量子程序并行化执行

组长：叶之帆
组员：孙晨寅，吴浩阳
2021年3月11日

CONTENT

01.项目背景

现有的量子程序执行模型，项目动机

02.物理模型

对量子机器各个操作代价的抽象

03.重排算法

在全连通假设下，通过重排算法最大化并行度

04.非全联通模型

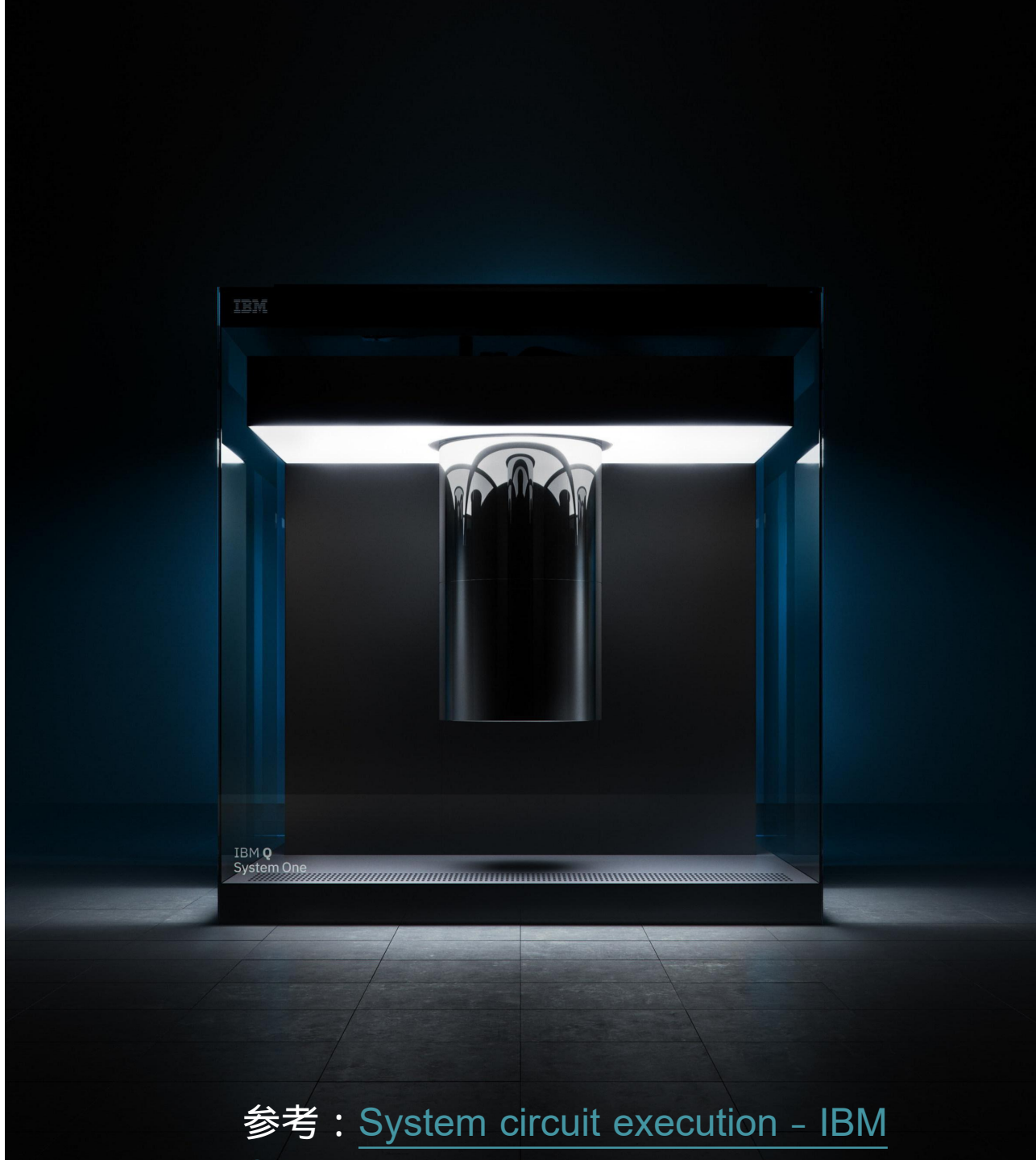
非全联通假设下，最大化并行度的执行模型

05.算法测试

测试算法执行效果

01.项目背景

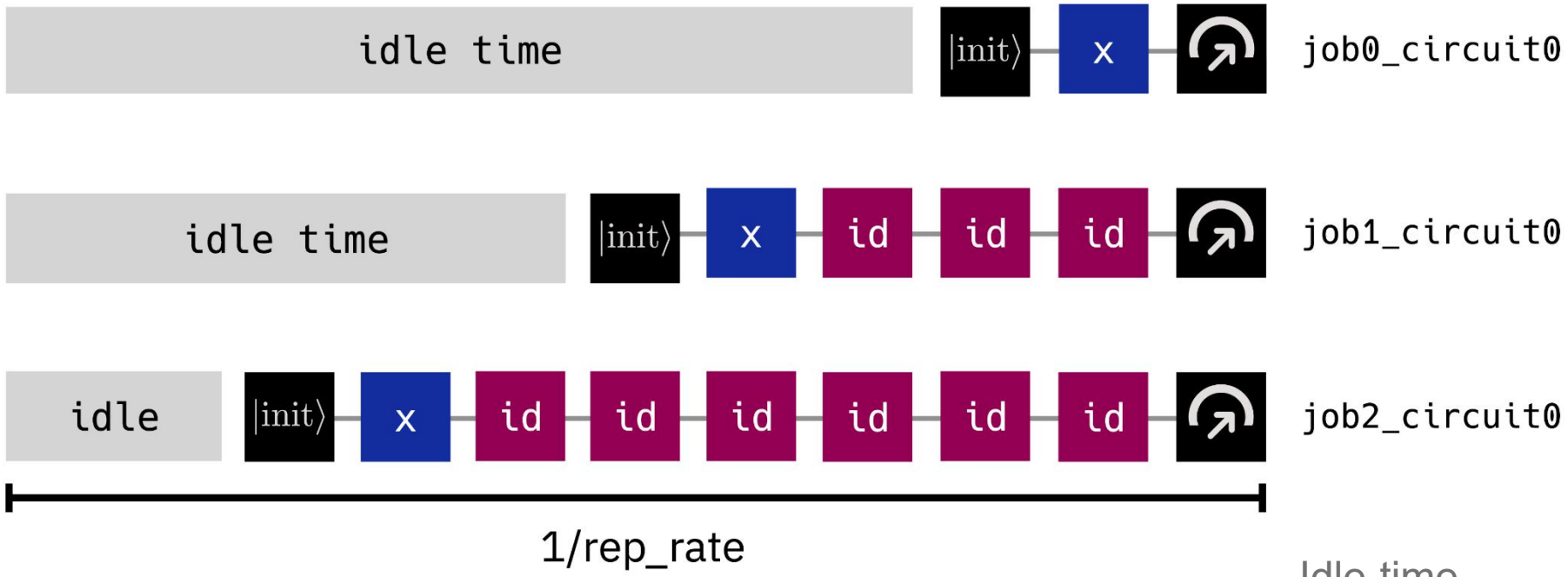
1. IBM-Q运行模型
2. 并行化的执行模型



参考：System circuit execution - IBM

IBM-Q执行模型：对单个任务

Individual circuit execution



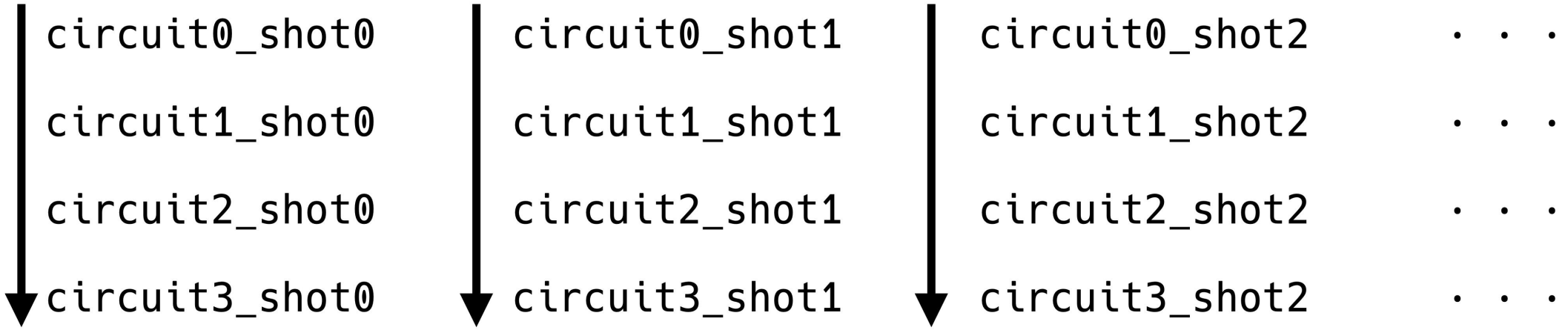
Idle time

IBM-Q执行模型：对单个任务

rep_rate: 量子机器的系统速率，一个 rep_rate 时间值内仅允许**执行一次量子程序**

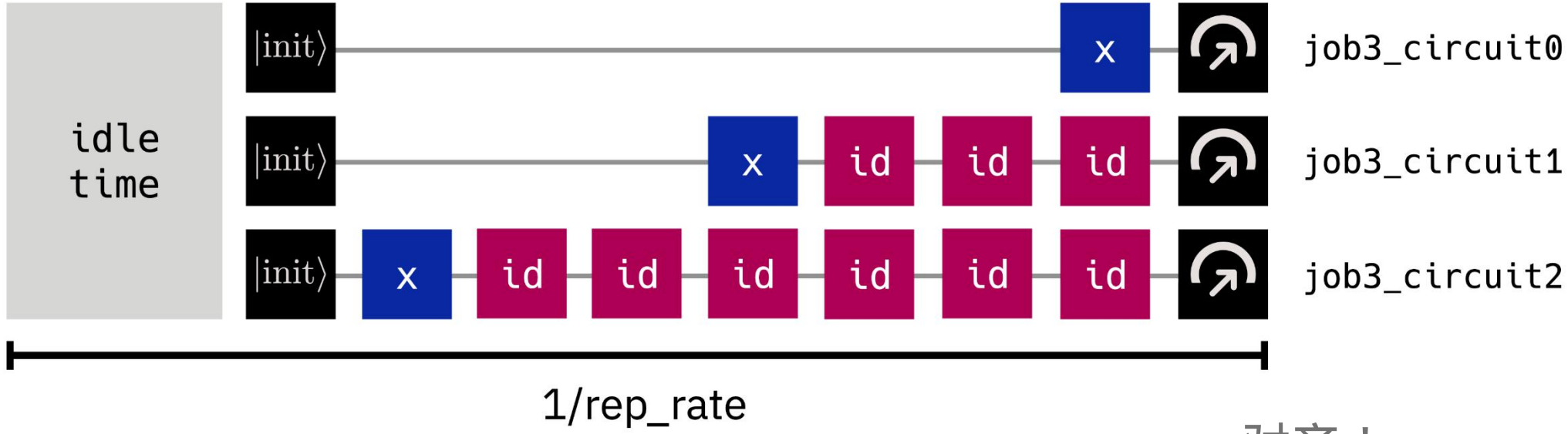
idle_time：量子程序的执行时间并不需要 rep_rate 那么长，则浪费掉了 idle_time 这么长的时间

IBM-Q执行模型：对批量提交的任务



IBM-Q执行模型：对批量提交的任务

Batched circuit execution



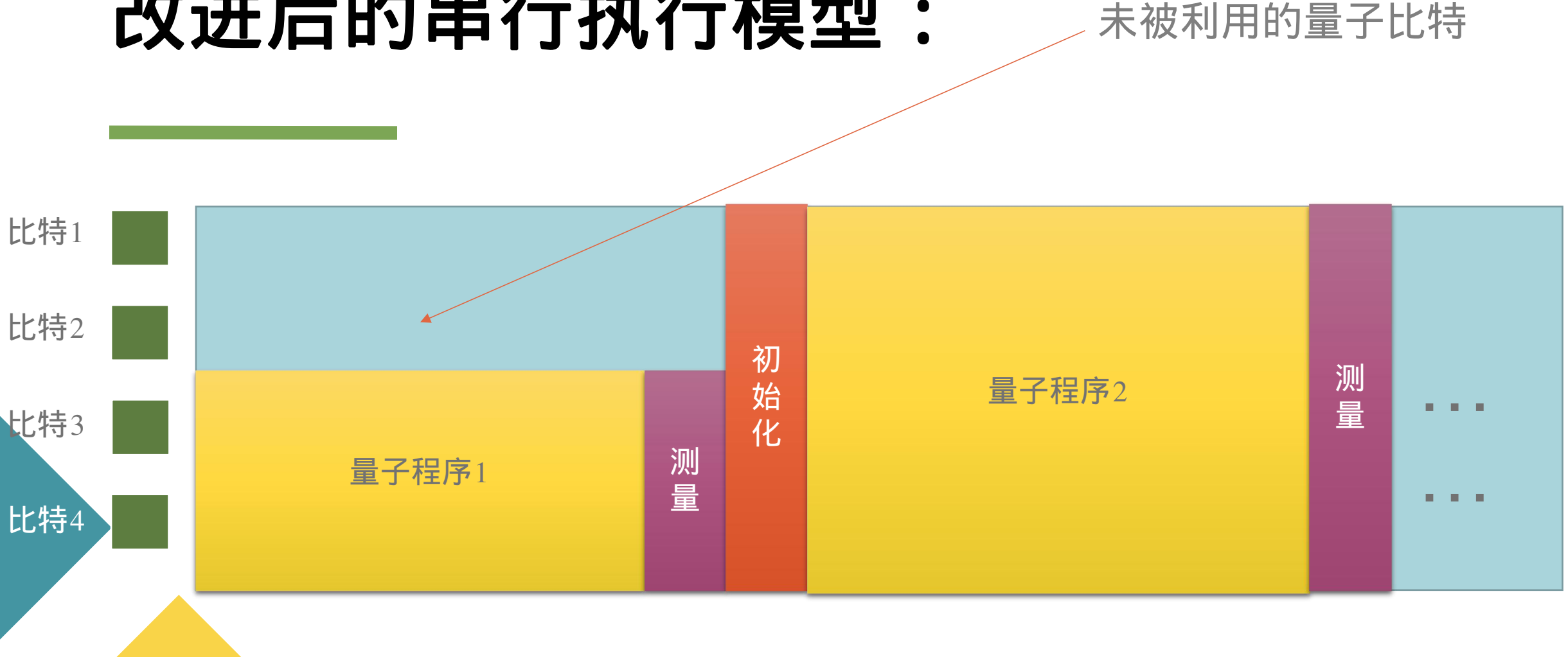
对齐！

IBM-Q的一些改进：

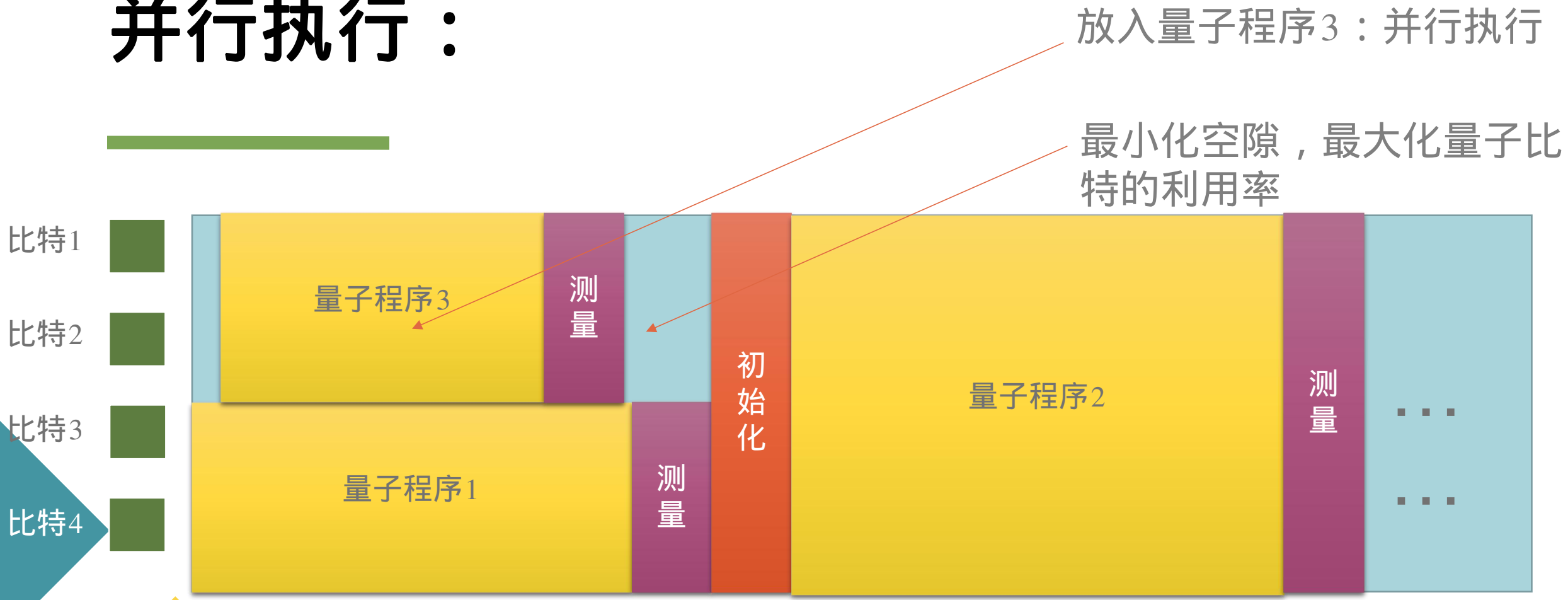
1. 允许动态调整 rep rate => 减少 idle time
2. 允许用户在任意时刻使用reset门和measure门 => 初始化时机和测量时机不必对齐

How to utilize the dynamic reset & measure gate ?

改进后的串行执行模型：



并行执行：


放入量子程序3：并行执行
最小化空隙，最大化量子比特的利用率
比特1
比特2
比特3
比特4
量子程序3
测量
量子程序1
测量
初始化
量子程序2
测量
. . .
. . .

02.物理模型

1. 量子门操作的耗时
2. reset与measure操作的耗时

量子门操作耗时：

超导体系：

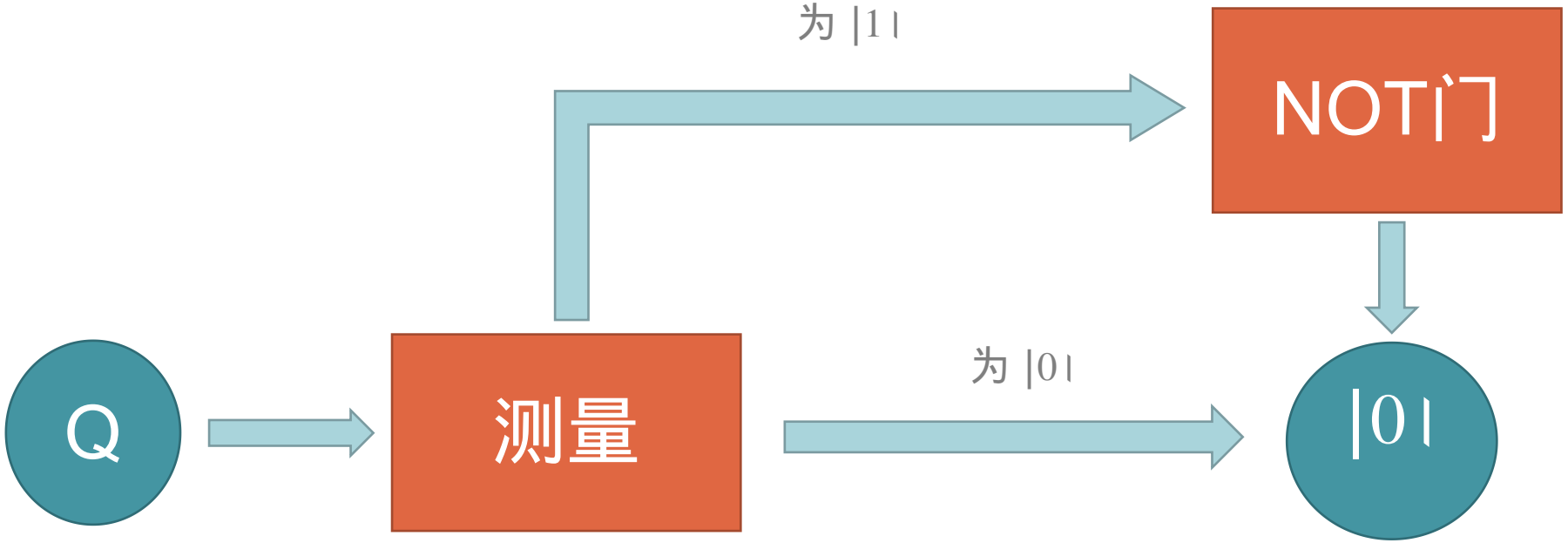
单量子比特门(Rx, Ry, Rz)：20ns

双量子比特门(CNOT门)：40ns

测量时间(Measure操作)：~20us

详细归纳过程：参见孙晨寅同学的调研报告

Reset门的实现方式：有条件的NOT门



Reset门的耗时：

Measure耗时 + 控制电路耗时 + NOT门耗时

≈ 20us

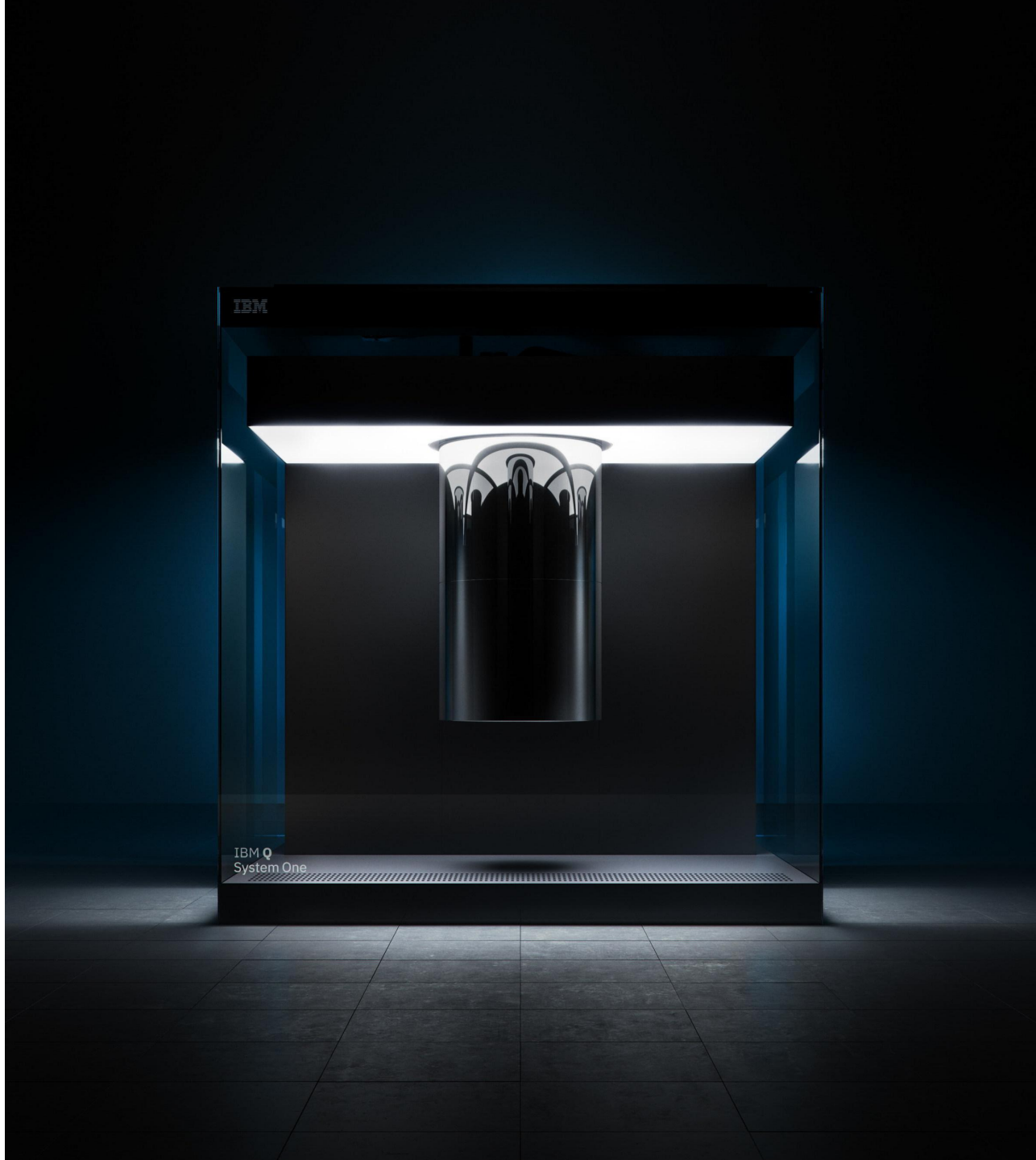
耗时抽象：

记单量子比特门的耗时为Π：

	耗时	抽象
单量子比特门	20ns	Π
双量子比特门	40ns	2Π
Measure门/Reset门	20us	1000Π

03.重排算法

在全连通假设下，通过重排算法最大化量子比特利用效率



重排算法：

批量提交

- 批量提交量子程序（每个量子程序的OPEN-QASM代码，执行次数）

逐个分析

- 针对提交的每个量子程序，分析其Open-QASM代码，提取特性（占用的量子比特数，执行耗时）

重排*

- 选定量子程序的执行顺序（每个量子程序每次执行的开始时机），从而最大化量子比特的利用率

*：由于存在全连通假设，重排算法唯一的约束是“任一时刻被占用的量子比特数不得超过量子比特总数”

一种贪婪算法：

while(等待执行的量子程序数>0)

选择**占用量子比特数最多且满足约束***的量子程序q_i进行执行；

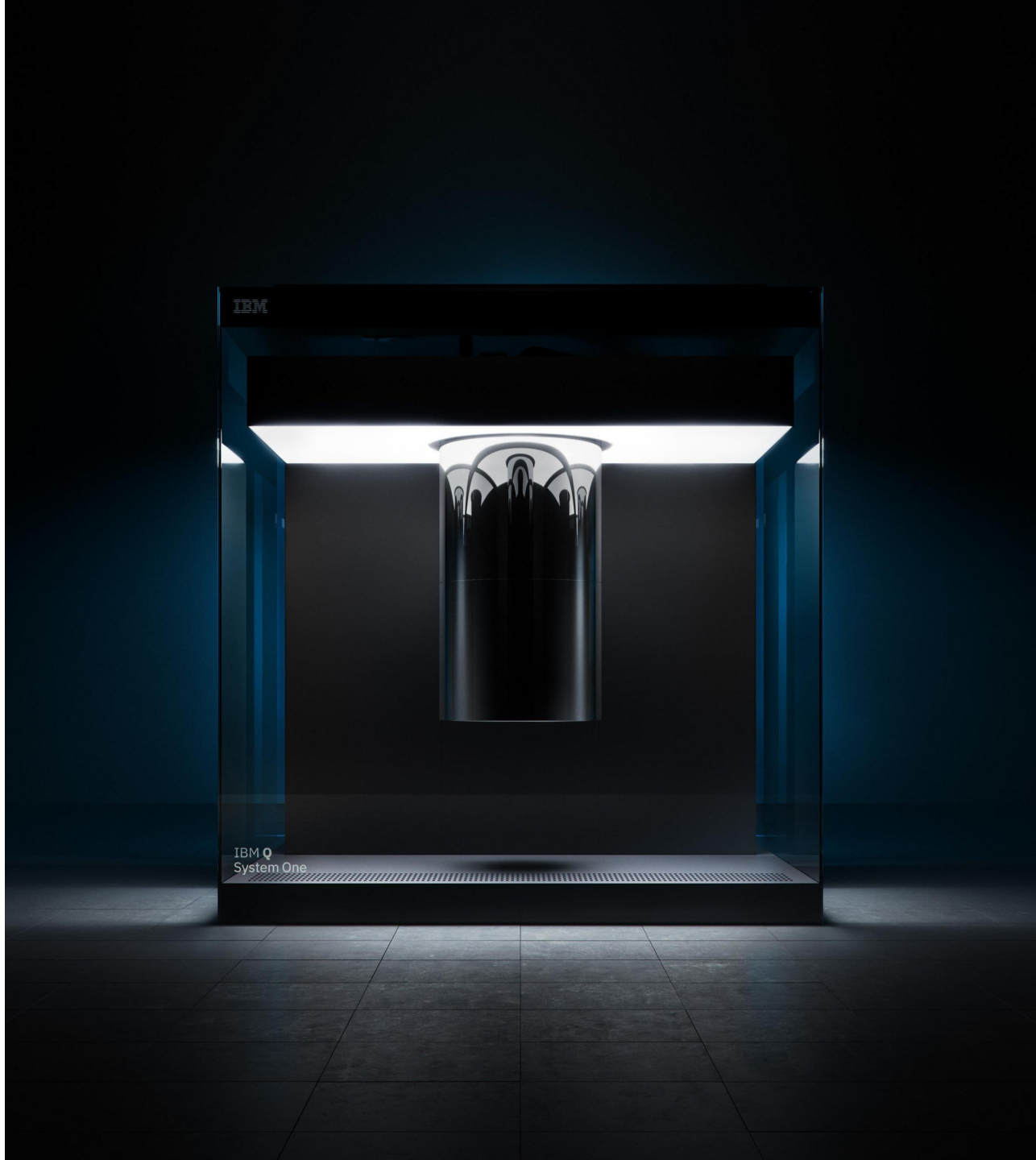
空闲量子比特数 -= n_i ；

T_i 时间后释放这些量子比特；

……

04.并行化映射算法

非全联通假设下，需要考虑底层量子比特的联通性



量子比特映射算法：

CNOT门 as an example:

CNOT门只作用于相邻的量子比特；

=>需要插入一系列SWAP门使得两个量子比特相邻；

=>增加耗时，增加错误率

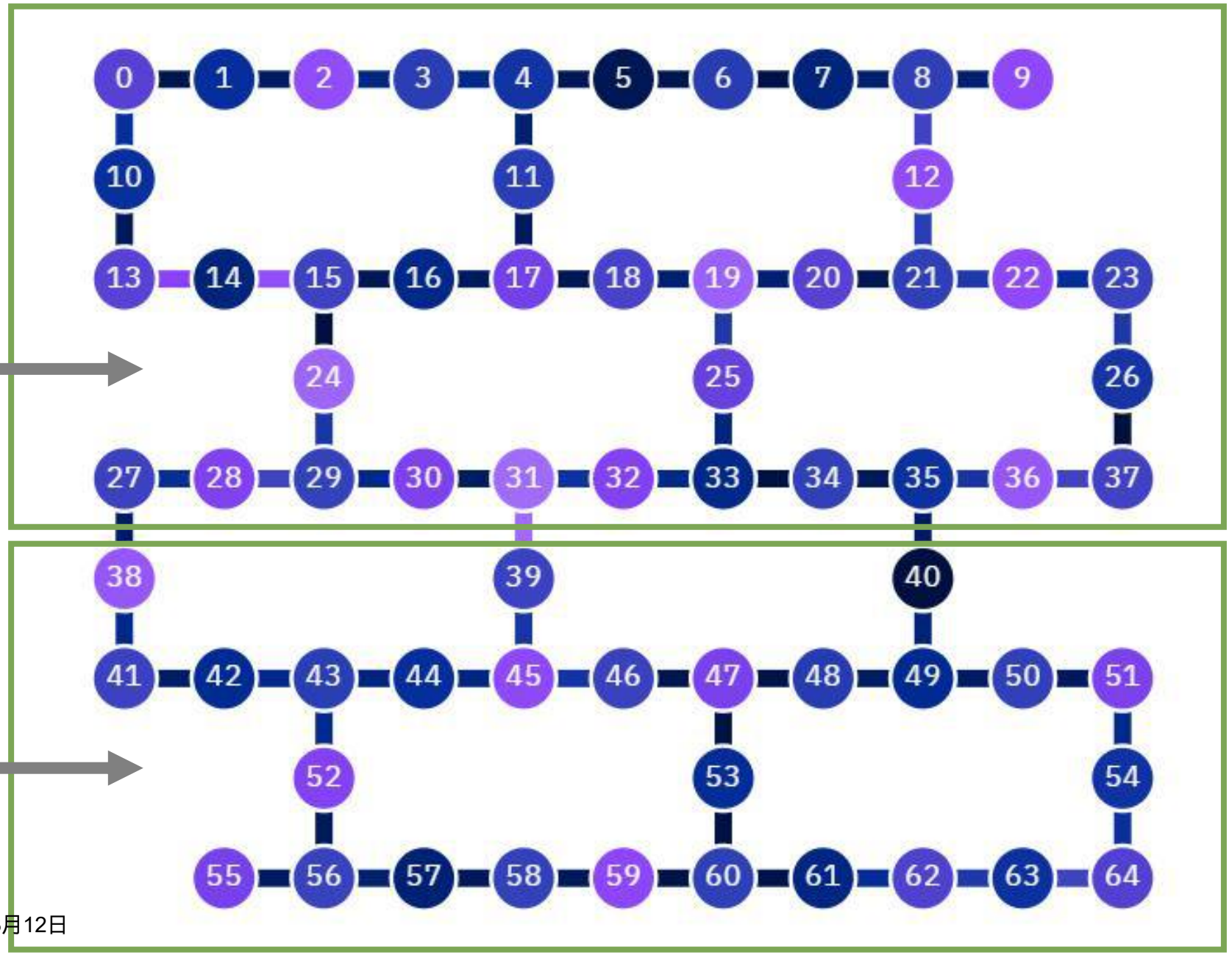
=>映射算法（选择逻辑量子比特到物理量子比特的映射），最小化实际执行时间

仅考虑最小化单个量子程序的执行时间

分区：

分区1

分区2


0 1 2 3 4 5 6 7 8 9
10 11 12
13 14 15 16 17 18 19 20 21 22 23
24 25 26
27 28 29 30 31 32 33 34 35 36 37
38 39 40
41 42 43 44 45 46 47 48 49 50 51
52 53 54
55 56 57 58 59 60 61 62 63 64

基于分区的映射：

将量子机器划分成一系列分区；

每个量子程序在一个或多个分区上执行；

不占用相同分区的量子程序一定可以并行执行

基于分区的并行化映射算法：

批量提交

- 批量提交量子程序（每个量子程序的OPEN-QASM代码，执行次数）

映射到分区

- 针对提交的每个量子程序，选择一个合适的映射算法*，将量子程序映射到一个或多个分区（可以有多种映射到分区的方案），提取属性（占用的分区，执行时间）

排序算法**

- 选定量子程序的执行顺序（每个量子程序每次执行的开始时机），从而最大化量子比特的利用率

*：可以是任何量子比特映射算法，这里我们选用的是Q-CODAR算法

**：现在的约束：同一时刻执行的量子程序不能占用重叠的分区

基于划分的并行化映射算法：

while(等待执行的量子程序数>0)

选择**占用分区数最多且满足约束****的量子程序q_i进行执行；

相应分区标记为非空闲；

T_i 时间后释放这些分区；

……

基于分区的映射：

简单快速

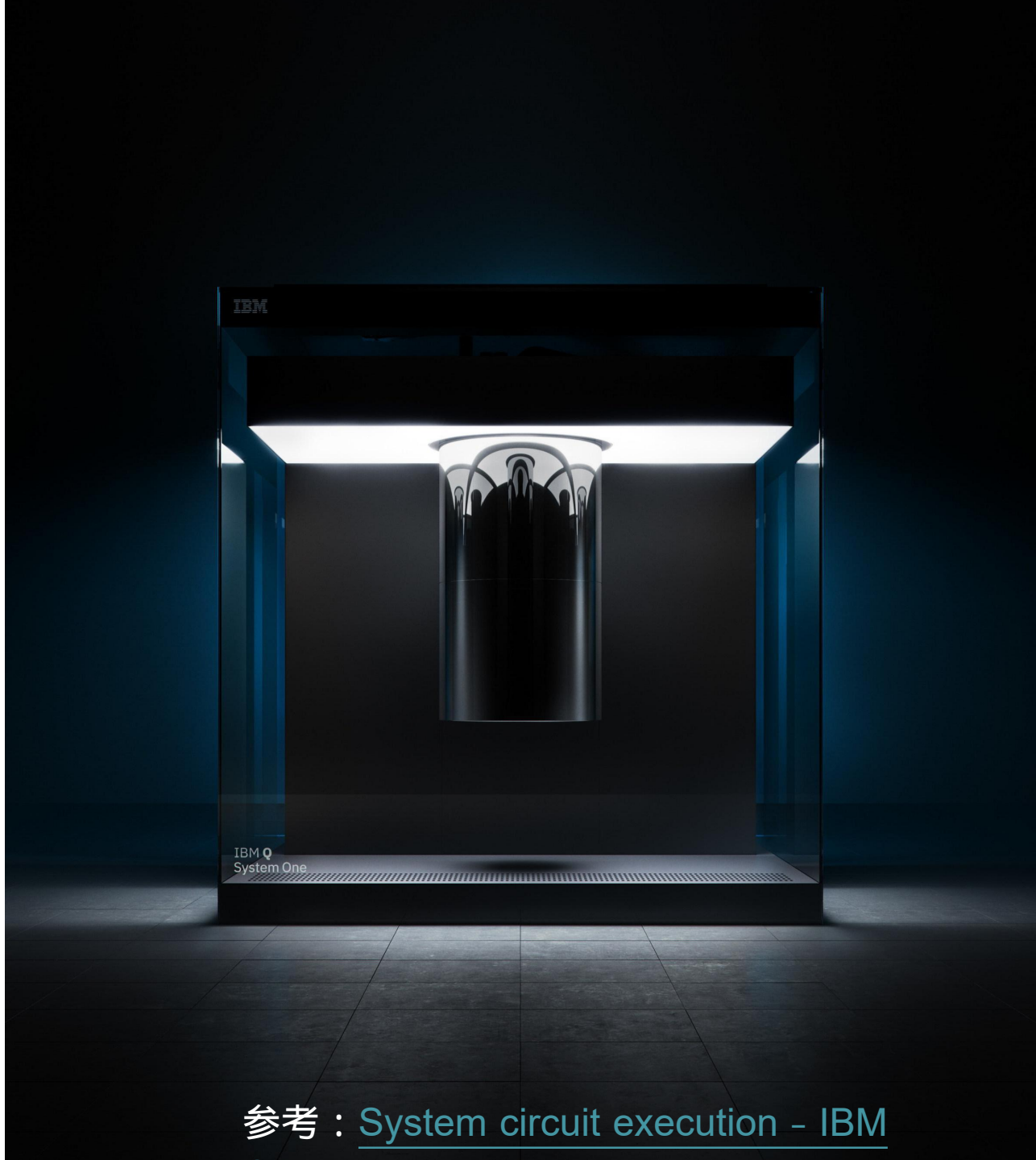
容易扩展到不同的量子机器上

可以复用针对单量子程序的映射算法（如Q-CODAR）

一次映射，多次使用

05.算法测试

1. 重排算法
2. 并行化映射算法



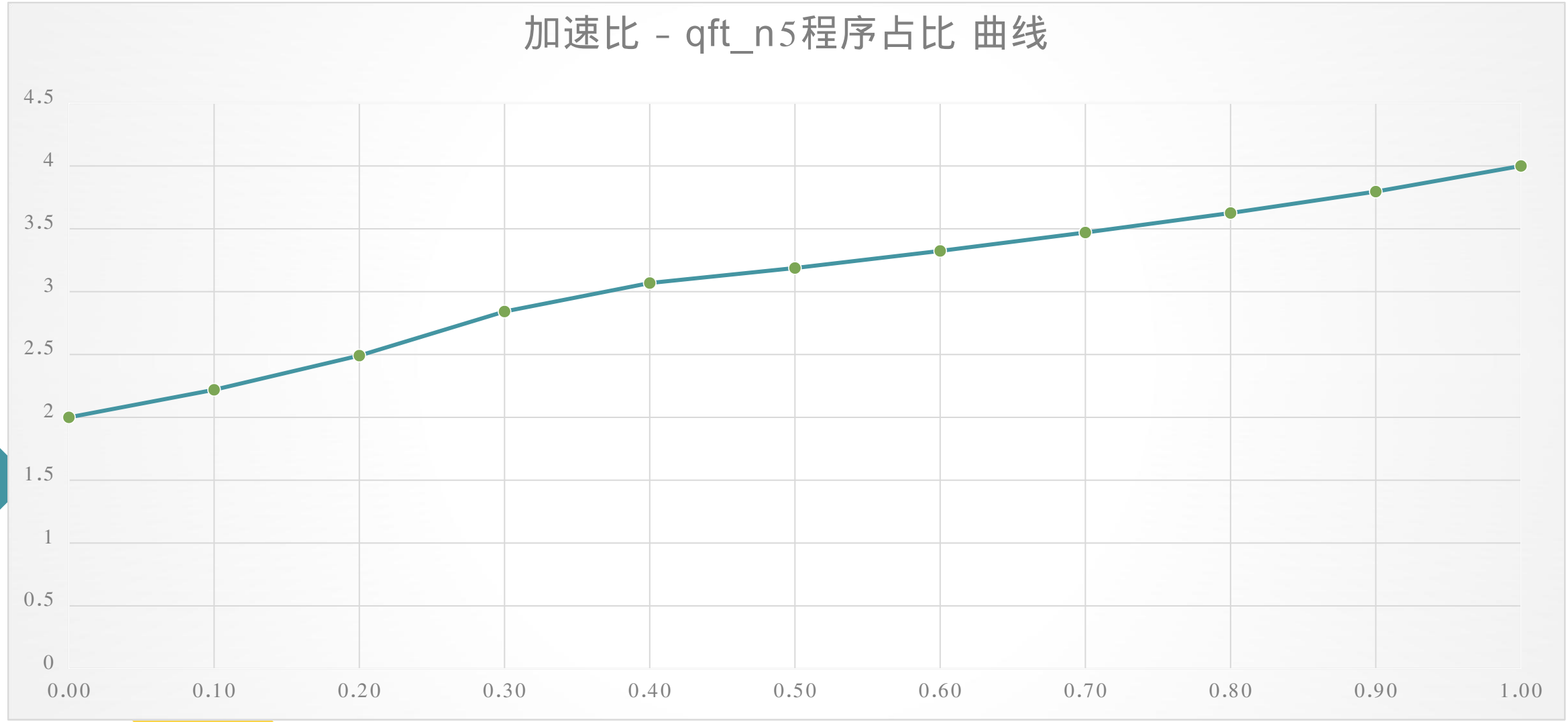
参考：System circuit execution - IBM

测试结果：重排算法

具有20个量子比特的全连通机器

提交两种量子程序（QFT_N5 和 QFT_N7），总计执行1000次

测试结果：重排算法

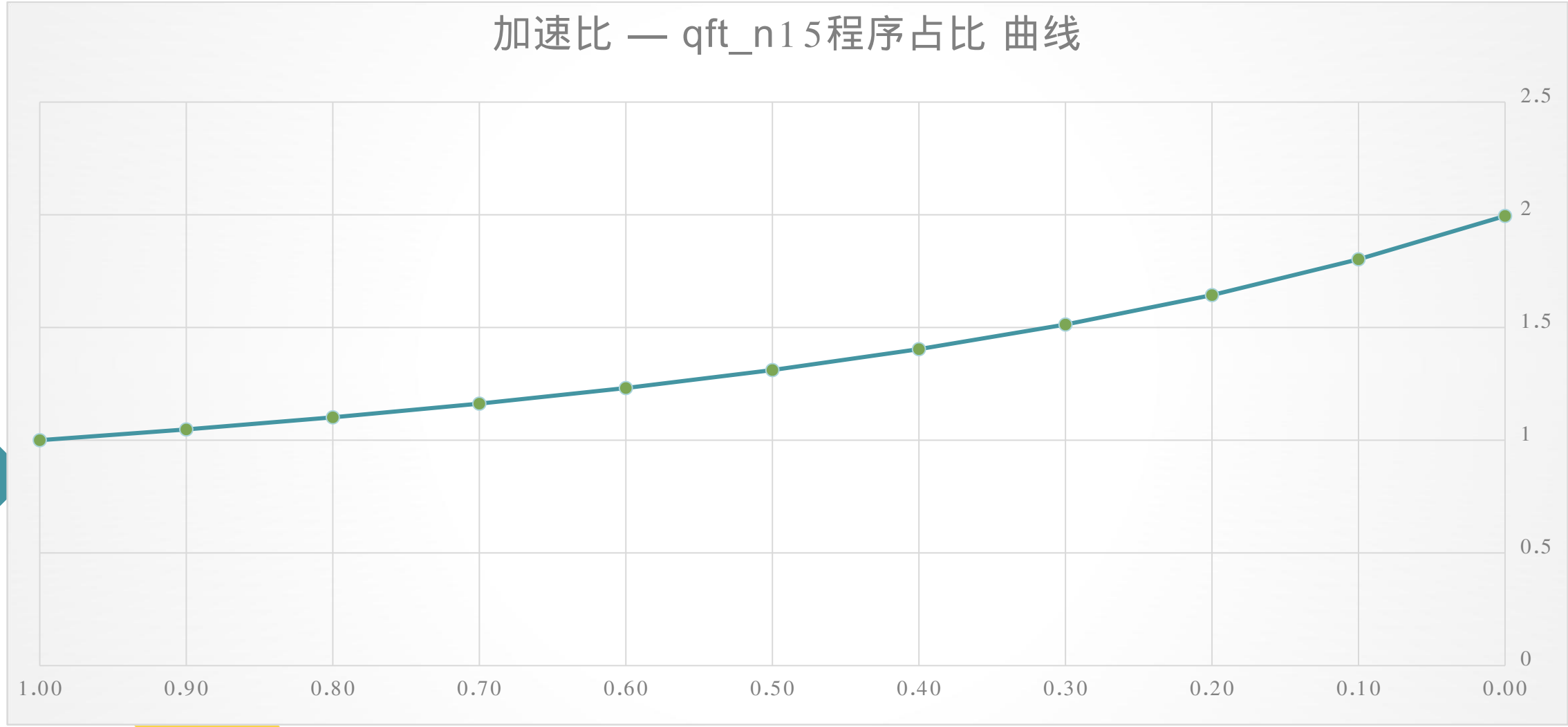

加速比 - qft_n5程序占比 曲线
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00

测试结果：并行化映射算法

将IBM-TOKYO机器分区（共20量子比特，0~9号量子比特分成一个分区，10~19号量子比特分入另一个分区）

提交两种量子程序（QFT_N7 和 QFT_N15），总计执行1000次

测试结果：并行化映射算法



分工

叶之帆

建模；

提出算法；

协助算法测试；

参与讨论

孙晨寅

相关物理知识的调研与建模；

参与讨论

吴浩洋

算法测试；

参与讨论

谢谢