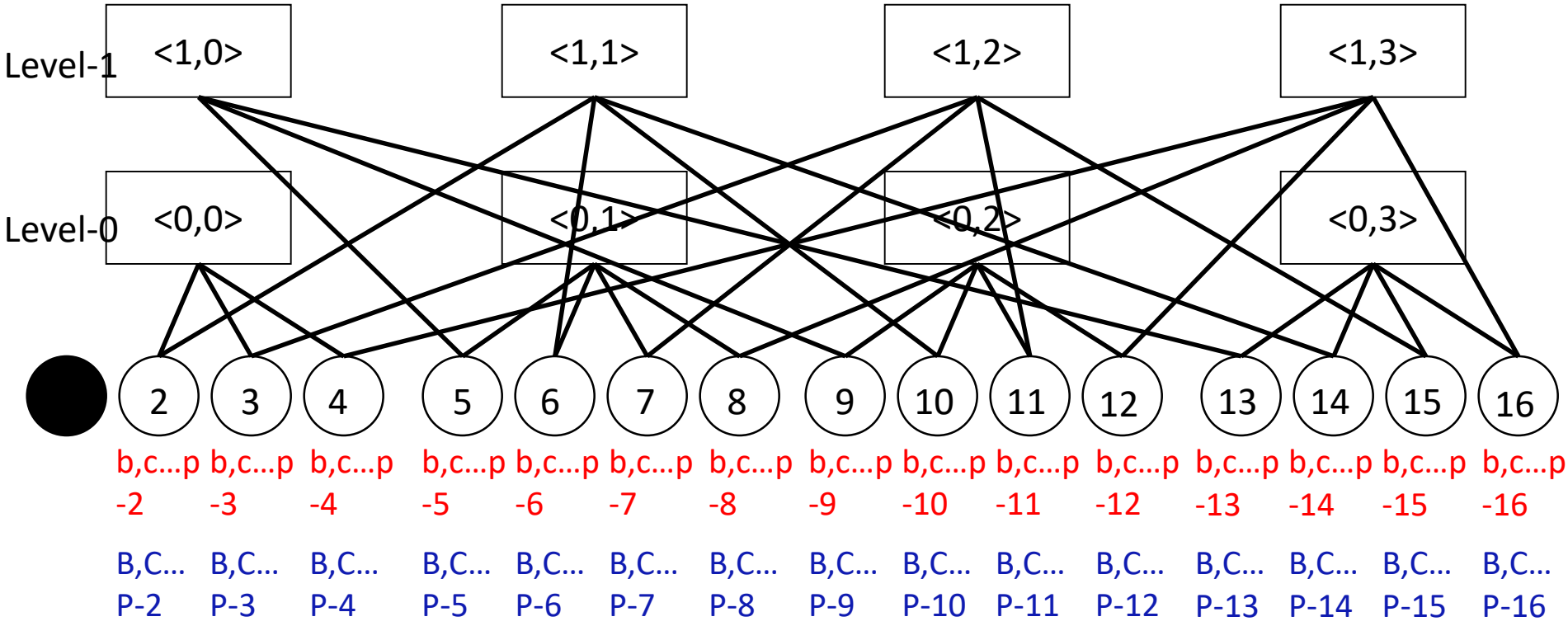
Bcube/BML有更高的性价比

- BCube网络用比Fat-Tree更少的交换机，实现了K倍的分布式机器学习训练加速
  - ✓K为服务器接入网络的接口数量

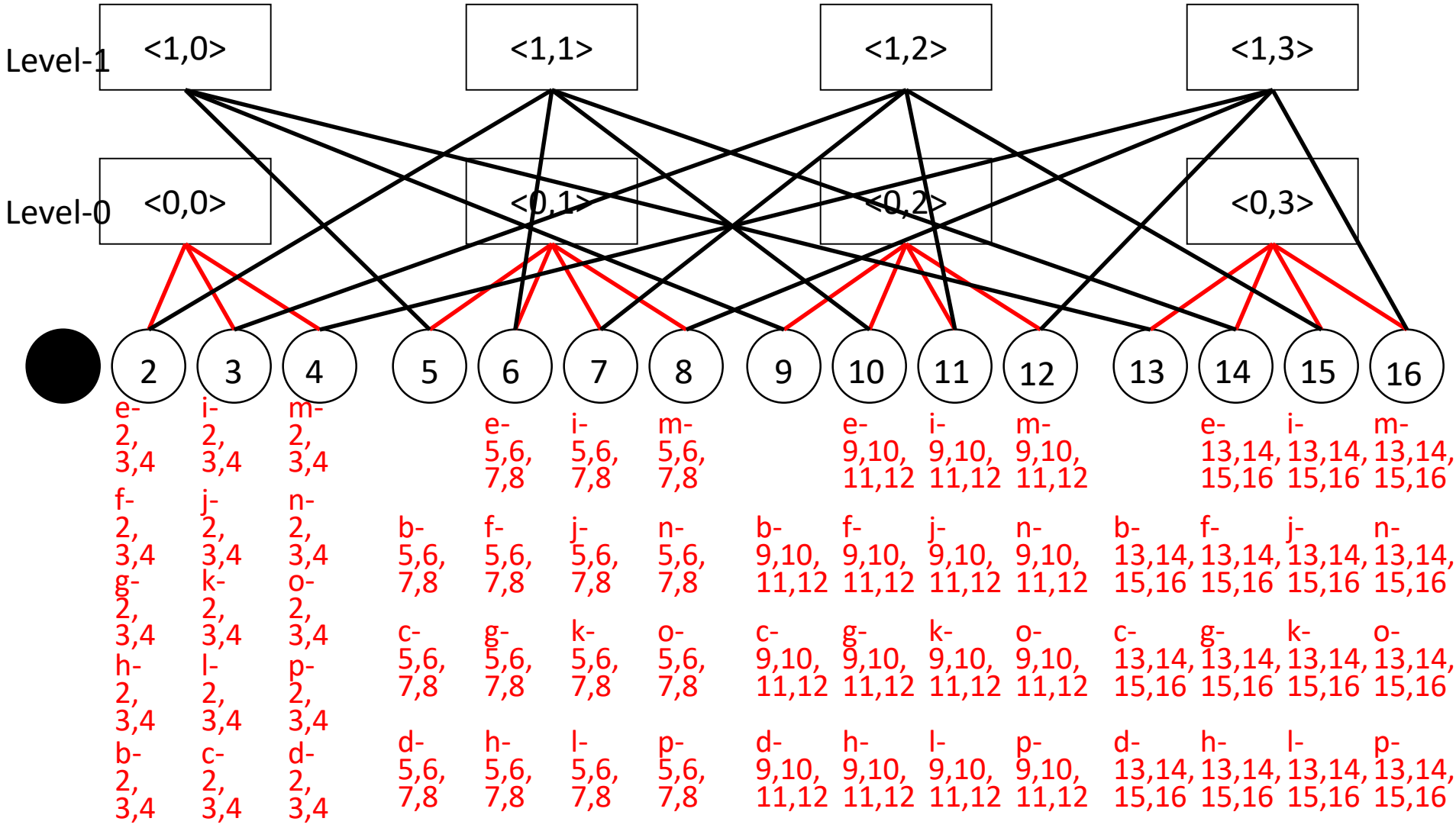
BML Fault Tolerance

- Server failure
  - ✓The remaining servers can work
  - ✓The parameter should be divided into n-f sets if f servers fail
  - ✓The worker computation time in each iteration becomes from t_M/n to $t_M/(n-f)$
- Link failure
  - ✓All the servers can still work
  - ✓Use detour path to bypass the failed link
  - ✓A switch failure equals to failure of all the links attached to it

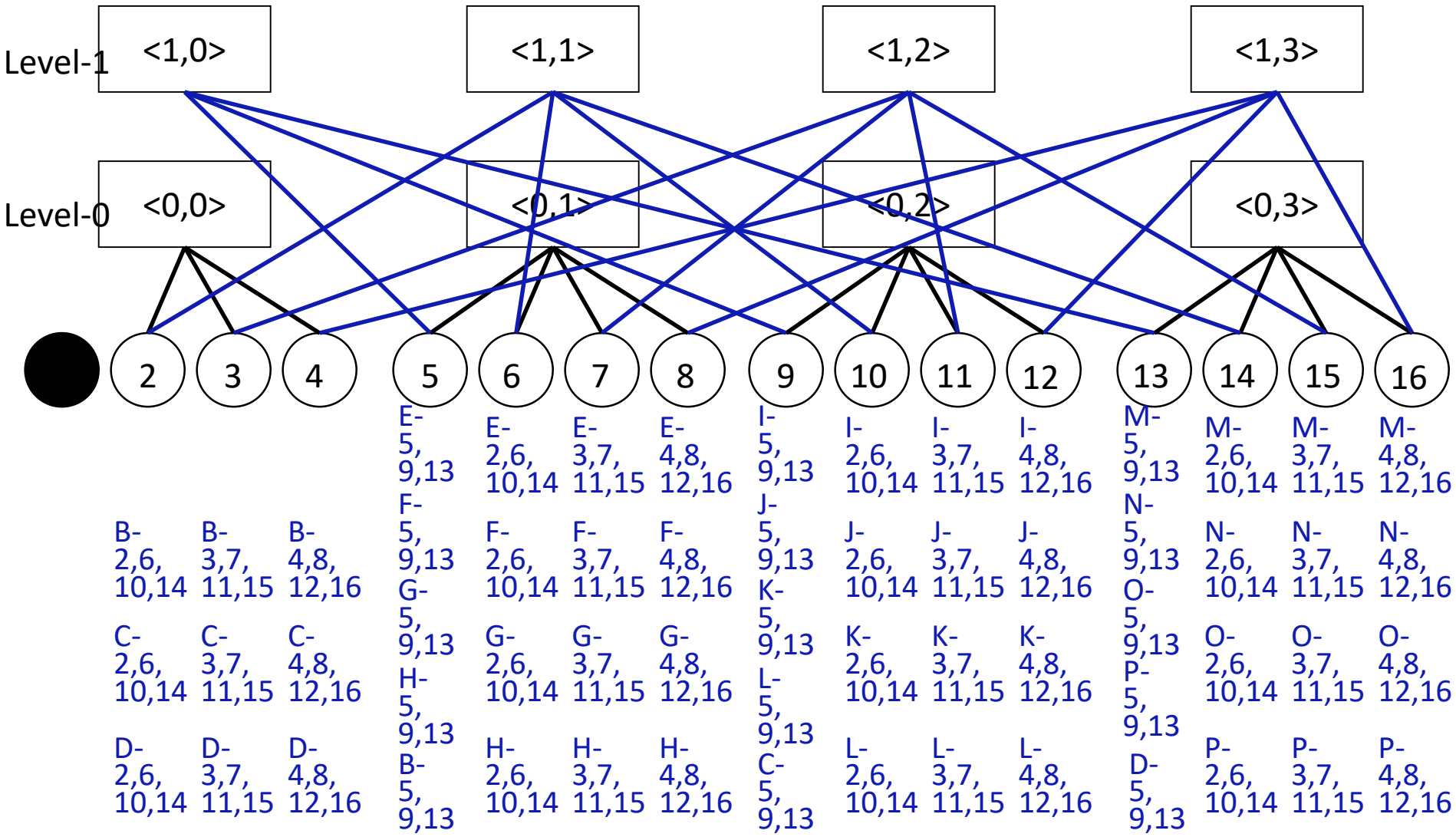
BML Server Failure: t=0



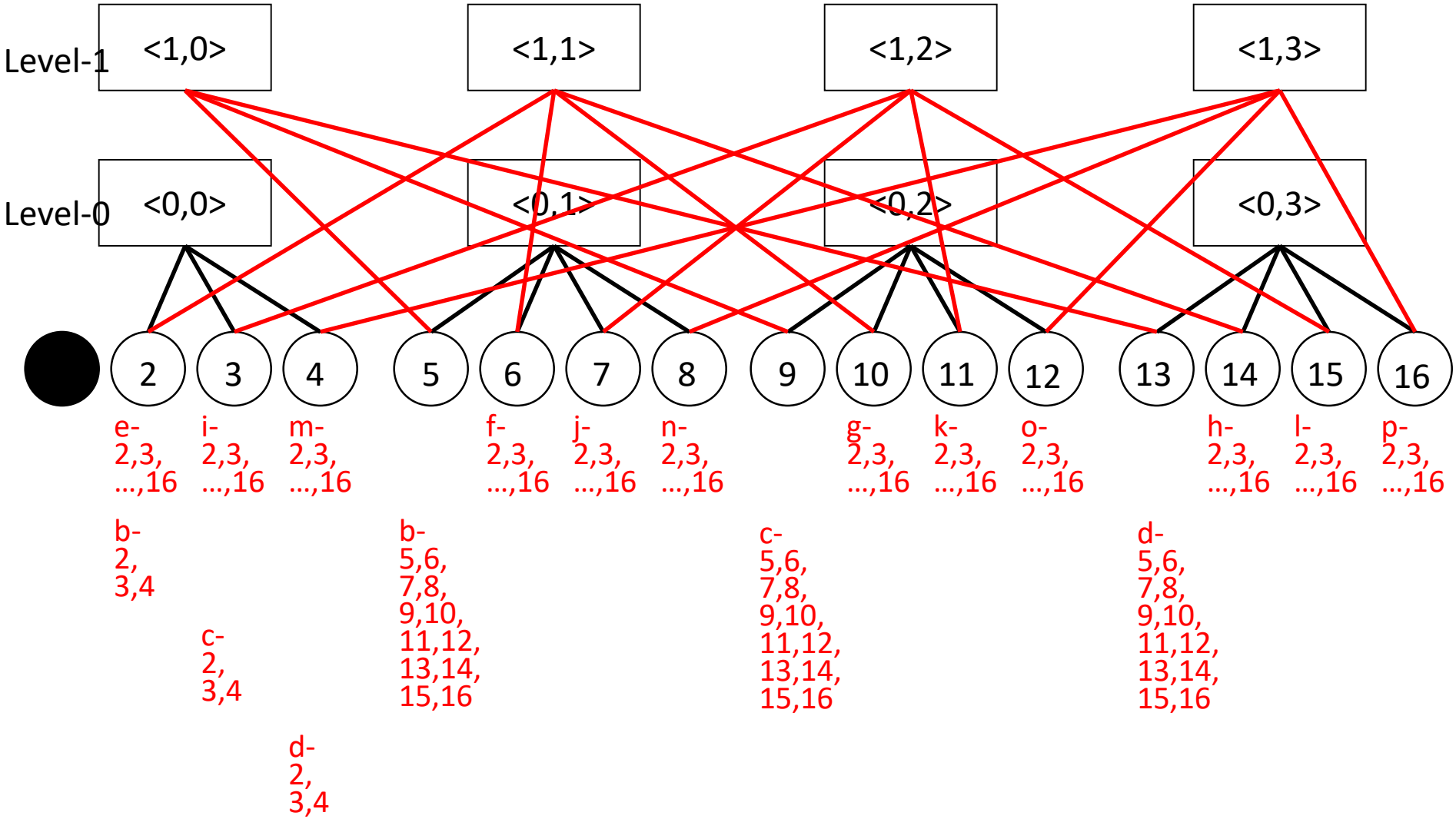
BML Server Failure : t=12*T/30



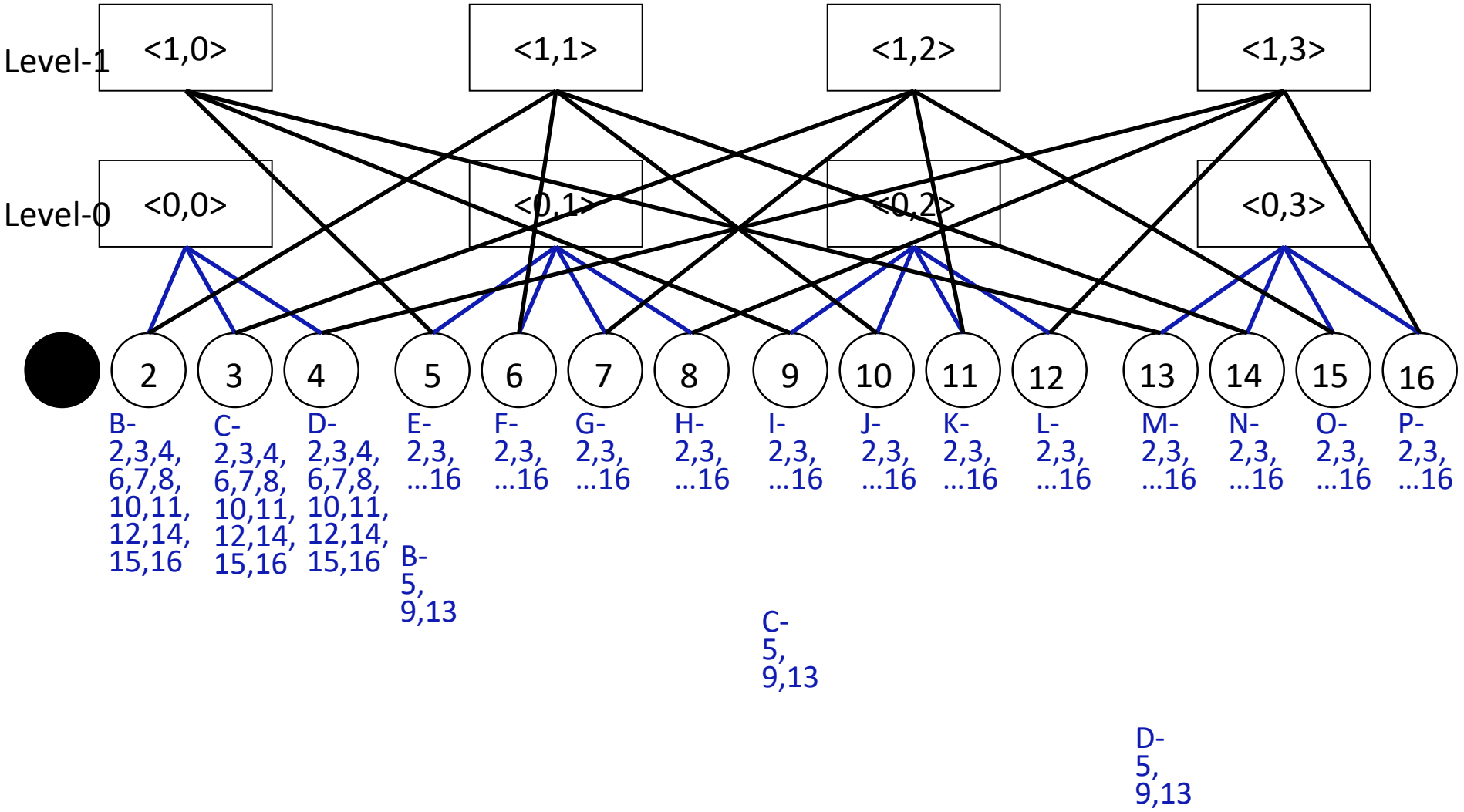
BML Server Failure: t=12*T/30



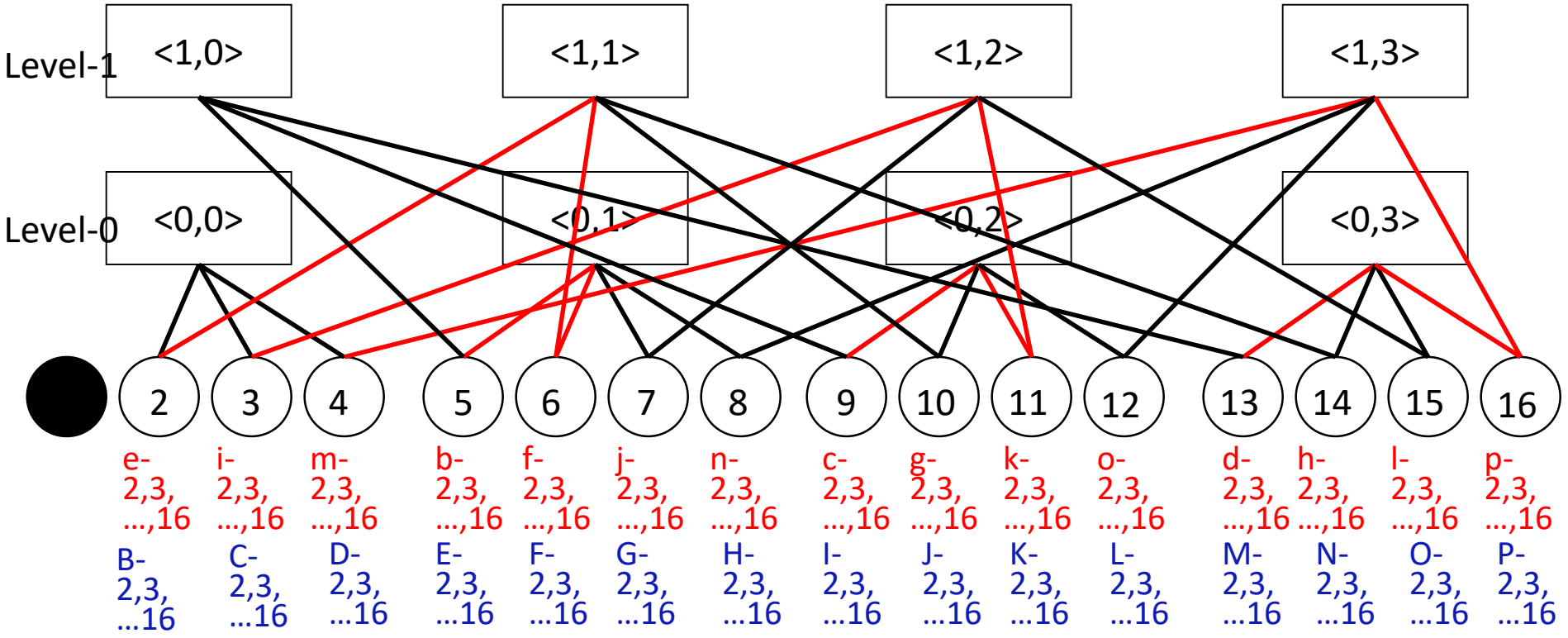
BML Server Failure: t=15*T/30



BML Server Failure: t=15*T/30



BML Server Failure: t=16*T/30



B-2,3,4 takes the detour path of 2->6->5

B-5,9,13 takes the detour path of 5->6->2

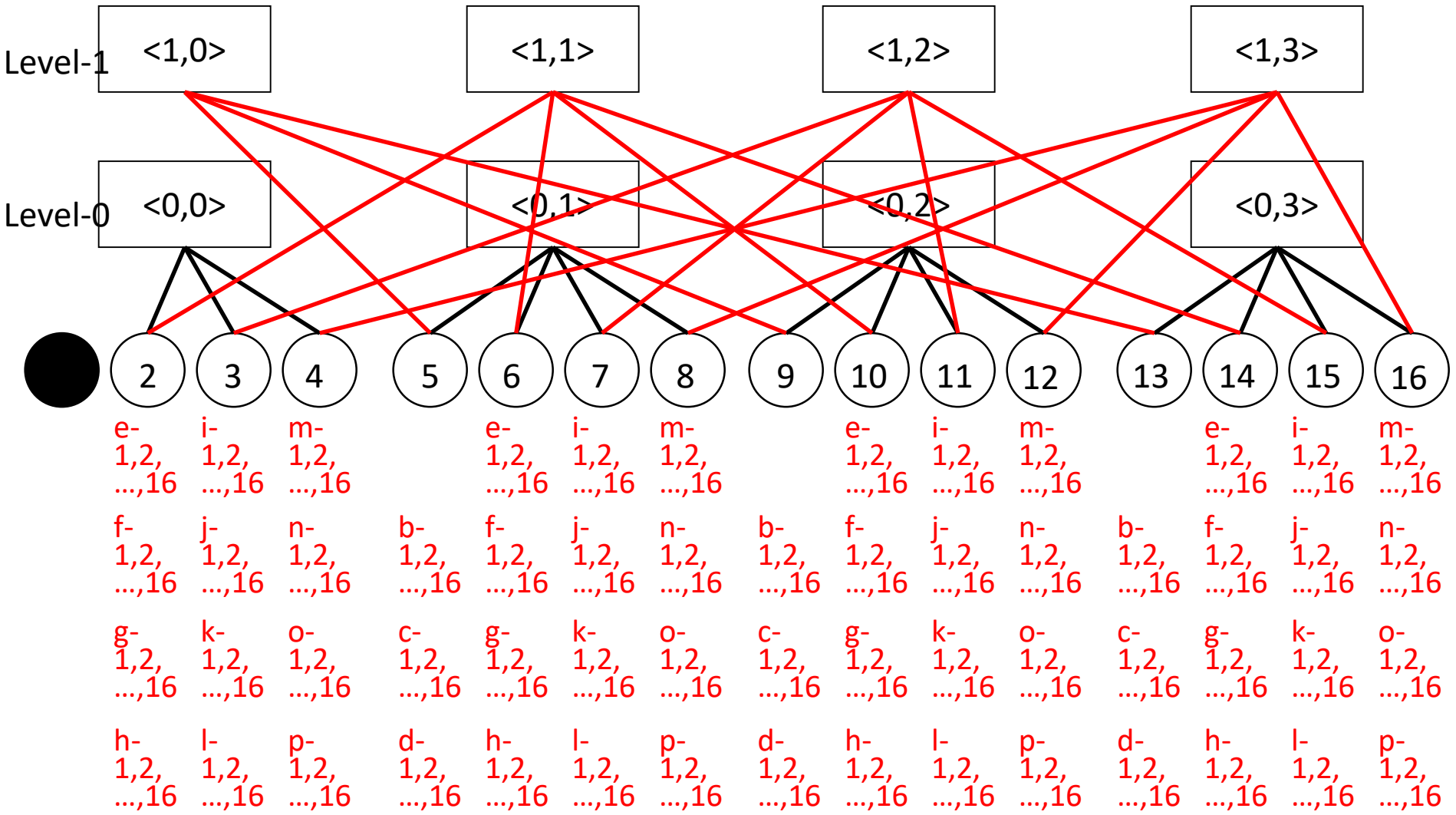
c-2,3,4 takes the detour path of 3->11->9

C-5,9,13 takes the detour path of 9->11->3

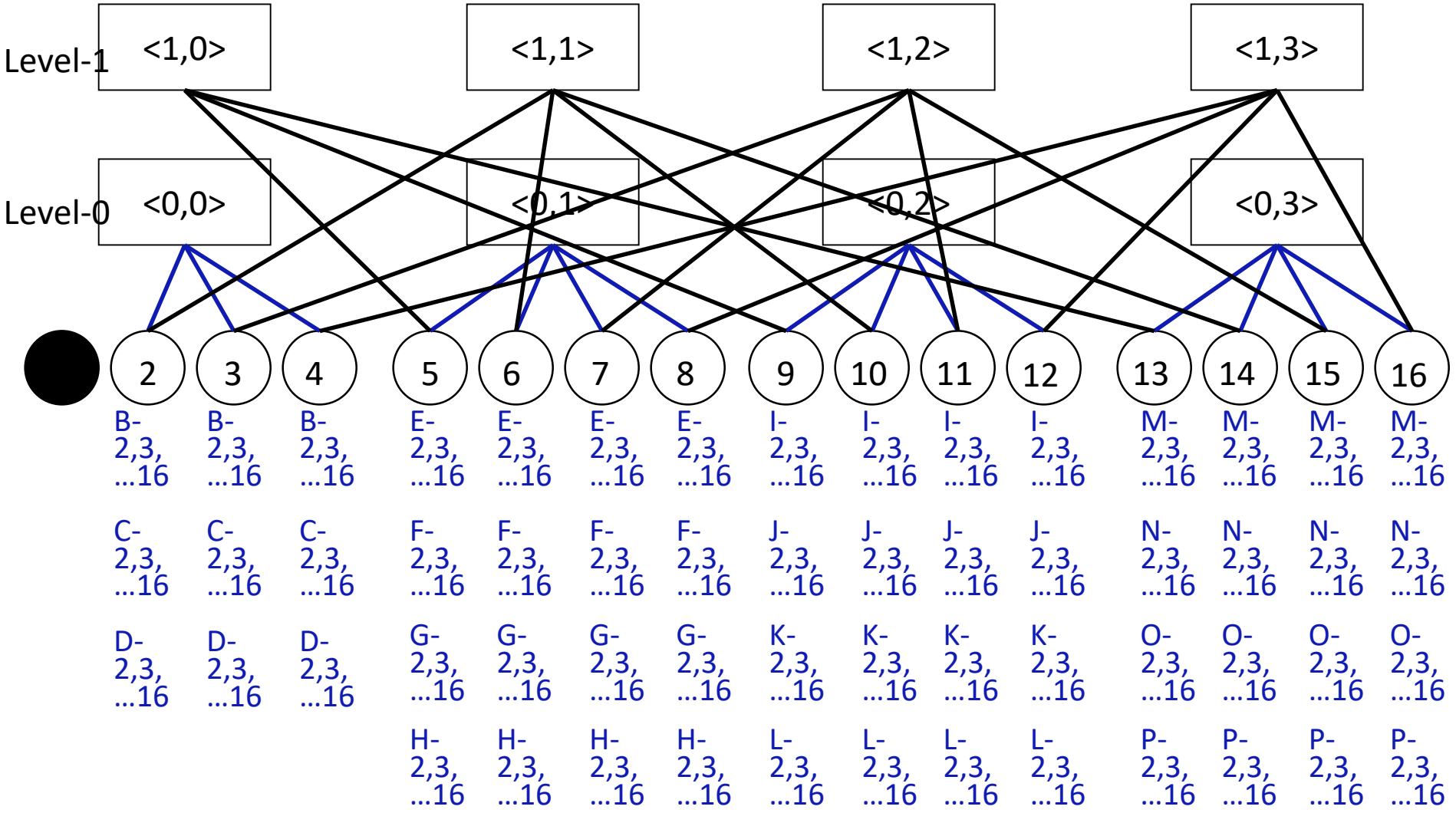
d-2,3,4 takes the detour path of 4->16->13

D-5,9,13 takes the detour path of 13->16->4

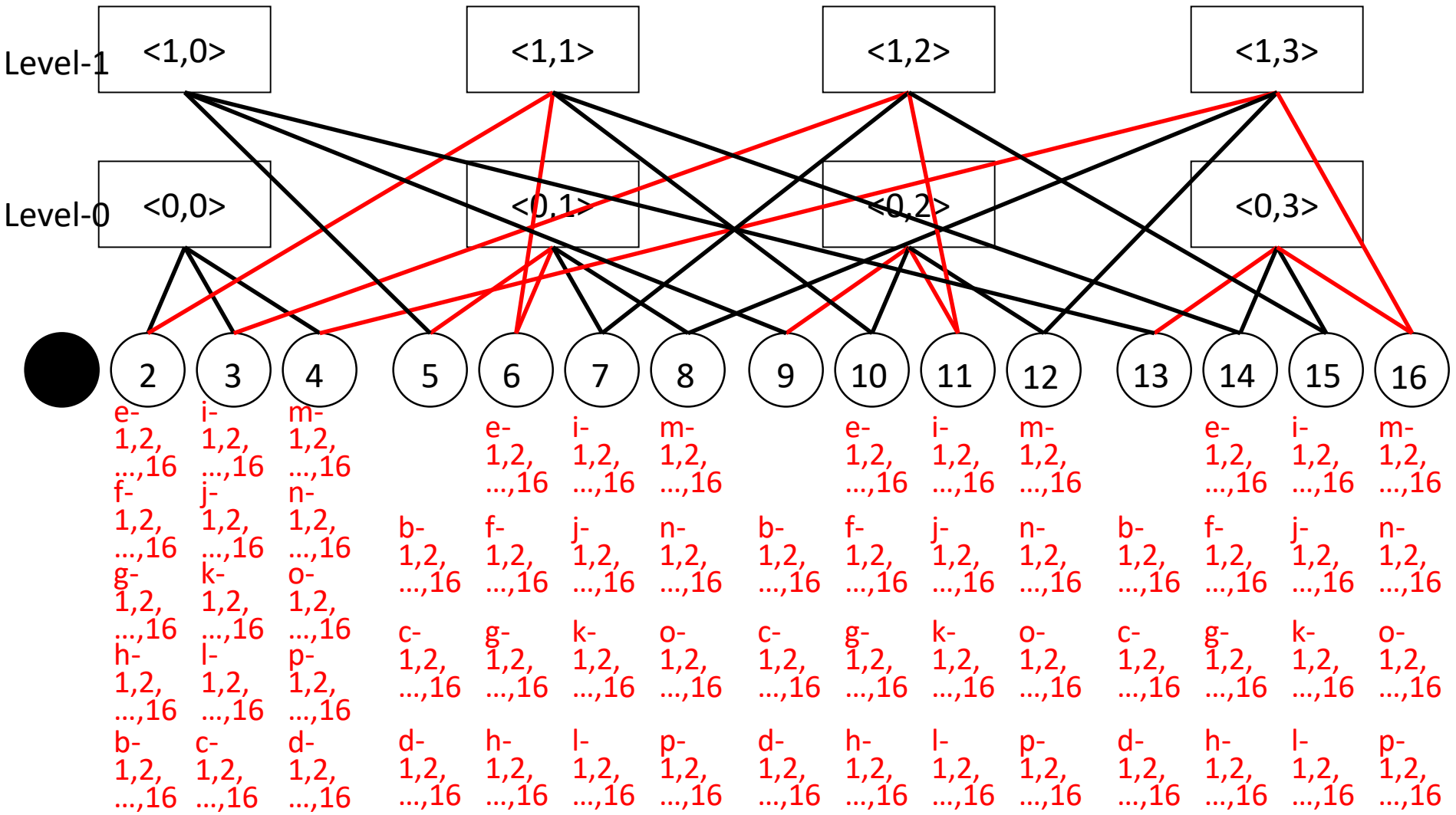
BML Server Failure: t=19*T/30



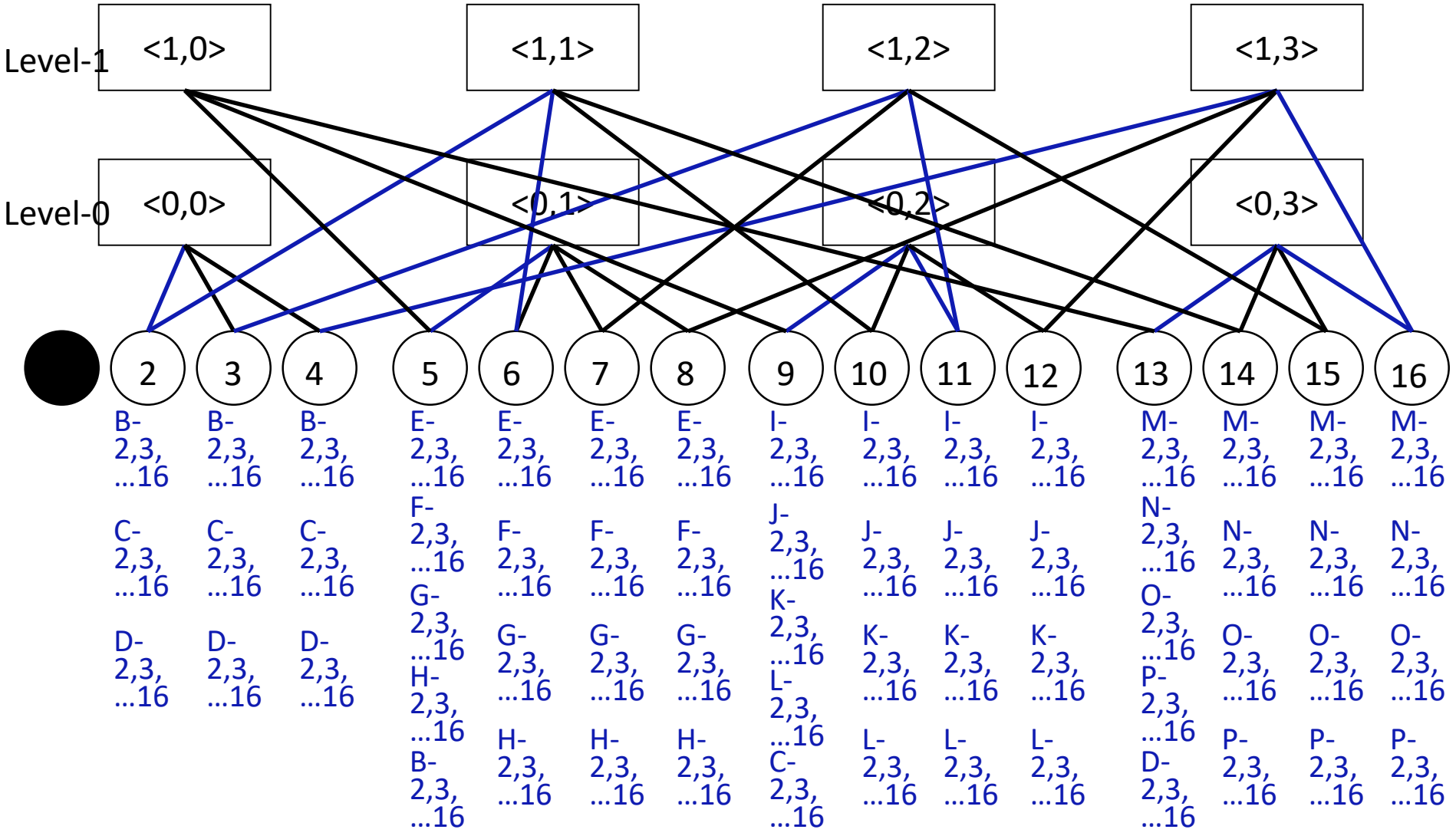
BML Server Failure: t=19*T/30



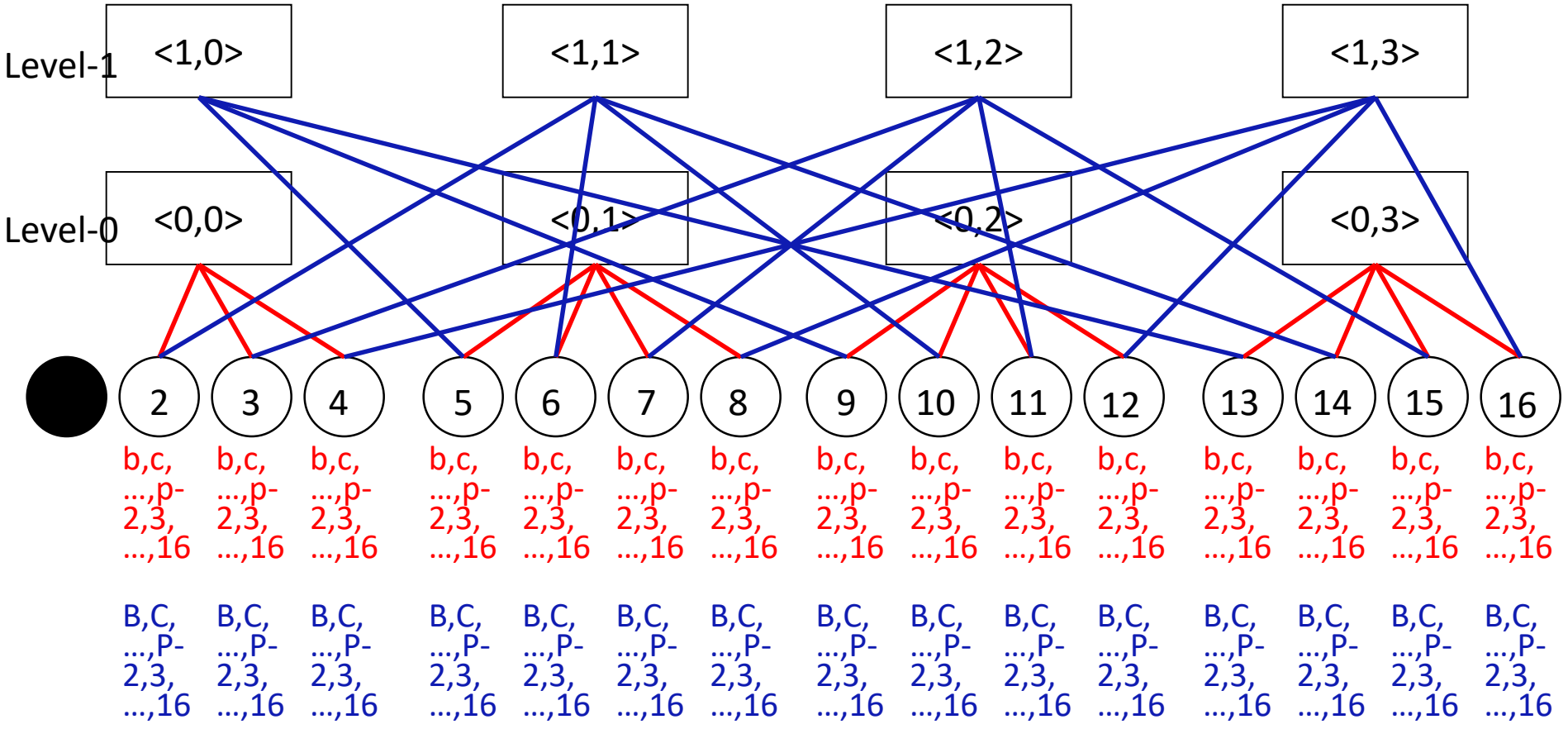
BML Server Failure: t=20*T/30



BML Server Failure: t=20*T/30



BML Server Failure: t=32*T/30



BCube/BML容错性强

- For BCube(m,2), the parameter update time when one server fails is
  - ✓$m^2*T/(m^2-1)$
  - ✓Compared with $(m^2-1)*T/m^2$ in complete BML
- For BCube(m,k), the parameter update time when one server fails is

$$\frac{2t}{k}*\left[\frac{n^{k-1}(n-1)}{n^k-1}+\frac{n^{k-1}-1}{n^k-1}+\frac{1}{(n^k-1)(n-1)}*\left(\frac{n^k-n}{n-1}-k+1\right)\right]$$