

# 分布数据一致性技术研究

## (博士学位论文答辩报告)

答辩人: 魏恒峰

导师: 吕建教授、黄宇副教授

南京大学软件所

2016 年 11 月 14 日



# 分布数据一致性技术研究

① 研究背景

② 研究问题

③ 技术框架

④ 相关工作

⑤ 本文工作

⑥ 总结与展望

# 分布数据一致性技术研究

① 研究背景

② 研究问题

③ 技术框架

④ 相关工作

⑤ 本文工作

⑥ 总结与展望

# 分布式应用



新浪微博社交应用<sup>1</sup>:

- ▶ 日均用户近一亿名
- ▶ 日均消息近一亿条

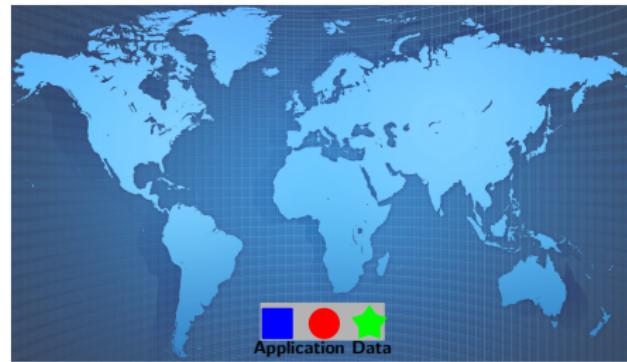
底层数据服务系统特性需求 ( $H^3L$ ):

- ▶ 低延迟, 高可用性 (4 个 9<sup>2</sup>)
- ▶ 高容错性, 高可扩展性

<sup>1</sup> 2015 第三季度; 数据来自 China Internet Watch.

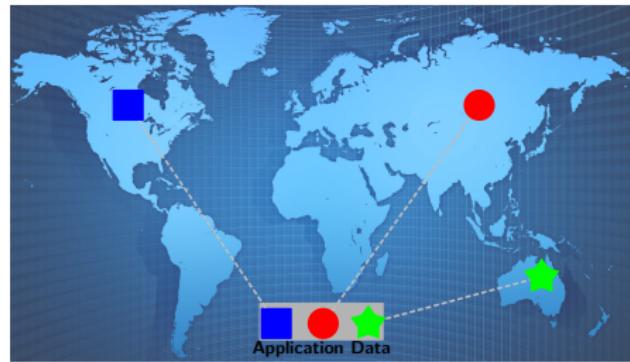
<sup>2</sup> 数据来自 InfoQ.

# 分布数据



应用数据:

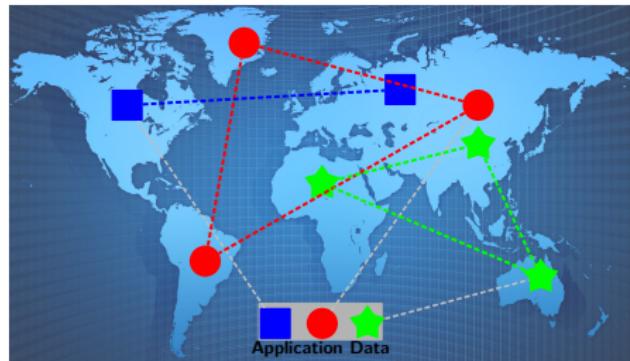
# 分布数据



应用数据:

1. 分区 ([partition](#)): 水平扩展

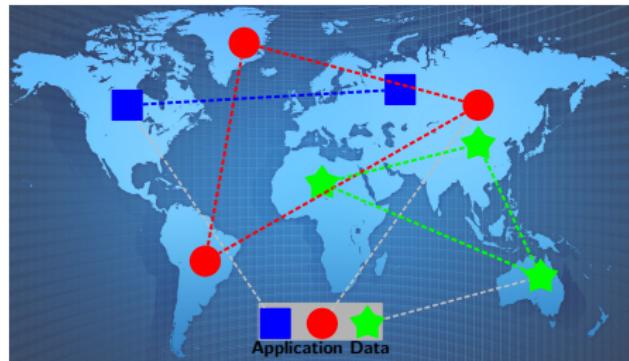
# 分布数据



应用数据:

1. 分区 (*partition*): 水平扩展
2. 副本 (*replication*): 就近访问, 容灾备份

# 分布数据



分布数据 (distributed data):

1. 分区 (partition): 水平扩展
2. 副本 (replication): 就近访问, 容灾备份

# 分布数据一致性技术研究

① 研究背景

② 研究问题

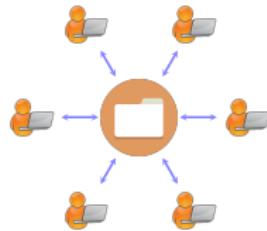
③ 技术框架

④ 相关工作

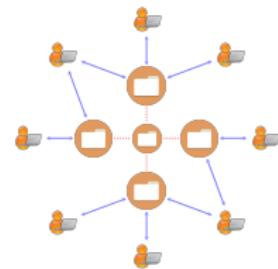
⑤ 本文工作

⑥ 总结与展望

# 分布数据一致性问题



共享数据



分布数据

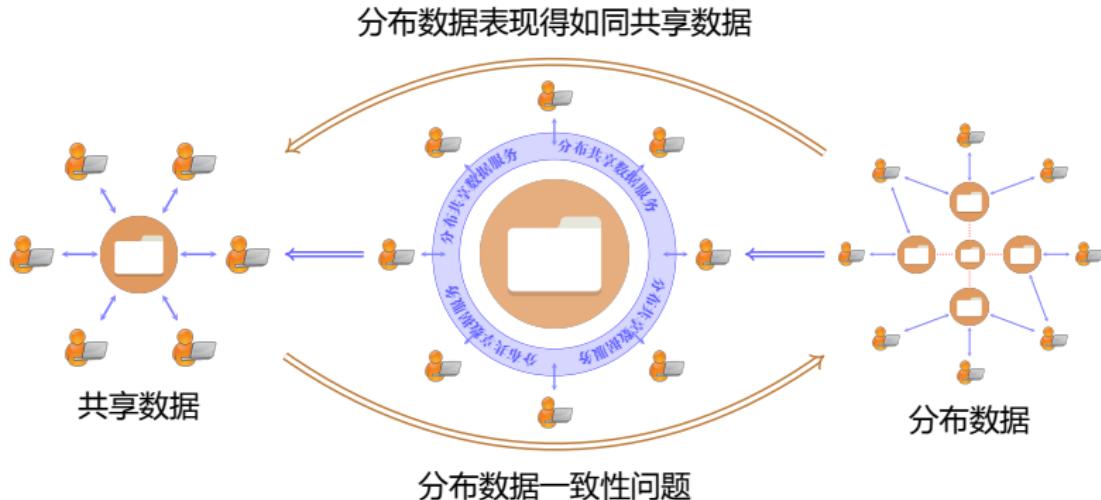
# 分布数据一致性问题



分布数据一致性问题 (应用视角):

- ▶ 分布数据的语义是什么?
- ▶ 如何“方便”地使用分布数据?

# 分布数据一致性问题



## 分布数据一致性问题 (应用视角):

- ▶ 分布数据的语义是什么?
- ▶ 如何“方便”地使用分布数据?

## 分布共享数据服务 (中间件):

- ▶ 屏蔽分布数据细节
- ▶ 提供共享数据抽象

# 分布数据一致性问题

理想情况:

- ▶ one-size-fits-all  
一致性模型
- ▶ 始终观察到最新副本

没有分布数据一致性问题

# 分布数据一致性问题

理想情况:

- ▶ one-size-fits-all  
一致性模型
- ▶ 始终观察到最新副本

没有分布数据一致性问题

实际情况 (tradeoffs):

$H^3L$   
Partition-tolerance  
Convergence  
Churn  
...

Consistency



# 分布数据一致性问题

理想情况:

- ▶ one-size-fits-all  
一致性模型
- ▶ 始终观察到最新副本

~~没有分布数据一致性问题~~

实际情况 (tradeoffs):

$H^3L$   
 Partition-tolerance  
 Convergence  
 Churn  
 ...

Consistency



在分布式系统中,  
 以数据一致性为核心的权衡 [Guerraoui@TCDE'16] 使得  
 分布数据一致性问题成为分布共享数据服务中的核心难题.

# 分布数据一致性问题

论文研究动机：

考虑到上述权衡，  
面向大规模分布式系统的  
分布数据一致性理论应体现什么特性？

# 分布数据一致性问题

论文研究动机：

考虑到上述权衡，  
面向大规模分布式系统的  
分布数据一致性理论应体现什么特性？

分布式系统设计与其应用有哪些特点，  
对分布数据一致性理论有何影响？

# 多处理器系统领域中的数据一致性理论

(分布) 数据一致性问题是多处理器系统领域中的传统问题:

## How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs

LESLIE LAMPORT

*Abstract*—Many large sequential computers execute operations in a different order than is specified by the program. A correct execution is achieved if the results produced are the same as would be produced by executing the program steps in order. For a multiprocessor computer, such a correct execution by each processor does not guarantee the correct execution of the entire program. Additional conditions are given which do guarantee that a computer correctly executes multiprocess programs.

(a) Sequential Consistency [[Lamport@TC'79](#)].



## Shared Memory Consistency Models: A Tutorial

The shared memory programming model has several advantages over the message passing model. In particular, it simplifies data partitioning and dynamic load distribution. Shared memory systems are therefore gaining wide acceptance for both technical and commercial applications.

To write correct and efficient shared memory programs, programmers need a precise notion of shared memory semantics. For example, in the

(b) Tutorial [[Adve@IEEE Computer'96](#)].

# 多处理器系统领域中的数据一致性理论

“以程序为导向、强调正确性”

“相对于 SC (Sequential Consistency)” 的正确性<sup>3</sup> [Adve@ISCA'90]:

一致性: 一致性模型 + 编程模型

程序: 遵循预定规则

系统: 提供 SC 保证

<sup>3</sup> 术语称为 SCNF: Sequential Consistency Normal Form.

# 多处理器系统领域中的数据一致性理论

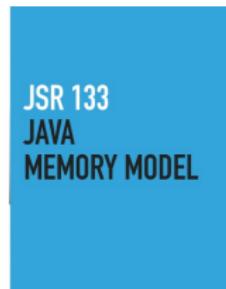
“以程序为导向、强调正确性”

“相对于 SC (Sequential Consistency)” 的正确性<sup>3</sup> [Adve@ISCA'90]:

一致性: 一致性模型 + 编程模型

程序: 遵循预定规则

系统: 提供 SC 保证



<sup>3</sup> 术语称为 SCNF: Sequential Consistency Normal Form.

# 多处理器系统领域中的数据一致性理论

“以程序为导向、强调正确性”

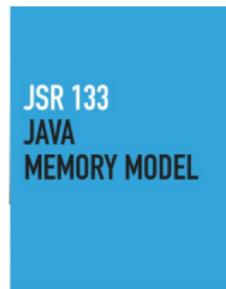
“相对于 SC (Sequential Consistency)” 的正确性<sup>3</sup> [Adve@ISCA'90]:

一致性: 一致性模型 + 编程模型

程序: 遵循预定规则

系统: 提供 SC 保证

权衡: 易编程性 vs. 性能



<sup>3</sup> 术语称为 SCNF: Sequential Consistency Normal Form.

# 分布式系统领域中的数据一致性理论

分布数据一致性问题是分布式系统设计的核心问题之一：

Managing Update Conflicts in Bayou,  
a Weakly Connected Replicated Storage System

Douglas B. Terry, Marvin M. Theimer, Karin Petersen, Alan J. Demers,  
Mike J. Spreitzer and Carl H. Hauser

Computer Science Laboratory  
Xerox Palo Alto Research Center  
Palo Alto, California 94304 U.S.A.

(a) Eventual Consistency [Terry@SOSP'95].



(b) 分布式存储系统 (左: 开源; 右: 商用).

# 分布式系统领域中的数据一致性理论

更复杂的权衡:

$H^3L$   
Partition-tolerance  
Convergence  
Churn                      **Consistency**  
...



# 分布式系统领域中的数据一致性理论

更复杂的权衡:

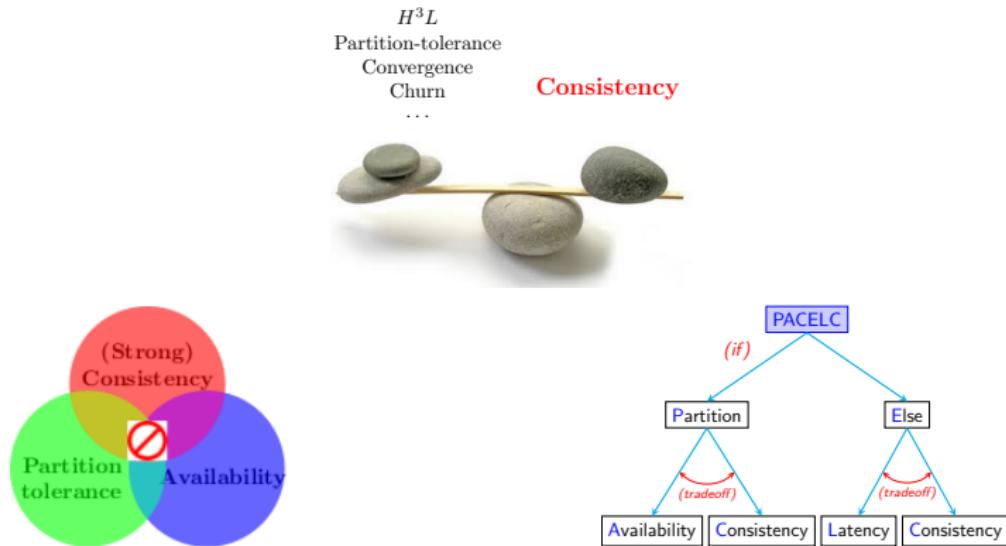


图: CAP 定理 [Brewer@PODC'00]

图: PACELC 权衡 [Abadi@IEEE Computer'12]

# 分布式系统领域中的数据一致性理论

更复杂的权衡  $\Rightarrow$  一致性理论不追求“相对于 SC”的正确性

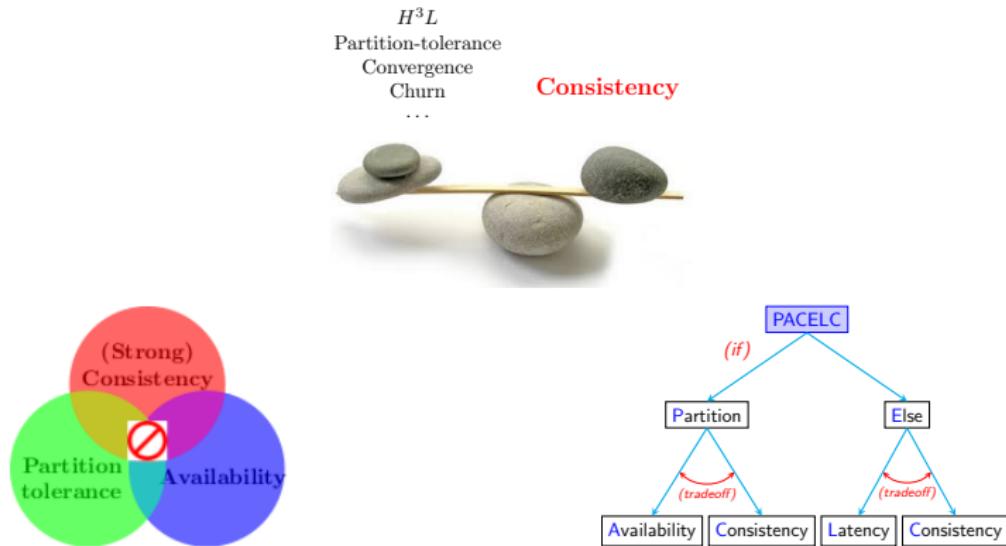


图: CAP 定理 [Brewer@PODC'00]

图: PACELC 权衡 [Abadi@IEEE Computer'12]

# 分布式系统领域中的数据一致性理论

更丰富的应用:



图: Cassandra 分布式存储系统.



图: Cassandra & DataStax 客户.

# 分布式系统领域中的数据一致性理论

更丰富的应用  $\Rightarrow$  一致性理论应体现应用的多样性



图: Cassandra 分布式存储系统.



图: Cassandra & DataStax 客户.

# 分布数据一致性问题研究理念 (I)

新平台, 新特点: 更复杂的权衡、更丰富的应用

# 分布数据一致性问题研究理念 (I)

新平台, 新特点: 更复杂的权衡、更丰富的应用

棒球赛应用一致性需求 [Terry@SOSP'13]:

记分员: 采用 `read-my-writes consistency` 更新比赛得分

电台记者: 采用 `monotonic reads consistency` 避免读到陈旧比分

# 分布数据一致性问题研究理念 (I)

新平台, 新特点: 更复杂的权衡、更丰富的应用

出租车实时位置查询一致性需求:

权衡: 为了保证低延迟, 允许违反 atomicity

有界: 所有读请求都要满足 2-atomicity

量化需求: 违反 atomicity 的读请求低于 1%

# 分布数据一致性问题研究理念 (I)

新平台, 新特点: 更复杂的权衡、更丰富的应用

需要更具“柔性”的数据一致性理论:

# 分布数据一致性问题研究理念 (I)

新平台, 新特点: 更复杂的权衡、更丰富的应用

需要更具“柔性”的数据一致性理论:

1. 多样化, 可调节
2. 精细化, 可度量

# 数据一致性问题研究理念 (II)

多样化: 从单一到融合 (mono- vs. multi-) [Terry@CACM'13]

- ▶ 融合强弱一致性: 不同操作, 不同一致性需求
- ▶ 融合一致与不一致: 容忍“有限度”的不一致



# 数据一致性问题研究理念 (II)

多样化: 从单一到融合 (mono- vs. multi-) [Terry@CACM'13]

- ▶ 融合强弱一致性: 不同操作, 不同一致性需求
- ▶ 融合一致与不一致: 容忍“有限度”的不一致



可调节: think *dynamically* [Terry@SOSP'13]

依据应用需求/系统状态调节数据一致性

# 数据一致性问题研究理念 (III)

精细化：从二元到连续谱 [Yu@TOCS'02]



# 数据一致性问题研究理念 (III)

精细化: 从二元到连续谱 [Yu@TOCS'02]



可度量: think *probabilistically* [Brewer@PODC'00]

量化系统执行, 后验系统对一致性的满足程度

# 数据一致性问题研究理念 (IV)

论文研究问题:

如何在大规模分布式系统中  
落实“多样化, 可调节; 精细化, 可度量”的  
数据一致性问题研究理念?

# 数据一致性问题研究理念 (IV)

## 论文研究问题:

如何在大规模分布式系统中  
落实“多样化, 可调节; 精细化, 可度量”的  
数据一致性问题研究理念?

## 论文主要贡献:

理念: 提出“多样化, 可调节; 精细化, 可度量”的数据  
一致性问题研究理念

# 数据一致性问题研究理念 (IV)

## 论文研究问题:

如何在大规模分布式系统中  
落实“多样化, 可调节; 精细化, 可度量”的  
数据一致性问题研究理念?

## 论文主要贡献:

理念: 提出“多样化, 可调节; 精细化, 可度量”的数据  
一致性问题研究理念

VPC: 验证 Pipelined-RAM Consistency [“精细化, 可度量”]

PA2AM: 量化 2-atomicity 协议 [“精细化, 可度量”]

RVSI: 可调节 Snapshot Isolation [“多样化, 可调节”]

# 分布数据一致性技术研究

① 研究背景

② 研究问题

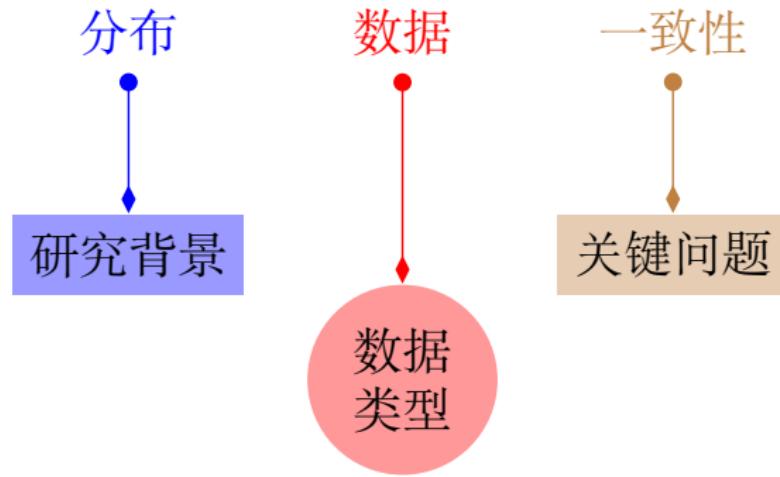
③ 技术框架

④ 相关工作

⑤ 本文工作

⑥ 总结与展望

# 分布数据一致性问题



# 技术框架

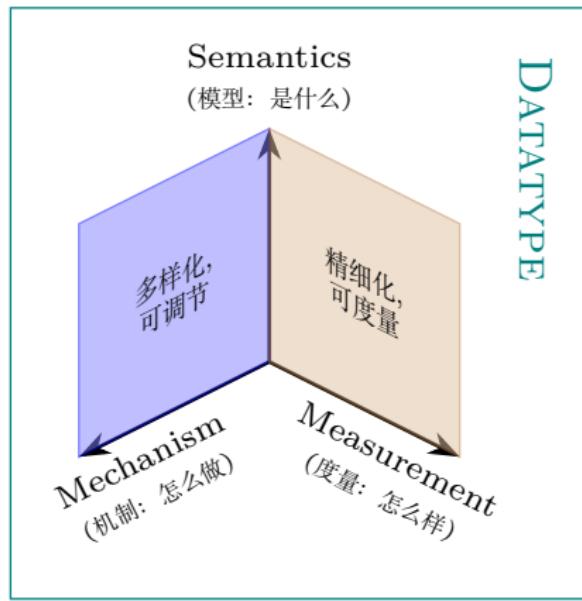


图: “一个基础, 三个维度” 技术框架.

# 基础: 数据类型

数据类型: 从个体到群组

# 基础: 数据类型

数据类型: 从个体到群组

- ▶ 读写寄存器

# 基础: 数据类型

数据类型: 从个体到群组

- ▶ 读写寄存器
- ▶ 事务对象
  - ▶ 事务  $\triangleq$  多个读写寄存器的操作序列
  - ▶ 支持 “all-or-none” 语义
  - ▶ 易于开发并发应用

# 维度一：一致性模型

一致性模型 [Steinke@JACM'04] [Adya@Thesis'99]:

- ▶ 多进程并发操作某数据类型
- ▶ 规定各操作的语义

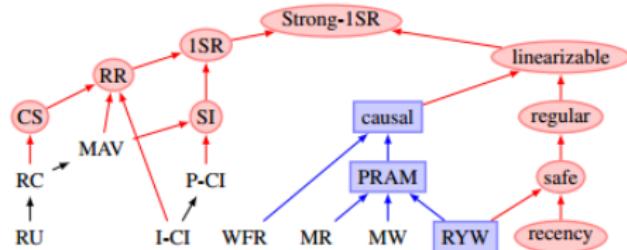


图: 一致性模型强弱关系 (来自 [Bailis@VLDB'14]).

# 维度一：一致性模型

一致性模型的集合定义：

# 维度一：一致性模型

一致性模型的集合定义：

系统执行  $e \triangleq$  该执行所产生的事件的序列

# 维度一：一致性模型

一致性模型的集合定义：

系统执行  $e \triangleq$  该执行所产生的事件的序列

分布式系统  $S \triangleq \{\text{该系统的所有可能执行}\}$

# 维度一：一致性模型

一致性模型的集合定义：

系统执行  $e \triangleq$  该执行所产生的事件的序列

分布式系统  $S \triangleq \{\text{该系统的所有可能执行}\}$

一致性模型  $C \triangleq \{\text{该模型所允许的所有系统执行}\}$

# 维度一：一致性模型

一致性模型的集合定义：

系统执行  $e \triangleq$  该执行所产生的事件的序列

分布式系统  $\mathcal{S} \triangleq \{\text{该系统的所有可能执行}\}$

一致性模型  $\mathcal{C} \triangleq \{\text{该模型所允许的所有系统执行}\}$



## 维度二：一致性实现机制

给定一致性模型  $\mathcal{C}$ , 设计系统  $\mathcal{S}$ :

$$\forall e \in \mathcal{S} : e \in \mathcal{C}.$$

i.e.,  $\mathcal{S} \subseteq \mathcal{C}$ .

## 维度二：一致性实现机制

给定一致性模型  $\mathcal{C}$ , 设计系统  $\mathcal{S}$ :

$$\forall e \in \mathcal{S} : e \in \mathcal{C}.$$

i.e.,  $\mathcal{S} \subseteq \mathcal{C}$ .

“多样化, 可调节” 研究理念的挑战:

- ▶ 兼容的混合一致性模型
- ▶ 实现手段之一: 参数化

# 维度三：一致性度量

给定系统  $S$  及一致性模型  $C$ ,

# 维度三: 一致性度量

给定系统  $\mathcal{S}$  及一致性模型  $\mathcal{C}$ ,

对于  $e \in \mathcal{S}$ :

验证 (verify):  $e \in \mathcal{C}?$   $\Rightarrow \{0, 1\}$

量化 (quantify):  $e \in \mathcal{C}?$   $\Rightarrow (0, 1)$



# 维度三: 一致性度量

给定系统  $S$  及一致性模型  $\mathcal{C}$ ,

对于  $e \in S$ :

验证 (verify):  $e \in \mathcal{C}?$   $\Rightarrow \{0, 1\}$

量化 (quantify):  $e \in \mathcal{C}?$   $\Rightarrow (0, 1)$



“精细化, 可度量” 研究理念的挑战:

验证: 算法设计、复杂度分析

量化: 数学建模与求解

# 分布数据一致性技术研究

1 研究背景

2 研究问题

3 技术框架

4 相关工作

5 本文工作

6 总结与展望

# 相关工作分类

表：“多样化, 可调节; 精细化, 可度量”研究理念相关工作.

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”					
“精细化, 可度量”	验证				
	量化				

# 相关工作分类

表：“多样化, 可调节; 精细化, 可度量”研究理念相关工作.

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”				软件 事务内存	
“精细化, 可度量”	验证				
	量化				

# 相关工作总结

**表:** “多样化, 可调节; 精细化, 可度量”研究理念相关工作.  
**(详见附录)**

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”		相关工作丰富 理论扎实	渐成趋势 理论欠缺	软件 事务内存	探索阶段
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究		理论全面 指导协议设计
	量化	暂无 强调正确性	量化执行易 量化协议难		量化协议难 相关工作少

# 分布数据一致性技术研究

1 研究背景

2 研究问题

3 技术框架

4 相关工作

5 本文工作

6 总结与展望

# 工作概述

**表:** 本文落实“多样化, 可调节; 精细化, 可度量”研究理念.

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”		相关工作丰富 理论扎实	渐成趋势 理论欠缺	软件 事务内存	探索阶段 (RVSI)
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 (VPC)		理论全面 指导协议设计
	量化	暂无 强调正确性	量化协议 (PA2AM)		量化协议难 相关工作少

# VPC 工作在技术框架中的位置

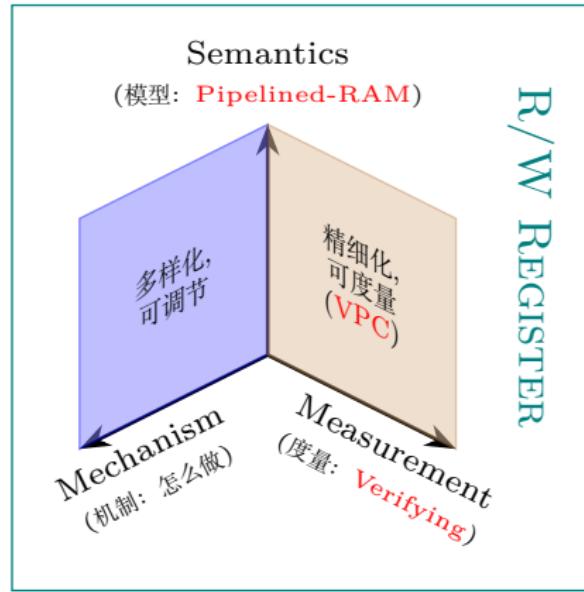


图: VPC — Pipelined-RAM 一致性验证.

# VPC 问题定义

定义 (VPC: Verifying PRAM Consistency)

VPC 判定问题:

实例: 系统执行 (*execution e*)

问题: 该执行是否满足 PRAM 一致性模型 ( $\mathcal{C}$ )?

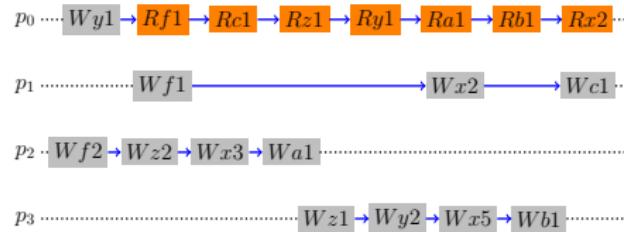
$$e \in \mathcal{C} \Rightarrow \{0, 1\}?$$

实例规模  $n$ : 系统执行中操作的总数

# VPC 问题定义

## 定义 (系统执行)

系统执行  $(e) \triangleq \{h_p \mid h_p : \text{进程 } p \text{ 上的读写操作序列}\}$



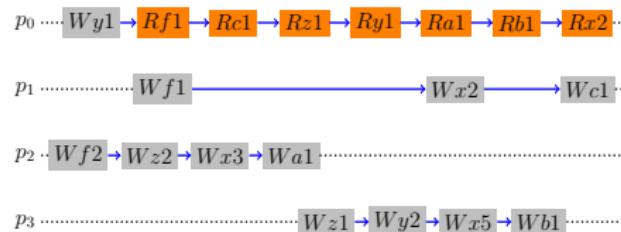
# VPC 问题定义

定义 (Pipelined-RAM 一致性模型)

系统执行  $e$  满足 PRAM 一致性

$\iff$

$\forall p : p$  上所有操作与其它进程上所有写操作存在合法调度.



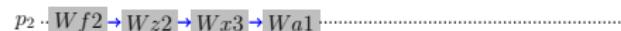
# VPC 问题定义

定义 (Pipelined-RAM 一致性模型)

系统执行  $e$  满足 PRAM 一致性

$\iff$

$\forall p : p$  上所有操作与其它进程上所有写操作存在合法调度.



$p_0 : W f2 \ W f1 \ W z2 \ W z1 \ W y2 \ W y1 \ Rf1 \ W x5 \ W x3 \ W x2 \ W cl \ Rcl$   
 $Rz1 \ R y1 \ W a1 \ R a1 \ W b1 \ R b1 \ R x2$

# 研究动机

问题: 为什么要验证 Pipelined-RAM (PRAM) 一致性?

验证: 用户与商家就数据一致性签订 SLA 协议

[Amazon@SOSP'07] [Golab@PODC'11]

- ▶ [商家] 系统测试手段之一
- ▶ [用户] 确认系统是否提供了其所声称的数据一致性

# 研究动机

问题: 为什么要验证 Pipelined-RAM (PRAM) 一致性?

验证: 用户与商家就数据一致性签订 SLA 协议

[Amazon@SOSP'07] [Golab@PODC'11]

- ▶ [商家] 系统测试手段之一
- ▶ [用户] 确认系统是否提供了其所声称的数据一致性

PRAM: 存储系统常提供“会话”(session) 一致性

[Saito@CSUR'05] [Terry@CACM'13]

- ▶ 包含了弱一致性的诸多变体
- ▶ 近似于 PRAM 一致性 [Brzeziński@PDP'04] [Bailis@VLDB'13]

# VPC 问题分类

表: VPC 问题的四种变体 (按“执行”的类型) 及复杂度分析 ([\*]: 本文工作).

	<i>(S)ingle variable</i>	<i>(M)ultiple variables</i>
<i>write (D)uplicate values</i>	VPC-SD	VPC-MD
<i>write (U)nique value</i>	VPC-SU	VPC-MU

# VPC 问题分类

表: VPC 问题的四种变体 (按“执行”的类型) 及复杂度分析 ([\*]: 本文工作).

	<i>(S)ingle variable</i>	<i>(M)ultiple variables</i>
<i>write (D)uplicate values</i>	VPC-SD <b>(NP-complete)</b> [*]	VPC-MD <b>(NP-complete)</b> [*]
<i>write (U)nique value</i>	VPC-SU <b>(P)</b> [Golab@PODC'11]	VPC-MU <b>(P)</b> [*]

# VPC 问题分类

表: VPC 问题的四种变体 (按“执行”的类型) 及复杂度分析 ([\*]: 本文工作).

	<i>(S)ingle variable</i>	<i>(M)ultiple variables</i>
<i>write (D)uplicate values</i>	VPC-SD <b>(NP-complete)</b> [*]	VPC-MD <b>(NP-complete)</b> [*]
<i>write (U)nique value</i>	VPC-SU <b>(P)</b> [Golab@PODC'11]	VPC-MU <b>(P)</b> [*]

Read-mapping [Gibbons@SICOMP'97]:  $\forall r, f(r) = w$ .

# VPC-SD (VPC-MD) 是 NP-complete 问题

多项式规约: 从 UNARY 3-PARTITION 到 VPC-SD.

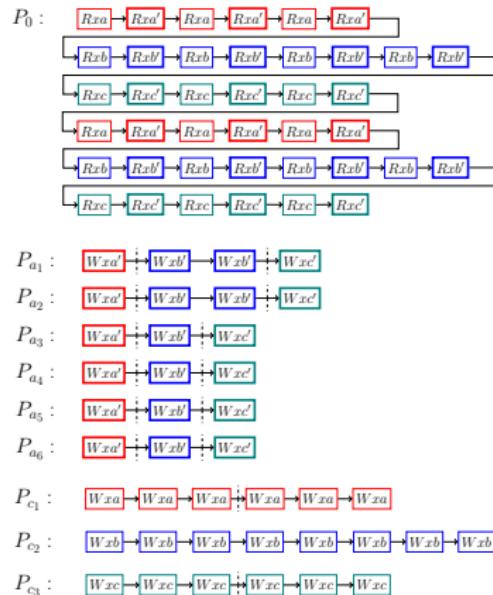


图: 对应于 UNARY 3-PARTITION 实例  $A = \{2, 2, 1, 1, 1, 1\}$ ,  $m = 2$ ,  $B = 4$  的 VPC 执行.

# VPC-MU 的多项式算法 RW-CLOSURE

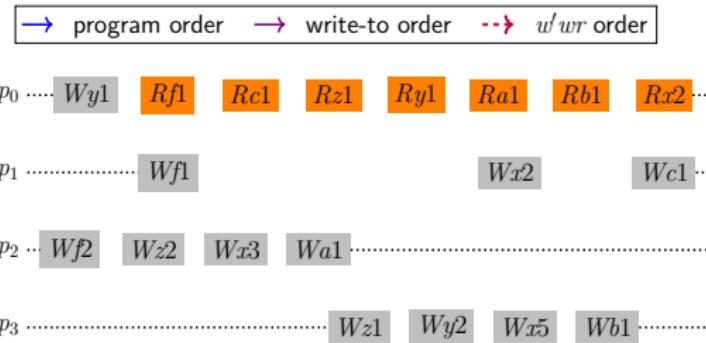


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

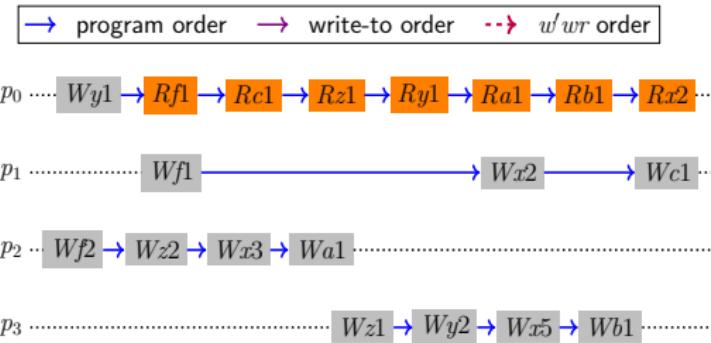


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

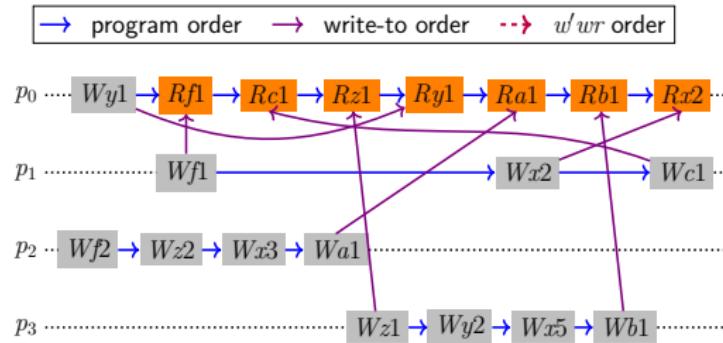


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

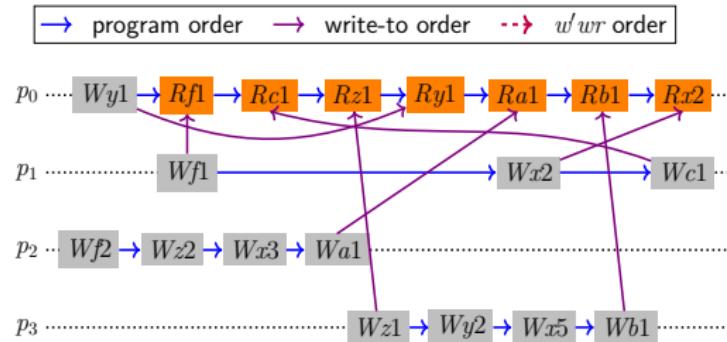


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

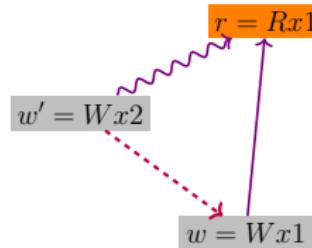


图:  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

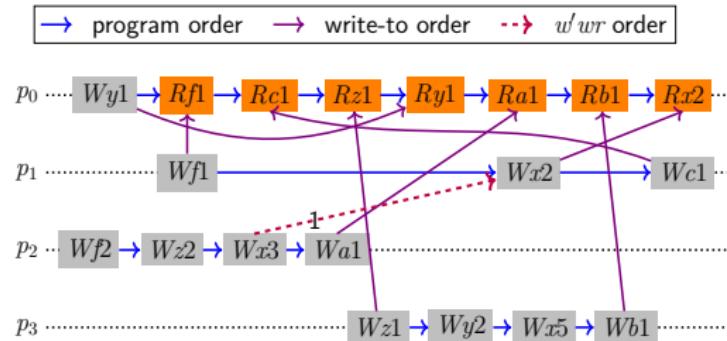


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w' wr$  规则.

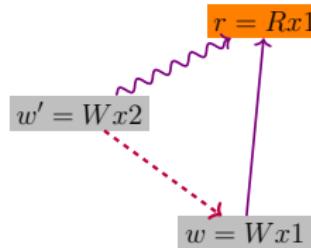
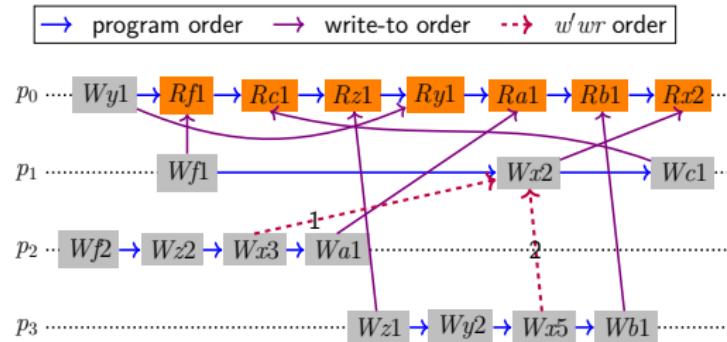
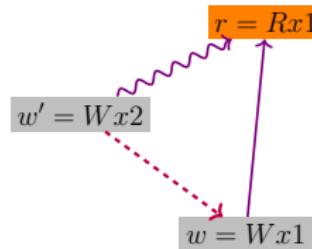


图:  $w' wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE



图：RW-CLOSURE 算法示例：在传递闭包之上迭代应用  $w'wr$  规则.



图： $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

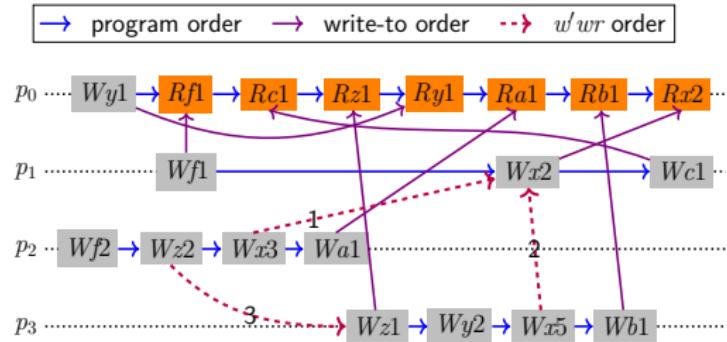


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

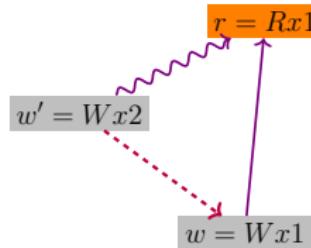
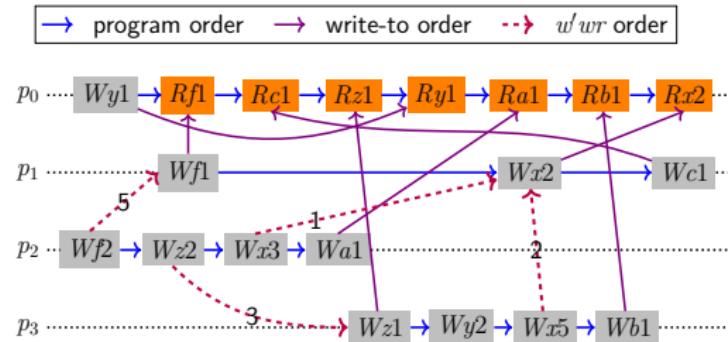
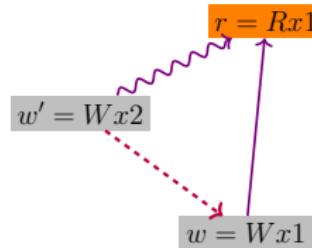


图:  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE



图：RW-CLOSURE 算法示例：在传递闭包之上迭代应用  $w' wr$  规则.



图： $w' wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

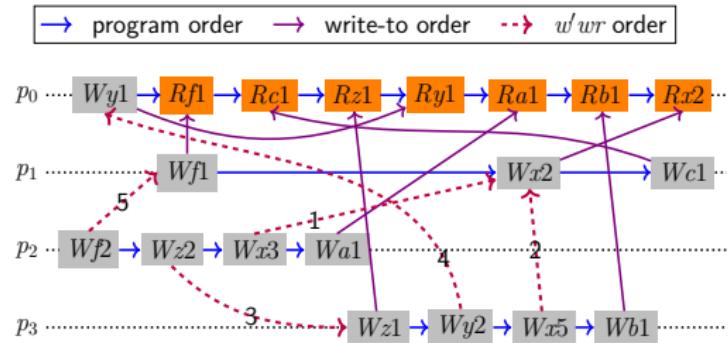


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

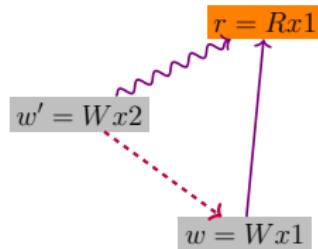


图:  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

定理 (RW-CLOSURE 算法正确性)

$VPC-MU$  实例满足  $PRAM$  一致性  
 $\iff$   
RW-CLOSURE 算法所得图是 DAG 图.

# VPC-MU 的多项式算法 RW-CLOSURE

定理 (RW-CLOSURE 算法正确性)

*VPC-MU 实例满足 PRAM 一致性*  
 $\iff$   
*RW-CLOSURE 算法所得图是 DAG 图.*

RW-CLOSURE 算法复杂度:

$$\underbrace{O(n^2)}_{\# \text{loops}} \cdot \underbrace{O(n^3)}_{\text{transitive closure}} = O(n^5)$$

# VPC-MU 的多项式算法 READ-CENTRIC

RW-CLOSURE 算法的缺点:

- ▶ 在全图上应用  $w'wr$  规则
- ▶ 应用  $w'wr$  规则无特定顺序

# VPC-MU 的多项式算法 READ-CENTRIC

RW-CLOSURE 算法的缺点:

- ▶ 在全图上应用  $w'wr$  规则
- ▶ 应用  $w'wr$  规则无特定顺序

READ-CENTRIC 算法要点:

- ▶ 增量式调度每个读操作
- ▶ 在读操作诱导的局部子图上按逆拓扑序应用  $w'wr$  规则

# VPC-MU 的多项式算法 READ-CENTRIC

RW-CLOSURE 算法的缺点:

- ▶ 在全图上应用  $w'wr$  规则
- ▶ 应用  $w'wr$  规则无特定顺序

READ-CENTRIC 算法要点:

- ▶ 增量式调度每个读操作
- ▶ 在读操作诱导的局部子图上按逆拓扑序应用  $w'wr$  规则

RW-CLOSURE 算法复杂度:

$$\underbrace{O(n)}_{\text{iterations}} \cdot \underbrace{O(n^3)}_{\text{TOPO-SCHEDULE}} = O(n^4)$$

# 实验评估

实验目的:

1. 考察 READ-CENTRIC 算法的实际效率 (*vs.* 演近时间复杂度)
2. 对比 READ-CENTRIC 算法与 RW-CLOSURE 算法的效率

# 实验评估

实验目的:

1. 考察 READ-CENTRIC 算法的实际效率 (*vs. 演近时间复杂度*)
2. 对比 READ-CENTRIC 算法与 RW-CLOSURE 算法的效率

两组实验:

1. 随机生成的读写记录
2. 满足 PRAM 一致性的读写记录 ( $\approx$  最坏情况输入)

# 实验评估

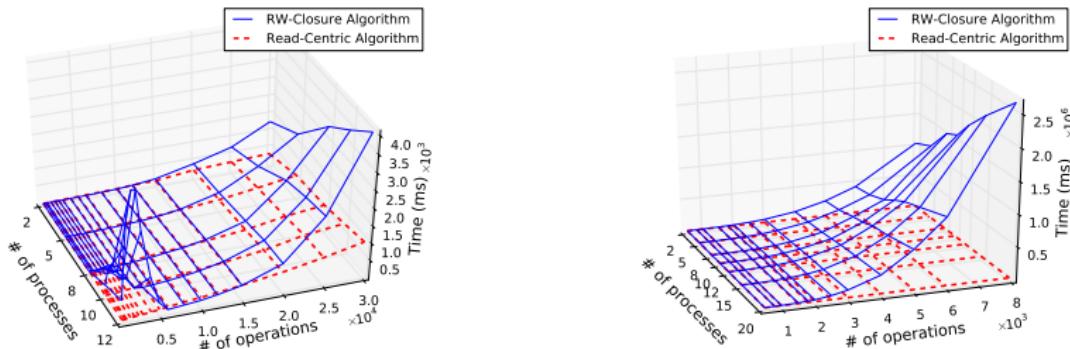


图: RW-CLOSURE 算法与 READ-CENTRIC 算法在 (左) 随机生成的读写记录及(右) 满足 PRAM 一致性的读写记录上的运行时间。

# 实验评估

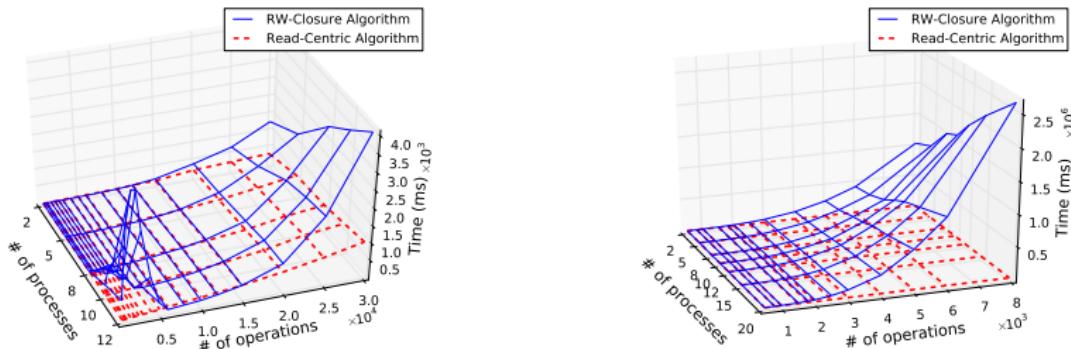


图: RW-CLOSURE 算法与 READ-CENTRIC 算法在 (左) 随机生成的读写记录及(右) 满足 PRAM 一致性的读写记录上的运行时间。

(右) 20 个进程、8,000 个操作:  
READ-CENTRIC 可获得 694 倍加速.

# 实验评估

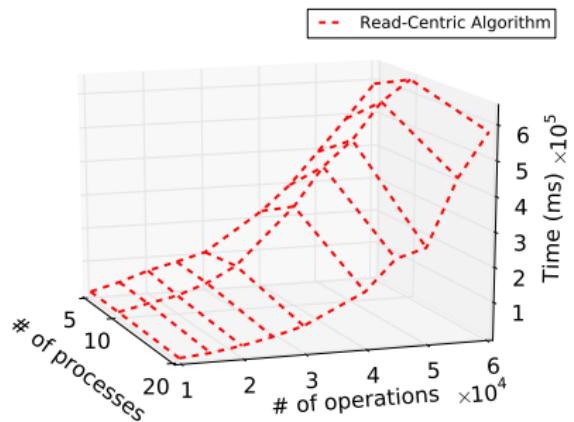


图: READ-CENTRIC 算法在满足 PRAM 一致性的读写记录上的运行时间.

READ-CENTRIC: 20 个进程、60,000 个操作 < 600s

RW-CLOSURE: 20 个进程、8,000 个操作 > 3,000s

# VPC 的意义

## 对 VPC 问题的系统研究:

1. READ-CENTRIC 算法可用于测试系统是否正确实现了 PRAM 一致性模型
2. NP-completeness 结果有助于理解弱一致性模型的复杂度

# PA2AM 工作在技术框架中的位置

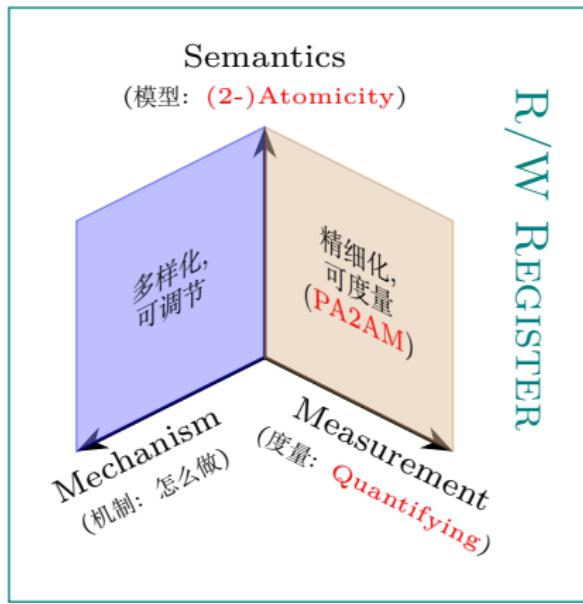


图: PA2AM — (2-)Atomicity 一致性维护与量化.

# 研究动机

问题: 为什么提出 probabilistically-atomic 2-atomicity (PA2AM) 一致性?

# 研究动机

问题: 为什么提出 probabilistically-atomic 2-atomicity (PA2AM) 一致性?

“数据一致性/访问延迟” PACELC 权衡 [Abadi@IEEE Computer'12]:



# 研究动机

问题: 为什么提出 probabilistically-atomic 2-atomicity (PA2AM) 一致性?

“数据一致性/访问延迟” PACELC 权衡 [Abadi@IEEE Computer'12]:



“低延迟” 至关重要:

- ▶ 100ms 额外延迟  $\Rightarrow$  1% 销售下滑 [Amazon@Blog'06]
- ▶ 100~400ms 额外延迟  $\Rightarrow$  0.2%~0.6% 搜索量下降 [Google@Blog'09]

# PA2AM 一致性

在保证**低延迟**的情况下获得**尽可能强**的数据一致性.

# PA2AM 一致性

“近乎强” 一致性: 在保证**低延迟**的情况下获得**尽可能强**的数据一致性.

# PA2AM 一致性

“近乎强” 一致性: 在保证**低延迟**的情况下获得**尽可能强**的数据一致性.

定义 (PA2AM 一致性)

# PA2AM 一致性

“近乎强” 一致性: 在保证**低延迟**的情况下获得**尽可能强**的数据一致性.

定义 (PA2AM 一致性)

低延迟: 读操作只需一轮网络通信

# PA2AM 一致性

“近乎强” 一致性: 在保证低延迟的情况下获得尽可能强的数据一致性.

定义 (PA2AM 一致性)

低延迟: 读操作只需一轮网络通信

定理 (不可能性结果)

(单写模型下) 不存在低延迟的 atomicity 维护算法 [Dutta@PODC'04].

# PA2AM 一致性

“近乎强” 一致性: 在保证低延迟的情况下获得尽可能强的数据一致性.

## 定义 (PA2AM 一致性)

低延迟: 读操作只需一轮网络通信

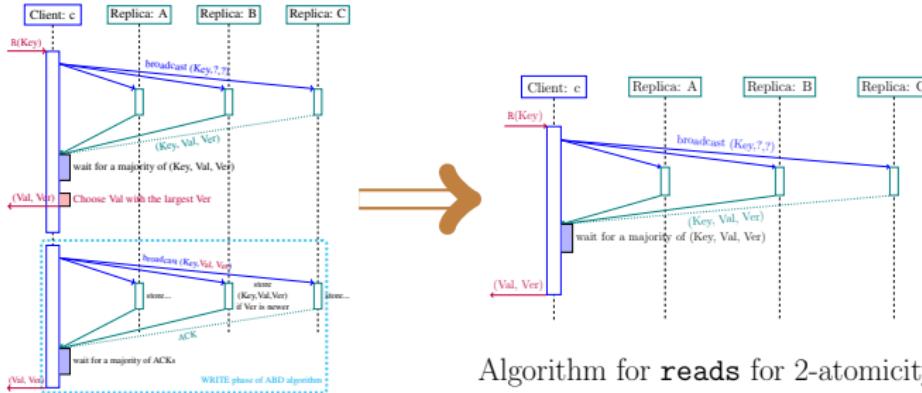
尽可能强: 对 *atomicity (strongest)* 的弱化

- ▶ (版本) 2-atomicity: 允许读陈旧值, 但陈旧度  $k \leq 2$
- ▶ (概率)  $\mathbb{P}(k = 2)$  很小

## 定理 (不可能性结果)

(单写模型下) 不存在低延迟的 *atomicity* 维护算法 [Dutta@PODC'04].

# PA2AM 维护算法



Algorithm for **reads** for 2-atomicity.

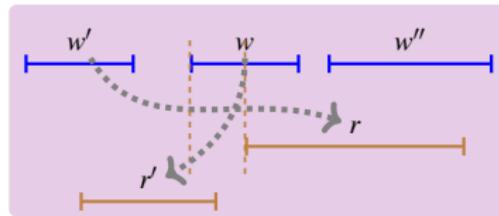
Algorithm for **reads** for atomicity.

图: 经典 atomicity 算法中, 读操作需两轮网络通信 [Attiya@JACM'95] [Dutta@PODC'04].  
PA2AM 算法实现 2-atomicity (单写模型下), 读操作只需一轮网络通信.

# PA2AM 量化分析

问题: PA2AM 算法在多大程度上违反了 atomicity?

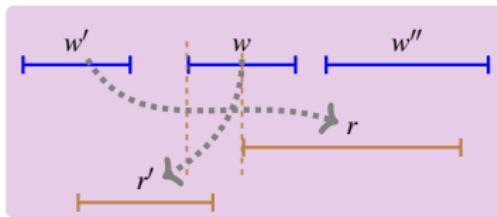
充要条件: ONI (old-new inversion) [Attiya@JACM'95]



# PA2AM 量化分析

问题: PA2AM 算法在多大程度上违反了 atomicity?

充要条件: ONI (old-new inversion) [Attiya@JACM'95]



将 ONI 分解为“并发模式”与“读写模式”:

定义 (CP: Concurrency Pattern)

1.  $r_{st} \in [w_{st}, w_{ft}]$ ;
2.  $w'$  immediately precedes  $w$ ;
3.  $r'_{ft} \in [w_{st}, r_{st}]$ .

定义 (RWP: Read-Write Pattern)

4.  $r = R(w')$ ;
5.  $r' = R(w)$ .

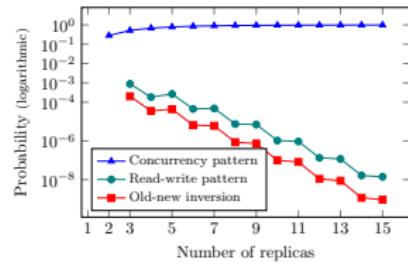
# PA2AM 量化分析

PA2AM 量化分析: 计算  $\mathbb{P}(\text{ONI})$ , 其值越小越好

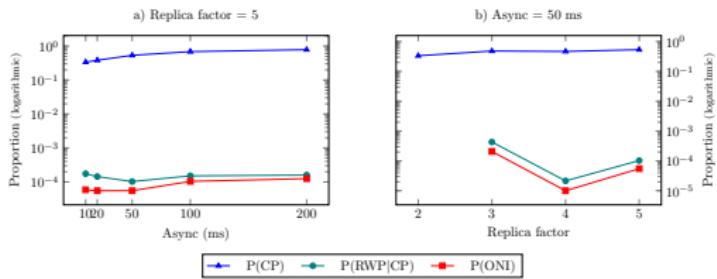
1.  $\text{ONI} \triangleq \text{CP} \cap \text{RWP}$
2. 排队论建模, 计算  $\mathbb{P}(\text{CP})$
3. 带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP} | \text{CP})$

$$\begin{aligned}
 \mathbb{P}\{\text{CP} | R' = m\} &= \mathbb{P}(E_{N-1,m}) \\
 &= \sum_{k=0}^{N-2} \binom{N-1}{k} \binom{m-1}{N-k-2} p_0^k r^{N-k-1} s^m \\
 &\quad \cdot \mathbb{P}\{\text{RWP} | R' = m\} \\
 &\leq \mathbb{P}\{r \neq R(w)\} \times \left(1 - \mathbb{P}\{r' \neq R(w) | r \neq R(w)\}^m\right) \\
 &\leq e^{-q\lambda_w t} \frac{\alpha^q B(q, \alpha(n-q)+1)}{B(q, n-q+1)} \\
 &\quad \cdot \left(1 - \left(\frac{J_1}{B(q, n-q+1)}\right)^m\right). \tag{4}
 \end{aligned}$$

# PA2AM 量化分析

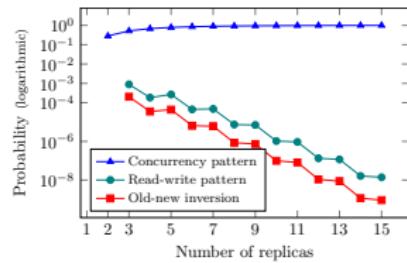


(a) 数值结果.

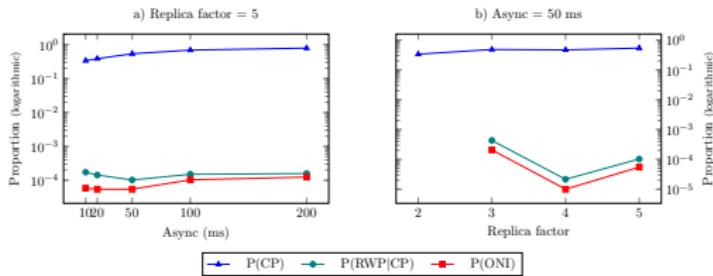


(b) 实验结果.

# PA2AM 量化分析



(a) 数值结果.

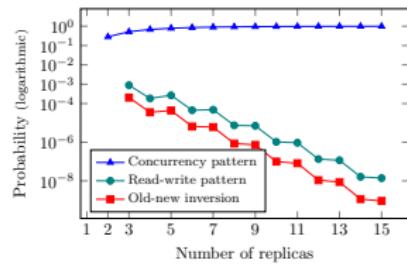


(b) 实验结果.

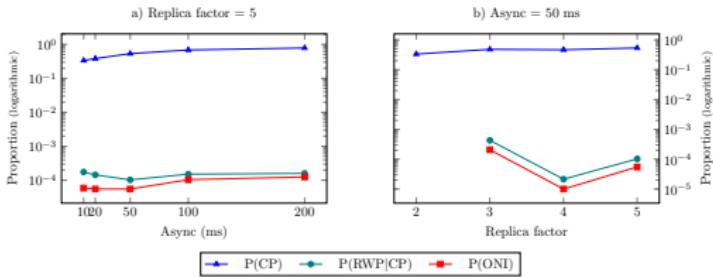
观察: (观察一)

从概率角度讲, PA2AM 算法极少违反 atomicity 一致性.

# PA2AM 量化分析



(a) 数值结果.



(b) 实验结果.

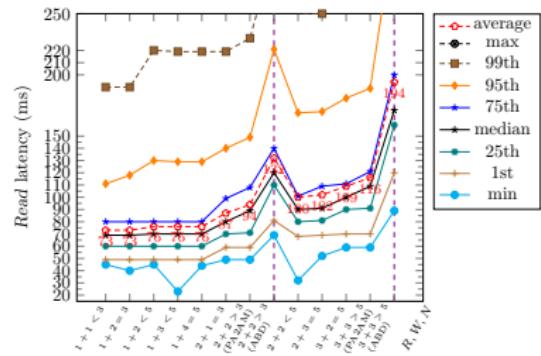
观察: (观察一)

从概率角度讲, PA2AM 算法极少违反 *atomicity* 一致性.

观察: (观察二)

与经常发生的并发模式相比, 读写模式起主导作用。

# PA2AM vs. 弱一致性模型



(a) 读操作延迟对比.

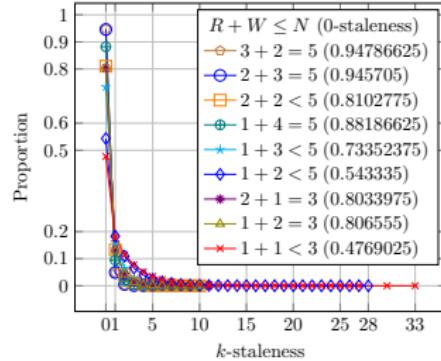
(b)  $k$ -陈旧度对比.

图: PA2AM 与 eventual consistency (RWN-Maj 协议) 对比结果.

表: 不同  $R, W, N$  配置下, RWN-All 执行中具有不同陈旧度的读操作的比率.

# replicas	replica factor = 3 (400,000 read operations)				replica factor = 5 (800,000 read operations)						
	$1 + 1 < 3$	$1 + 2 = 3$	$2 + 1 = 3$	$2 + 2 > 3$ (PA2AM)	$1 + 2 < 5$	$1 + 3 < 5$	$1 + 4 = 5$	$2 + 2 < 5$	$2 + 3 = 5$	$3 + 2 = 5$	$3 + 3 > 5$ (PA2AM)
$R, W, N$	6	4	2	1	2	2	2	2	2	1	
$\sum_{k>1}$ -staleness	0.0084125	0.000315	0.0004675	0.0000085	0.00377875	0.002755	0.00406	0.0027225	0.0020275	0.002255	0.0003525

# PA2AM 的意义

PA2AM 可作为一致性/延迟权衡的一种有价值的选项:

1. 既 (在统计意义上) 满足强一致性模型对数据一致性的高标准
2. 又具有弱一致性模型的性能优势

# RVSI 工作在技术框架中的位置

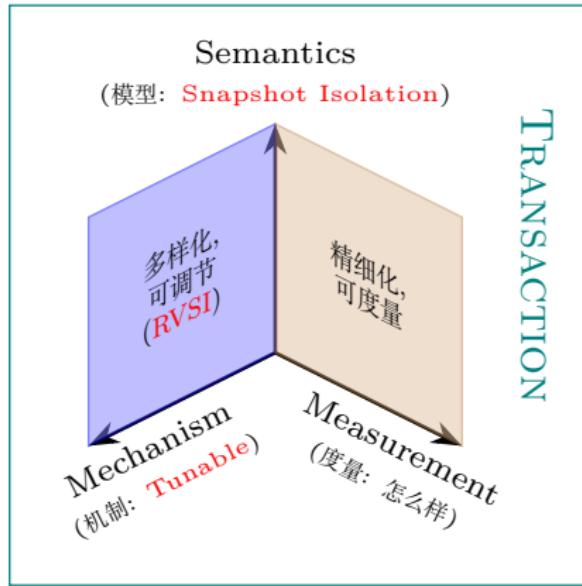


图: RVSI — Snapshot Isolation 一致性弱化与维护.

# 研究动机

问题: 为什么要提出 Relaxed Version Snapshot Isolation (RVSI) 一致性?

分布式事务:

▶ “all-or-none” 语义

▶ 受到分布式存储系统的关注

[Cassandra@CASSANDRA-ISSUE-7056'14]

弱一致性: PCSI [Elkilicety@SRDS'05] SI [Lin@TODS'09]

PSI [Sovran@SOSP'11] NMSI [Ardekani@SRDS'13]

# 研究动机

问题: 为什么要提出 Relaxed Version Snapshot Isolation (RVSI) 一致性?

分布式事务:

▶ “all-or-none” 语义

▶ 受到分布式存储系统的关注

[Cassandra@CASSANDRA-ISSUE-7056'14]

弱一致性: PCSI [Elnikety@SRDS'05] SI [Lin@TODS'09]

PSI [Sovran@SOSP'11] NMSI [Ardekani@SRDS'13]

异常控制: 容忍“有限度的”异常 [Yu@TOCS'02]

可调节:

▶ 不同应用对一致性需求不同 [Terry@CACM'13]

▶ 运行时决定 [Terry@SOSP&TR'13]

# RVSIs 定义

RVSIs (Relaxed Version Snapshot Isolation) 定义原则:

- ▶ RC ⊂ RVSIs( $k_1, k_2, k_3$ ) ⊂ SI

RC (Read Committed): 只能读取成功提交的更新

SI (Snapshot Isolation): 读取最近的系统快照 + 无写冲突事务

# RVSI 定义

RVSI (Relaxed Version Snapshot Isolation) 定义原则:

- ▶  $\text{RC} \supset \text{RVSI}(k_1, k_2, k_3) \supset \text{SI}$
- ▶ 参数  $k_1, k_2, k_3$  控制“异常”程度
- ▶  $\text{RVSI}(\infty, \infty, \infty) = \text{RC}; \quad \text{RVSI}(1, 0, *) = \text{SI}$

RC (Read Committed): 只能读取成功提交的更新

SI (Snapshot Isolation): 读取最近的系统快照 + 无写冲突事务

# RVSI 定义

## 定义 (RVSI (要点))

弱化 SI 关于“读取最近的系统快照”的要求。

单变量读  $\text{read}(x)$ :

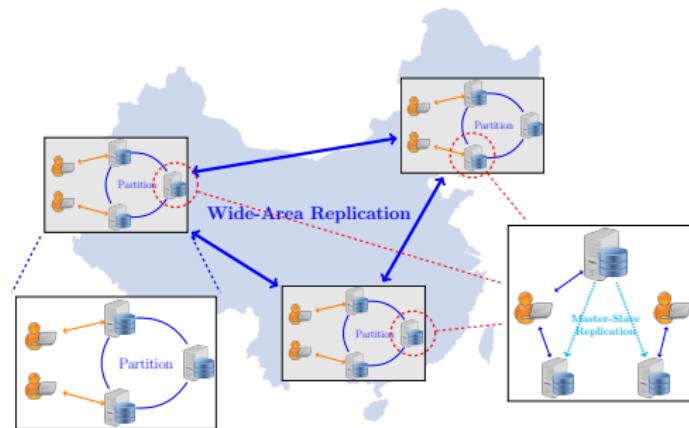
1. 允许读  $\leq k_1$  陈旧值 ( $k_1\text{-BV}$ )
2. 允许读  $\leq k_2$  并发更新 ( $k_2\text{-FV}$ )

多变量读  $\text{read}(x), \text{read}(y)$ :

3.  $\text{dist}(\text{snap}(x), \text{snap}(y)) \leq k_3$

# CHAMELEON 分布式事务键值存储原型系统设计

系统架构: 阿里云<sup>4</sup> 多数据中心 ( $9 = 3 \times 3$ )



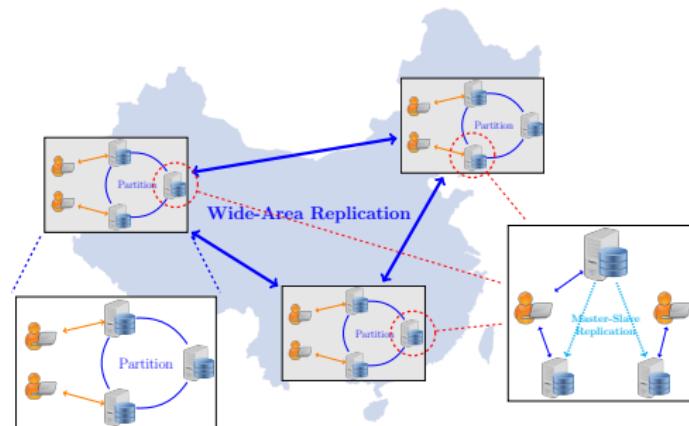
<sup>4</sup>阿里云: <https://www.aliyun.com/>.

# CHAMELEON 分布式事务键值存储原型系统设计

系统架构: 阿里云<sup>4</sup> 多数据中心 ( $9 = 3 \times 3$ )

数据分区: 同一数据中心

数据副本: 跨数据中心; 主从结构



<sup>4</sup>阿里云: <https://www.aliyun.com/>.

# RVSI 维护算法

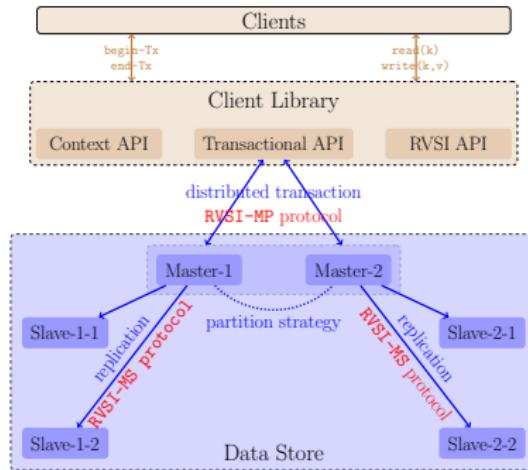
$$\text{RC} \supset \text{RVSI}(k_1, k_2, k_3) \supset \text{SI}$$

RVSI 维护算法:

- ▶ 以分布式 RC 和 SI 协议为基础
- ▶ 事务提交前, 计算 RVSI “版本约束” ( $k_1, k_2, k_3$  相关不等式)
  - $k_1\text{-BV}$ :  $\mathcal{O}_x(T_i.sts) - \mathcal{O}_x(T_j.cts) < k_1$
  - $k_2\text{-FV}$ :  $\mathcal{O}_x(T_j.cts) - \mathcal{O}_x(T_i.sts) \leq k_2$
  - $k_3\text{-SV}$ :  $\mathcal{O}_x(T_l.cts) - \mathcal{O}_x(T_j.cts) \leq k_3$
- ▶ 事务提交时, 检查 RVSI “版本约束”

# RVI 维护算法

系统组件: 客户端库 + 数据中心

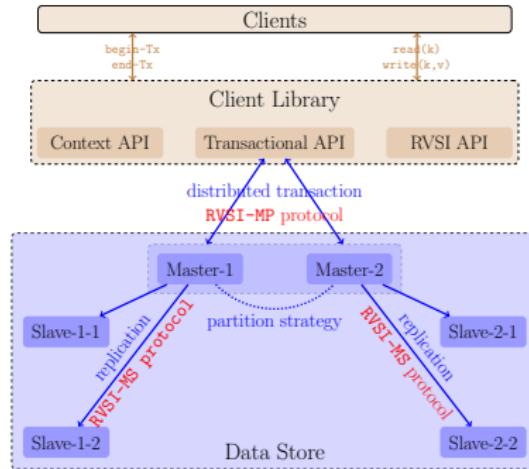


# RVS1 维护算法

系统组件: 客户端库 + 数据中心

数据分区: 分布式事务原子提交协议 (2PC)

数据副本: 懒惰复制 (lazy replication) 协议



# RVI 实验评估

表: 事务负载参数表.  
(评估目标: RVI 对事务中止率的影响)

Parameter	F(ixed)/V(ariable)/R(andom)	Value	Explanation
#keys	F	5	size of keyspace
mpl	V	5, 10, 15, 20, 25, 30	multiprogramming level: number of concurrent clients
#txs/client	F	1000	number of txs per client
#ops/tx	R	~ Binomial(20, 0.5)	number of operations per tx
rwRatio	V	1:2, 1:1, 4:1	#reads/#writes
zipfExponent	F	1	parameter for Zipfian distribution
$(k_1, k_2, k_3)$	V	(1,0,0) (1,0,2) (1,1,0) (2,0,0) (2,1,2) (2,2,1)	for $k_1$ -BV, $k_2$ -FV, and $k_3$ -SV
minInterval	F	0ms	minimum inter-transactions time
maxInterval	F	10ms	maximum inter-transactions time
meanInterval	R	5ms	mean inter-transactions time for exponential distribution

# RVI 实验评估

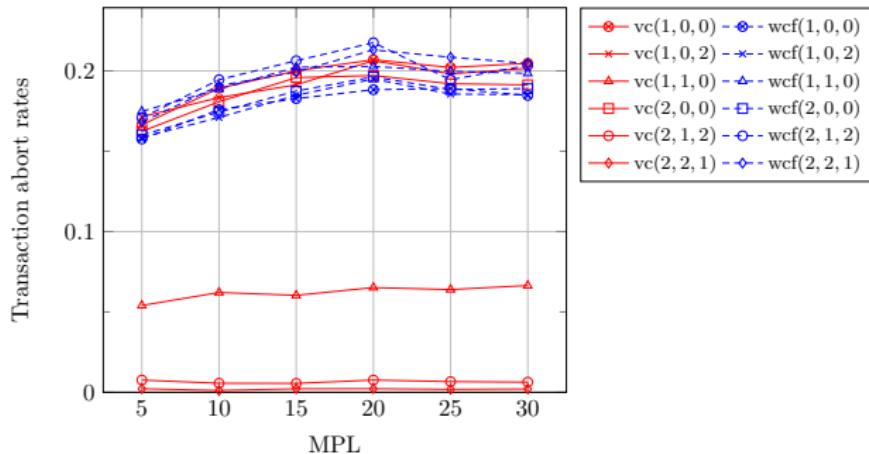
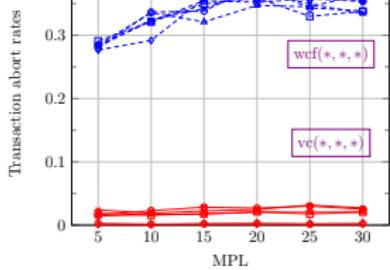


图: 读频繁 ( $rwRatio = 4:1$ ) 负载下 RVI 对事务中止率的影响.

wcf-aborted: 无显著变化 ( $wcf(1, 0, 0) = 0.184733$ )

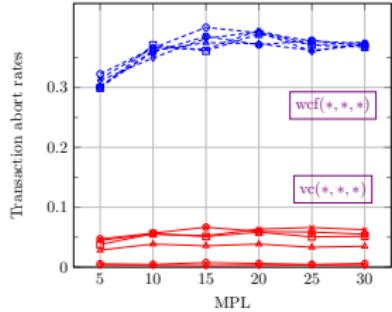
vc-aborted: 显著减少 ( $vc(1, 0, 0) = 0.204733; vc(1, 1, 0) = 0.066433; vc(2, 2, 1) = 0.002033$ )

# RVSI 实验评估



vc(\*, \*, \*)  
wcf(\*, \*, \*)

图: 写频繁 ( $rwRatio = 1:2$ ) 负载下 RVSI 对事务中止率的影响.



vc(\*, \*, \*)  
wcf(\*, \*, \*)

图: 读写相当 ( $rwRatio = 1:1$ ) 负载下 RVSI 对事务中止率的影响.

# RVSI 的意义

## RVSI 对事务中止率的影响:

1. 适当放松事务对 RVSI 版本规约的要求可降低事务中止率
2. RVSI 能否“显著”降低事务中止率与负载类型相关

# 分布数据一致性技术研究

① 研究背景

② 研究问题

③ 技术框架

④ 相关工作

⑤ 本文工作

⑥ 总结与展望

# 工作总结

## 论文主要贡献:

理念: 提出“多样化, 可调节; 精细化, 可度量”的数据一致性问题研究理念

VPC: 验证 Pipelined-RAM Consistency ["精细化, 可度量"]

PA2AM: 量化 2-atomicity 协议 ["精细化, 可度量"]

RVSI: 可调节 Snapshot Isolation ["多样化, 可调节"]

# 未来工作

落实“多样化, 可调节; 精细化, 可度量”的数据一致性问题研究理念:

# 未来工作

落实“多样化, 可调节; 精细化, 可度量”的数据一致性问题研究理念:

1. VHC (HC: Hybrid Consistency) 问题 [Attiya@SICOMP'98] (遵循 VPC 工作思路)

# 未来工作

落实“多样化, 可调节; 精细化, 可度量”的数据一致性问题研究理念:

1. VHC (HC: Hybrid Consistency) 问题 [Attiya@SICOMP'98] (遵循 VPC 工作思路)

2. “多写模型”下的 atomic 寄存器 (扩展 PA2AM 工作)

低延迟: 读操作只需一轮网络通信

可能性问题: 是否存在低延迟 ( $k$ -)atomicity 算法?

尽可能强: 如何定义并量化 pAM ( $p$ : probabilistic)?

# 未来工作

落实“多样化, 可调节; 精细化, 可度量”的数据一致性问题研究理念:

1. VHC (HC: Hybrid Consistency) 问题 [Attiya@SICOMP'98] (遵循 VPC 工作思路)
2. “多写模型”下的 atomic 寄存器 (扩展 PA2AM 工作)  
    **低延迟**: 读操作只需一轮网络通信  
    **可能性问题**: 是否存在低延迟 ( $k$ -)atomicity 算法?  
    **尽可能强**: 如何定义并量化 pAM ( $p$ : probabilistic)?
3. 考虑更丰富的数据类型 (replicated data types) [Burckhardt@POPL'14]

# 发表论文

- ▶ [TC'16] **Hengfeng Wei**, Yu Huang, Jian Lu.  
Probabilistically-Atomic 2-Atomicity: Enabling Almost Strong Consistency in Distributed Storage Systems. In *IEEE Trans. Comput.*, xx(x):x–x, PrePrints, 2016.
- ▶ [TPDS'16] **Hengfeng Wei**, Marzio De Biasi, Yu Huang, Jiannong Cao, Jian Lu. Verifying Pipelined-RAM Consistency over Read/Write Traces of Data Replicas. In *IEEE Trans. Parallel Distrib. Syst.*, 27(5):1511–1523, 2016.
- ▶ [PerCom'12] **Hengfeng Wei**, Yu Huang, Jiannong Cao, Xiaoxing Ma, Jian Lu. Formal Specification and Runtime Detection of Temporal Properties for Asynchronous Context. In *Proceedings of the 10th IEEE International Conference on Pervasive Computing and Communications*, pages 30–38, 2012.
- ▶ [WIP for VLDB'17] **Hengfeng Wei**, Yu Huang, Jian Lu. Relaxed Version Snapshot Isolation in Distributed Transactional Key-Value Stores.

# 致谢

- ▶ 导师: 吕建教授、黄宇副教授
- ▶ 软件所全体师生
- ▶ 论文评审与答辩老师



[hengxin0912@gmail.com](mailto:hengxin0912@gmail.com)



# 分布数据一致性技术研究

7 附录: 参考文献

8 附录: 相关工作

# 分布数据一致性技术研究

7 附录: 参考文献

8 附录: 相关工作

# 相关工作分类

表: “多样化, 可调节; 精细化, 可度量” 研究理念相关工作.

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”					
“精细化, 可度量”	验证				
	量化				

# 相关工作分类

表: “多样化, 可调节; 精细化, 可度量” 研究理念相关工作.

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”				软件 事务内存	
“精细化, 可度量”	验证				
	量化				

# “多样化, 可调节” 的研究理念 (一)

	读写寄存器		事务	
	多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”	✓			

# “多样化, 可调节” 的研究理念 (一)

	读写寄存器	事务	
	多处理器系统	分布式系统	多处理器系统
“多样化, 可调节”	✓		

典型: Hybrid consistency [Attiya@SICOMP'98]

思想: 将操作分为强弱两类

# “多样化, 可调节” 的研究理念 (一)

	读写寄存器	事务	
	多处理器系统	分布式系统	多处理器系统
“多样化, 可调节”	✓		

**典型:** Hybrid consistency [Attiya@SICOMP'98]

**思想:** 将操作分为强弱两类

**其它:** “带同步的” 一致性模型 [Dubois@IEEE Computer'88] [Steinke@JACM'04]

**特点:** 强调正确性 (properly synchronized)

# “多样化, 可调节” 的研究理念 (一)

	读写寄存器	事务		
	多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”	相关工作丰富 理论扎实			

典型: Hybrid consistency [Attiya@SICOMP'98]

思想: 将操作分为强弱两类

其它: “带同步的” 一致性模型 [Dubois@IEEE Computer'88] [Steinke@JACM'04]

特点: 强调正确性 (properly synchronized)

# “多样化, 可调节” 的研究理念 (二)

	读写寄存器		事务	
	多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”	相关工作丰富 理论扎实	✓		

# “多样化, 可调节” 的研究理念 (二)

	读写寄存器	事务	
	多处理器系统	分布式系统	多处理器系统
“多样化, 可调节”	相关工作丰富 理论扎实	✓	

思想: 借鉴并发展 Hybrid consistency 的思想

- 典型:
- ▶ Causal+forced+immediate operations [Ladin@TOCS'92]
  - ▶ RedBlue consistency [Li@OSDI'12]
  - ▶ Pileus [Terry@SOSP'13]

# “多样化, 可调节” 的研究理念 (二)

	读写寄存器	事务	
	多处理器系统	分布式系统	多处理器系统
“多样化, 可调节”	相关工作丰富 理论扎实	✓	

**思想:** 借鉴并发展 Hybrid consistency 的思想

- 典型:**
- ▶ Causal+forced+immediate operations [Ladin@TOCS'92]
  - ▶ RedBlue consistency [Li@OSDI'12]
  - ▶ Pileus [Terry@SOSP'13]

**特点:** 更细粒度的多一致性模型共存、更能容忍数据不一致

# “多样化, 可调节” 的研究理念 (二)

	读写寄存器	事务		
	多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”	相关工作丰富 理论扎实	渐成趋势 理论欠缺		

**思想:** 借鉴并发展 Hybrid consistency 的思想

- 典型:**
- ▶ Causal+forced+immediate operations [Ladin@TOCS'92]
  - ▶ RedBlue consistency [Li@OSDI'12]
  - ▶ Pileus [Terry@SOSP'13]

**特点:** 更细粒度的多一致性模型共存、更能容忍数据不一致

# “多样化, 可调节” 的研究理念 (三)

	读写寄存器	事务		
	多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”	相关工作丰富 理论扎实	渐成趋势 理论欠缺	软件事务内存	✓

思想: 多个事务一致性模型共存

- 典型:
- ▶ RC-SR (relaxed currency serializability) [Bernstein@SIGMOD'06]
  - ▶ Pileus consistency choices [Terry@MSR-TR'13]
  - ▶ Multi-level CSI (Causal Snapshot Isolation) [Tripathi@BigData'15]

# “多样化, 可调节” 的研究理念 (三)

	读写寄存器	事务		
	多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”	相关工作丰富 理论扎实	渐成趋势 理论欠缺	软件事务内存	✓

思想: 多个事务一致性模型共存

- 典型:
- ▶ RC-SR (relaxed currency serializability) [Bernstein@SIGMOD'06]
  - ▶ Pileus consistency choices [Terry@MSR-TR'13]
  - ▶ Multi-level CSI (Causal Snapshot Isolation) [Tripathi@BigData'15]

挑战: “多样化” 事务语义; 可扩展的系统实现

# “多样化, 可调节” 的研究理念 (三)

	读写寄存器	事务	
	多处理器系统	分布式系统	多处理器系统
多样化, 可调节”	相关工作丰富 理论扎实	渐成趋势 理论欠缺	软件事务内存

思想: 多个事务一致性模型共存

典型:

- ▶ RC-SR (relaxed currency serializability) [Bernstein@SIGMOD'06]
- ▶ Pileus consistency choices [Terry@MSR-TR'13]
- ▶ Multi-level CSI (Causal Snapshot Isolation) [Tripathi@BigData'15]

挑战: “多样化” 事务语义; 可扩展的系统实现

# “精细化, 可度量” 的研究理念 (一)

		读写寄存器	事务		
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证				
	量化				

# “精细化, 可度量”的研究理念 (一)

		读写寄存器	事务		
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	✓			
	量化				

典型的一致性模型验证 (Verify) 问题:

- ▶ VSC (Sequential Consistency), VL (Linearizability) [Gibbons@SICOMP'97]
- ▶ VMC (Memory Coherence) [Cantin@SPAA'03] [Cantin@TPDS'05]
- ▶ VTSO (Total Store Order) [Hangal@ISCA'03] [Manovit@SPAA'05] [Roy@CAV'06] [Baswana@CAV'08]

# “精细化, 可度量”的研究理念 (一)

		读写寄存器	事务		
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面			
	量化				

典型的一致性模型验证 (Verify) 问题:

- ▶ VSC (Sequential Consistency), VL (Linearizability) [Gibbons@SICOMP'97]
- ▶ VMC (Memory Coherence) [Cantin@SPAA'03] [Cantin@TPDS'05]
- ▶ VTSO (Total Store Order) [Hangal@ISCA'03] [Manovit@SPAA'05] [Roy@CAV'06] [Baswana@CAV'08]

# “精细化, 可度量”的研究理念 (一)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面			
	量化	暂无 强调正确性			

典型的一致性模型验证 (Verify) 问题:

- ▶ VSC (Sequential Consistency), VL (Linearizability) [Gibbons@SICOMP'97]
- ▶ VMC (Memory Coherence) [Cantin@SPAA'03] [Cantin@TPDS'05]
- ▶ VTSO (Total Store Order) [Hangal@ISCA'03] [Manovit@SPAA'05] [Roy@CAV'06] [Baswana@CAV'08]

# “精细化, 可度量”的研究理念 (二)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	✓		
	量化	暂无 强调正确性			

动机: 商业条款 SLA (Service Level Agreement) [Amazon@SOSP'07]

# “精细化, 可度量”的研究理念 (二)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	✓		
	量化	暂无 强调正确性			

**动机:** 商业条款 SLA (Service Level Agreement) [Amazon@SOSP'07]

**特点:** 在线验证 safeness, regularity, atomicity [Golab@PODC'11]

# “精细化, 可度量”的研究理念 (二)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究		
	量化	暂无 强调正确性			

**动机:** 商业条款 SLA (Service Level Agreement) [Amazon@SOSP'07]

**特点:** 在线验证 safeness, regularity, atomicity [Golab@PODC'11]

**不足:** 常用 Pipelined-RAM consistency, causal consistency, hybrid consistency 验证问题有待研究

# “精细化, 可度量”的研究理念 (三)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究		
	量化	暂无 强调正确性			

# “精细化, 可度量”的研究理念 (三)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究		
	量化	暂无 强调正确性	✓		

**量化执行:**  $k/\Delta/\Gamma$ -atomicity [Golab@PODC'11, ICDCS'13, ICDCS'14, PODC'15]

**量化协议:** probabilistic regularity/atomicity

[Yu@DISC'03] [Lee@DC'05] [Gramoli@OPODIS'07] [Bailis@PVLDB'12]

# “精细化, 可度量”的研究理念 (三)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究		
	量化	暂无 强调正确性	量化执行易 量化协议难		

**量化执行:**  $k/\Delta/\Gamma$ -atomicity [Golab@PODC'11, ICDCS'13, ICDCS'14, PODC'15]

**量化协议:** probabilistic regularity/atomicity

[Yu@DISC'03] [Lee@DC'05] [Gramoli@OPODIS'07] [Bailis@PVLDB'12]

# “精细化, 可度量”的研究理念 (四)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究	软件事务内存	✓
	量化	暂无 强调正确性	量化执行易 量化协议难		

## ▶ SR (Serializability) 强一致性模型及变体

[Papadimitriou@JACM'79] [Bernstein@TODS'83] [Yannakakis@JACM'84]

## ▶ SI (Snapshot Isolation) 等弱一致性模型

[Adya@Phd-Thesis'99] [Fekete@TODS'05] [Cahill@SIGMOD'08] [Zellag@VLDB'14]

# “精细化, 可度量”的研究理念 (四)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究	软件事务内存	✓
	量化	暂无 强调正确性	量化执行易 量化协议难		

## ▶ SR (Serializability) 强一致性模型及变体

[Papadimitriou@JACM'79] [Bernstein@TODS'83] [Yannakakis@JACM'84]

## ▶ SI (Snapshot Isolation) 等弱一致性模型

[Adya@Phd-Thesis'99] [Fekete@TODS'05] [Cahill@SIGMOD'08] [Zellag@VLDB'14]

# “精细化, 可度量”的研究理念 (四)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究	软件事务内存	理论全面 指导协议设计
	量化	暂无 强调正确性	量化执行易 量化协议难		

## ▶ SR (Serializability) 强一致性模型及变体

[Papadimitriou@JACM'79] [Bernstein@TODS'83] [Yannakakis@JACM'84]

## ▶ SI (Snapshot Isolation) 等弱一致性模型

[Adya@Phd-Thesis'99] [Fekete@TODS'05] [Cahill@SIGMOD'08] [Zellag@VLDB'14]

# “精细化, 可度量”的研究理念 (五)

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究	软件事务内存	理论全面 指导协议设计
	量化	暂无 强调正确性	量化执行易 量化协议难		量化协议难 相关工作少

- ▶ 量化 SI (Snapshot Isolation) 与 RC (Read Committed) 协议 [Fekete@VLDB'09]

# 相关工作总结

表: “多样化, 可调节; 精细化, 可度量” 研究理念相关工作.

		读写寄存器		事务	
		多处理器系统	分布式系统	多处理器系统	分布式系统
“多样化, 可调节”		相关工作丰富 理论扎实	渐成趋势 理论欠缺	软件 事务内存	探索阶段 理论全面 指导协议设计 量化协议难 相关工作少
“精细化, 可度量”	验证	典型模型 理论全面	弱模型验证 有待研究		
	量化	暂无 强调正确性	量化执行易 量化协议难		