

# 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 26 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 主要成果

⑤ 总结展望

# 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 主要成果

5 总结展望

# 大规模分布式应用



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

---

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

# 大规模分布式应用



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

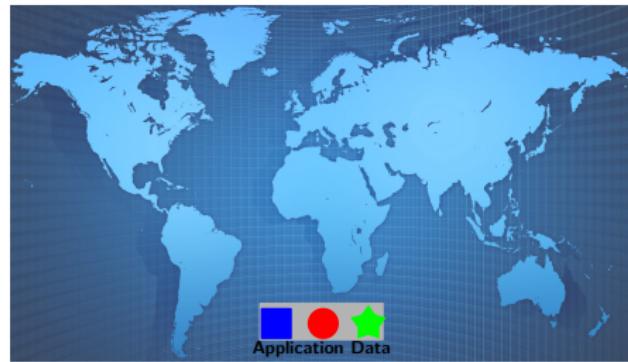
底层数据服务系统特性需求 ( $H^3L$ ):<sup>2</sup>

- ▶ 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.

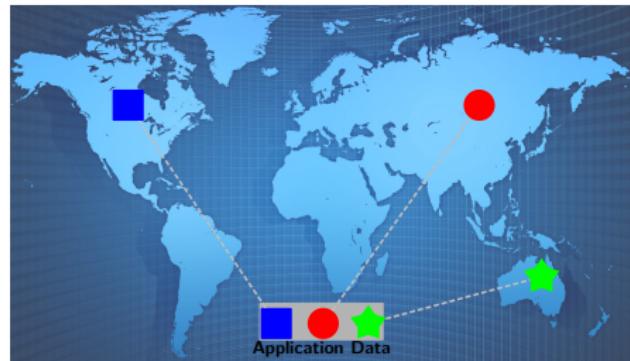
<sup>2</sup> 数据来自 InfoQ.

# 分布数据



分布数据 (distributed data):

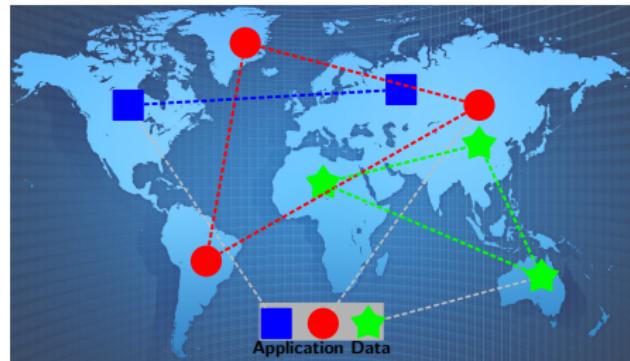
# 分布数据



分布数据 (distributed data):

1. 分区 (partition): 水平扩展

# 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

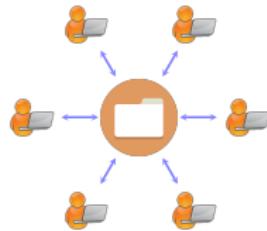
2 研究问题

3 技术框架

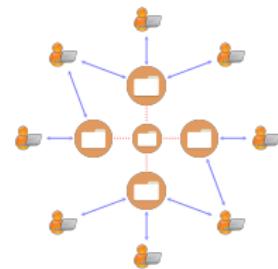
4 主要成果

5 总结展望

# 分布数据一致性问题

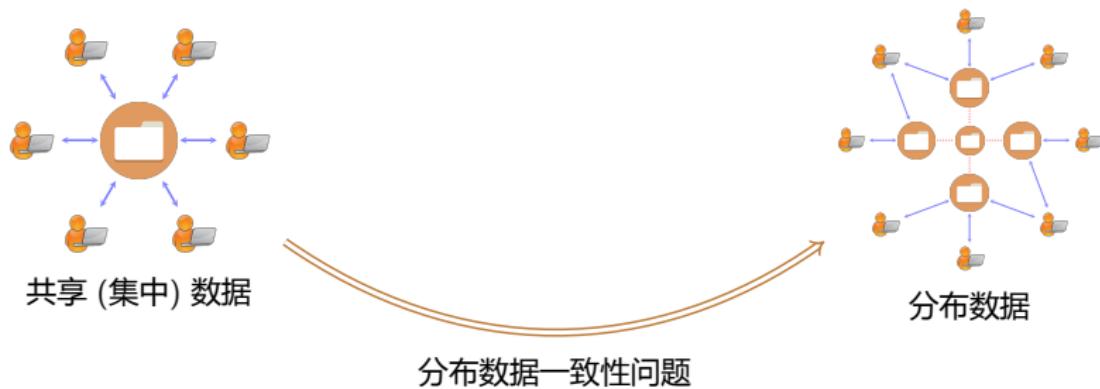


共享(集中)数据



分布数据

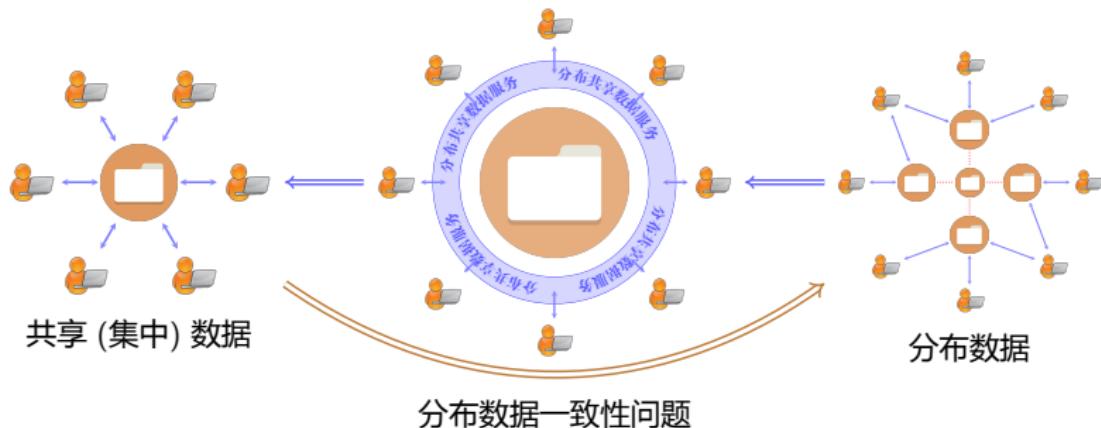
# 分布数据一致性问题



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



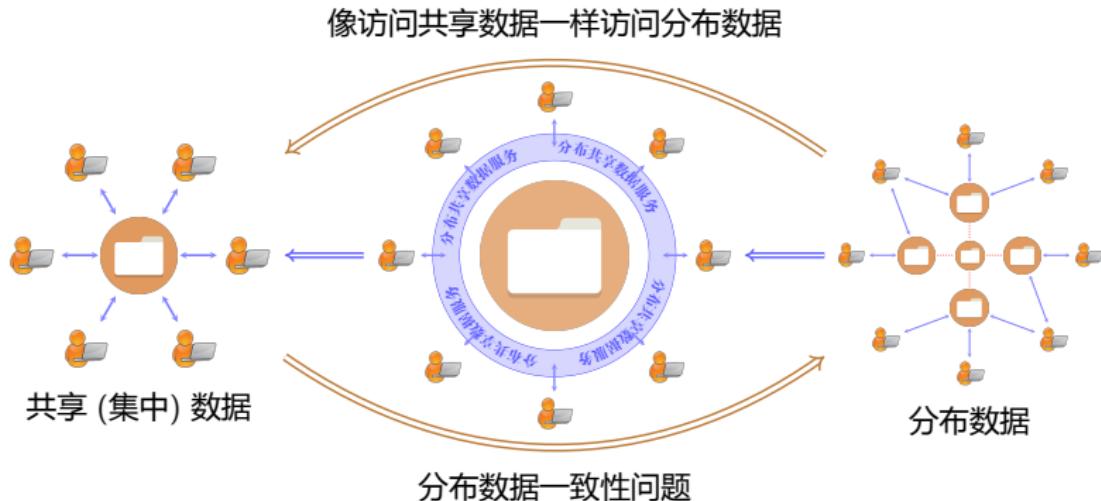
## 分布数据一致性问题 (应用视角):

- ▶ 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



## 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  
一致性方案
- ▶ 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  
一致性方案
- ▶ 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  
一致性方案
- ▶ 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



在大规模分布式系统中,  
 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

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

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](#)].



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

## 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

# 多处理器系统领域中的数据一致性理论

特性：“以程序为导向、强调正确性”

一致性：追求“相对于 SC (Sequential Consistency)”的正确性<sup>1</sup> [Adve@ISCA'90]

权衡：易编程性 vs. 性能

---

<sup>1</sup> 术语称为 SCNF: Sequential Consistency Normal Form [Adve@ISCA'90].

# 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:



可用性 vs. 一致性



图: CAP 定理 [Brewer@PODC'00]

访问延迟 vs. 一致性

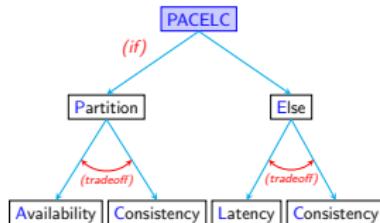


图: PACELC 权衡 [Abadi@IEEE Computer'12]

# 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



可用性 vs. 一致性



图: CAP 定理 [Brewer@PODC'00]

访问延迟 vs. 一致性

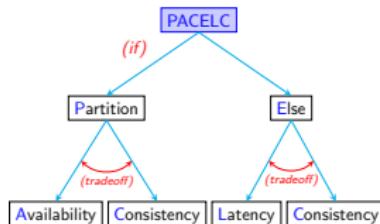


图: PACELC 权衡 [Abadi@IEEE Computer'12]

# 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:

# 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  $\implies$  正确性变得模糊 (blurred)

# 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  $\implies$  正确性变得模糊 (blurred)
- ▶ 更丰富的应用  $\implies$  多样性变得重要

# 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  $\implies$  正确性变得模糊 (blurred)
- ▶ 更丰富的应用  $\implies$  多样性变得重要

需要更具“柔性”的分布数据一致性理论:

# 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  $\Rightarrow$  正确性变得模糊 (blurred)
- ▶ 更丰富的应用  $\Rightarrow$  多样性变得重要

需要更具“柔性”的分布数据一致性理论:

- 理念一: 多样化, 可调节
- 理念二: 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  $\{0, 1\}$  到连续谱  $(0, 1)$  [Yu@TOCS'02]



# 数据一致性问题研究理念 (III)

精细化: 从二元  $\{0, 1\}$  到连续谱  $(0, 1)$  [Yu@TOCS'02]



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

度量系统满足一致性的程度

# 数据一致性问题研究理念 (IV)

## 论文研究问题

如何面向大规模分布式系统研究  
能体现“多样化, 可调节; 精细化, 可度量”  
理念的分布数据一致性技术?

# 数据一致性问题研究理念 (IV)

## 论文研究问题

如何面向大规模分布式系统研究  
能体现“多样化, 可调节; 精细化, 可度量”  
理念的分布数据一致性技术?

## 论文主要贡献

框架: 提出包含“一个基础、三个维度”的技术框架

# 数据一致性问题研究理念 (IV)

## 论文研究问题

如何面向大规模分布式系统研究  
能体现“多样化, 可调节; 精细化, 可度量”  
理念的分布数据一致性技术?

## 论文主要贡献

框架: 提出包含“一个基础、三个维度”的技术框架

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

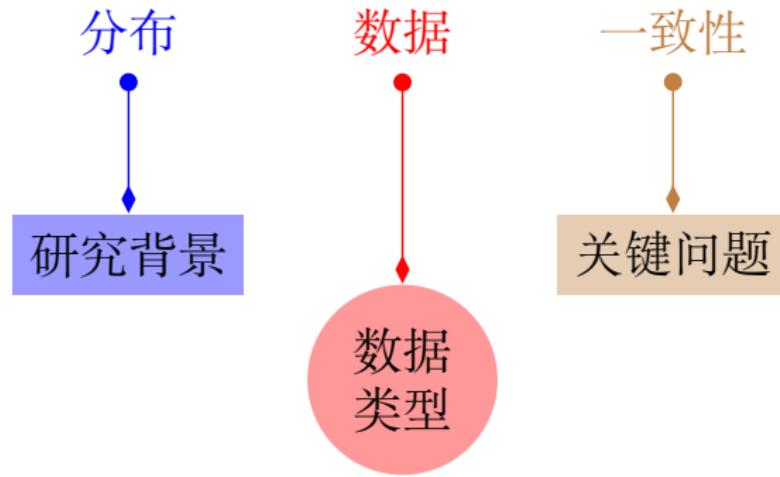
2 研究问题

3 技术框架

4 主要成果

5 总结展望

# 分布数据一致性问题



# 技术框架

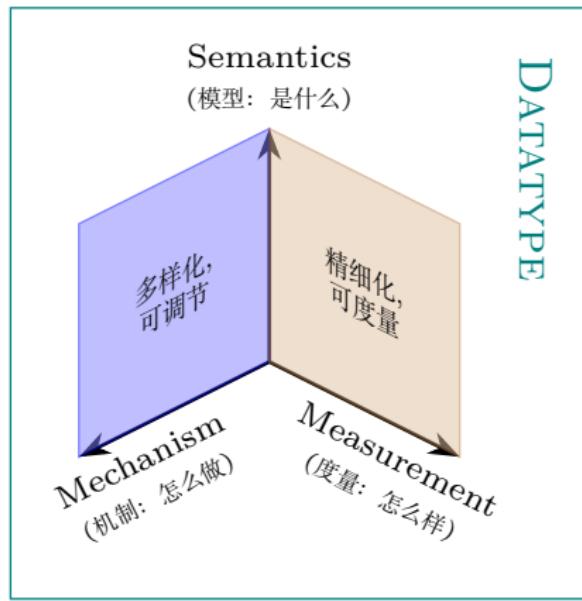


图: “一个基础, 三个维度” 技术框架.

# 基础: 数据类型

数据类型: 从个体到群组

## 1. 读写寄存器<sup>1</sup>

- ▶ 单个变量
- ▶ 读/写操作

---

<sup>1</sup>由于历史的原因, 分布式计算理论中使用术语“寄存器”[\[Lamport@DC'86\]](#).

# 基础: 数据类型

数据类型: 从个体到群组

## 1. 读写寄存器<sup>1</sup>

- ▶ 单个变量
- ▶ 读/写操作

## 2. 事务对象 (transaction; transactional object)

- ▶ 事务  $\triangleq$  多个读写寄存器的操作序列

`beginTx r(x) r(y) w(y) w(x) endTx`

- ▶ 原子性: “all-or-none” 语义, 易于开发并发应用

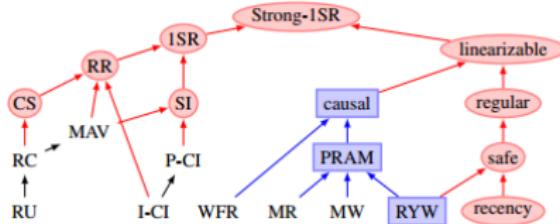
---

<sup>1</sup>由于历史的原因, 分布式计算理论中使用术语“寄存器”[\[Lamport@DC'86\]](#).

# 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
- ▶ 规定各操作的合法返回值

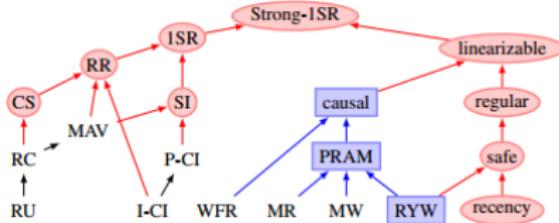


图：一致性模型强弱关系 (来自 [Bailis@VLDB'14]).

# 维度一：一致性模型

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

- ▶ 多进程并发操作某数据类型
- ▶ 规定各操作的合法返回值



图：一致性模型强弱关系 (来自 [Bailis@VLDB'14]).

## 维度二：一致性实现机制

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

## 维度二：一致性实现机制

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

$\mathcal{S}$  满足  $\mathcal{C}$ .

即, 每个操作的返回值都合法.

## 维度二: 一致性实现机制

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

$\mathcal{S}$  满足  $\mathcal{C}$ .

即, 每个操作的返回值都合法.

“多样化, 可调节” 研究理念的挑战:

- ▶ 严格定义 “多样化”
- ▶ 系统实现 “可调节”

# 维度三：一致性度量

给定一致性模型  $\mathcal{C}$  及系统  $\mathcal{S}$ ,

# 维度三: 一致性度量

给定一致性模型  $\mathcal{C}$  及系统  $\mathcal{S}$ ,

验证 (verify): 执行  $e$  是否满足  $\mathcal{C}$

量化 (quantify): 协议  $\mathcal{P}$  满足  $\mathcal{C}$  的程度

# 维度三: 一致性度量

给定一致性模型  $\mathcal{C}$  及系统  $\mathcal{S}$ ,

验证 (verify): 执行  $e$  是否满足  $\mathcal{C}$

量化 (quantify): 协议  $\mathcal{P}$  满足  $\mathcal{C}$  的程度

“精细化, 可度量” 研究理念的挑战:

验证: 算法设计与分析

量化: 数学建模与求解

# 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 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		

# 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

## 1. 渐成趋势

# 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 理论全面 指导协议设计
	验证	典型模型 理论全面	弱模型验证 有待研究		
“精细化, 可度量”	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

## 1. 渐成趋势

## 2. 可资借鉴

# 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”	验证	相关工作丰富	渐成趋势	软件	探索阶段
		理论扎实	理论欠缺		
“精细化, 可度量”	量化	典型模型	弱模型验证	事务内存	理论全面
		理论全面	有待研究		指导协议设计
		暂无	量化执行易		量化协议难
		强调正确性	量化协议难		相关工作少

## 1. 渐成趋势

## 2. 可资借鉴

## 3. 问题凸显

# 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 主要成果

5 总结展望

# 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

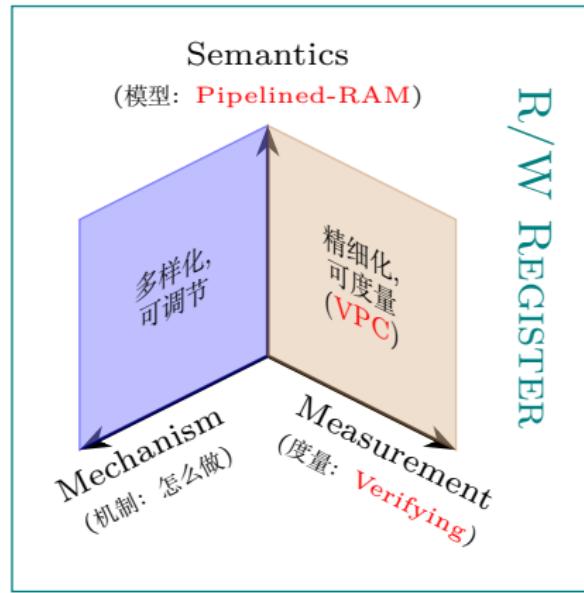


图: VPC — Pipelined-RAM (PRAM) 一致性验证.

# 研究动机

问题: 为什么要验证 PRAM 一致性?

验证: 用户与商家就数据一致性签订 SLA (Service Level Agreement) 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 PRAM 一致性?

验证: 用户与商家就数据一致性签订 SLA (Service Level Agreement) 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 *e* 是否满足 PRAM 一致性模型 ( $\mathcal{C}$ )?

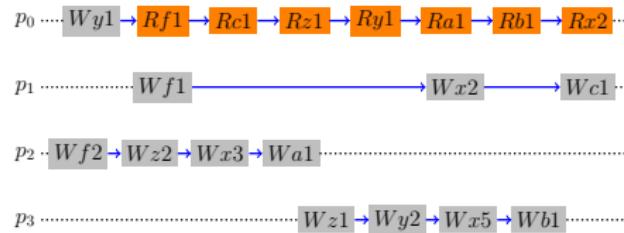
$$e \in \mathcal{C} \Rightarrow \{0, 1\}?$$

# VPC 问题定义

## 定义 (系统执行)

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

规模  $n$ : 系统执行中操作的总数



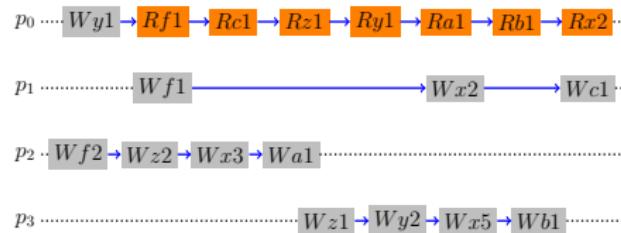
# VPC 问题定义

定义 (PRAM 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.



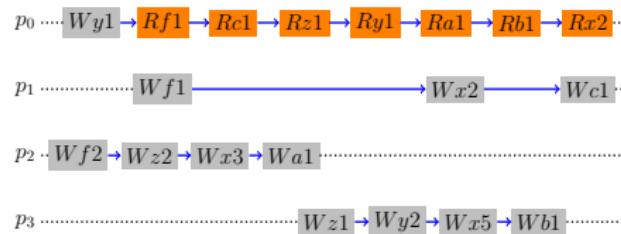
# VPC 问题定义

定义 (PRAM 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.



$p_0 : Wf2 \ Wf1 \ Wz2 \ Wz1 \ Wy2 \ Wy1 \ Rf1 \ Wx5 \ Wx3 \ Wx2 \ Wc1 \ Rcl$   
 $Rz1 \ Ry1 \ Wa1 \ Ra1 \ Wb1 \ Rb1 \ Rx2$

# 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]:  $\forall r, \exists! w, f(r) = w.$

# VPC-SD (VPC-MD) 是 NP-complete 问题

多项式规约: 从 UNARY 3-PARTITION 到 VPC-SD.

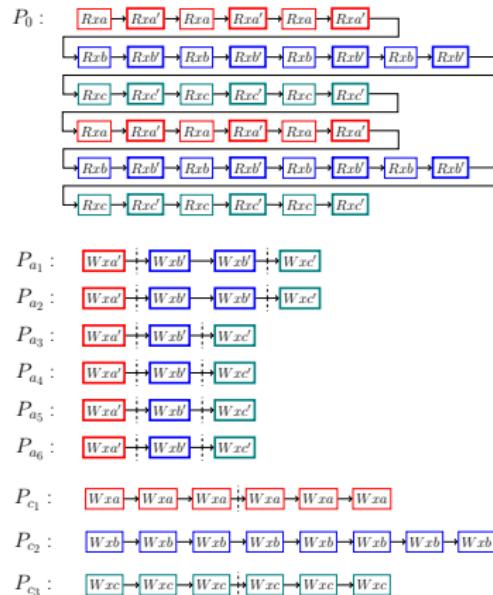


图: 对应于 UNARY 3-PARTITION 实例  $A = \{2, 2, 1, 1, 1, 1\}$ ,  $m = 2$ ,  $B = 4$  的 VPC 执行.

# VPC-MU 的多项式算法 RW-CLOSURE

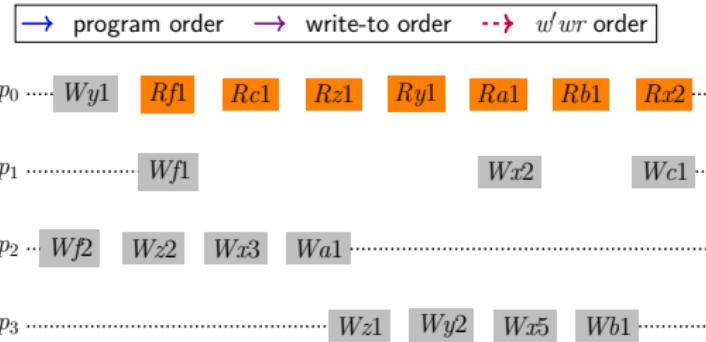


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

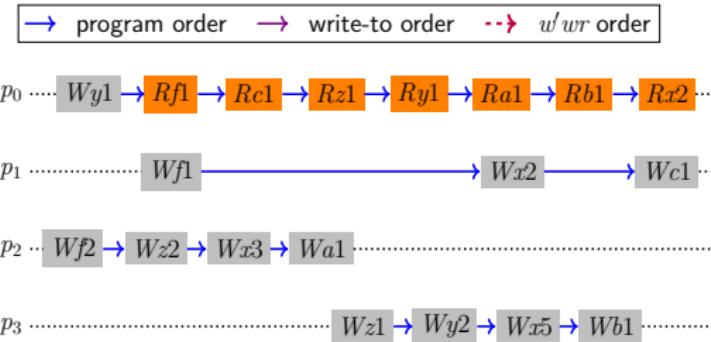


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

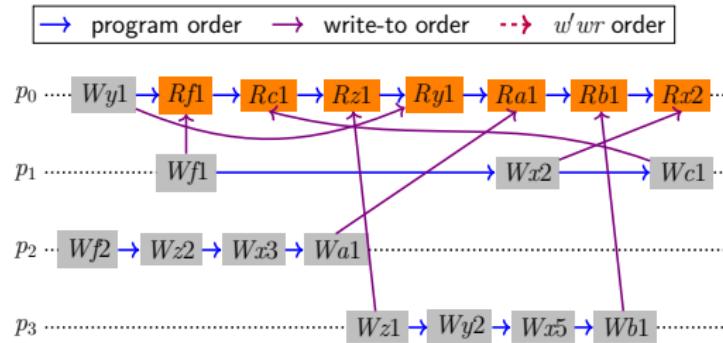


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 w'wr 规则.

# VPC-MU 的多项式算法 RW-CLOSURE

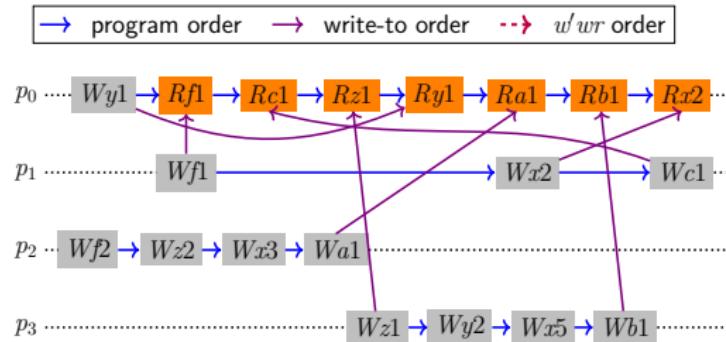


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w' wr$  规则.

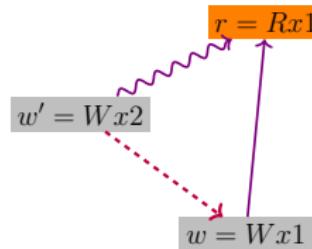


图:  $w' wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

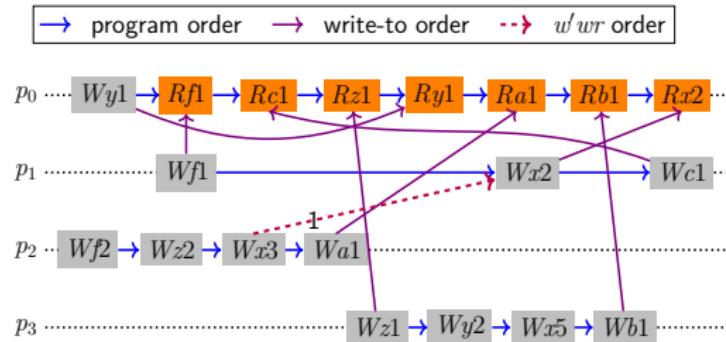


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w' wr$  规则.

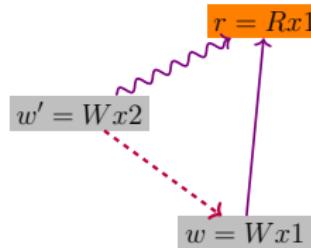


图:  $w' wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

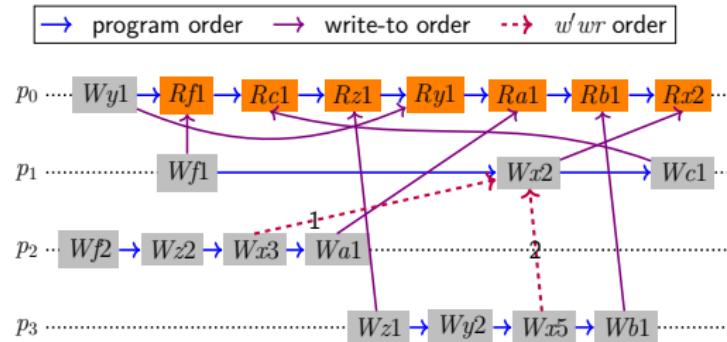


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w' wr$  规则.

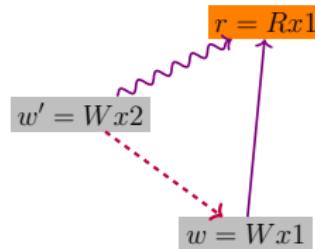


图:  $w' wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

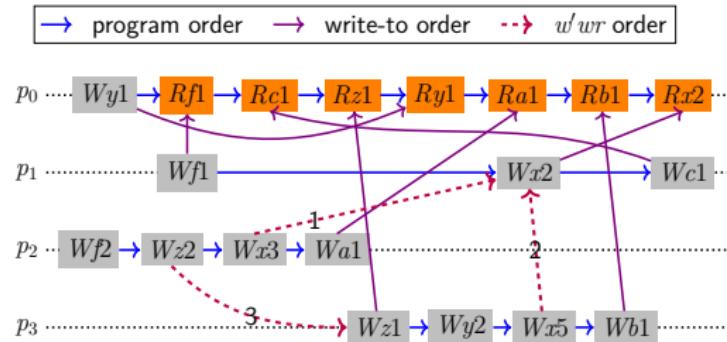


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w' wr$  规则.

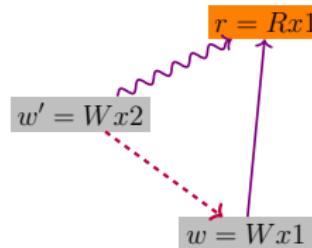


图:  $w' wr$  规则.

## VPC-MU 的多项式算法 RW-CLOSURE

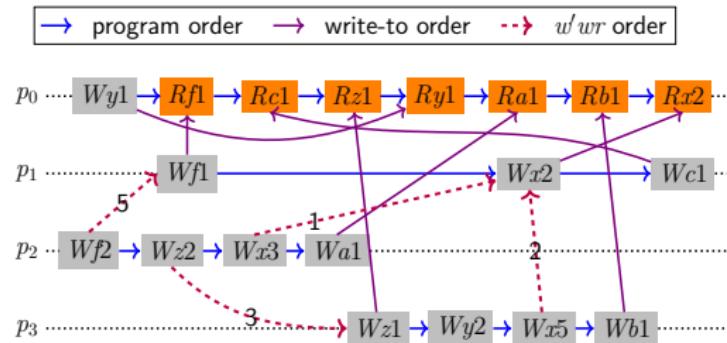


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

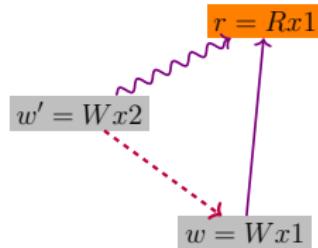


图:  $w'wr$  规则.

# VPC-MU 的多项式算法 RW-CLOSURE

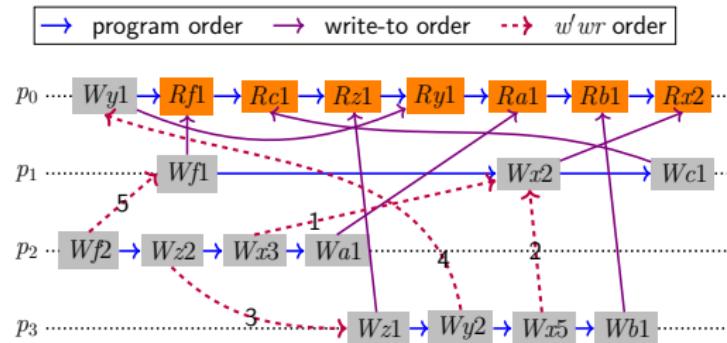


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用  $w'wr$  规则.

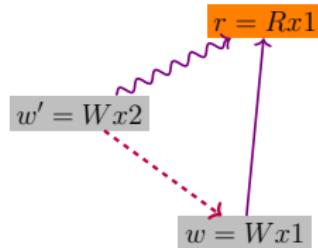


图:  $w'wr$  规则.

# 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.

证明

“ $\implies$ ” 反证法.

“ $\impliedby$ ” **难点:**  $DAG$  图蕴含着多个全序  
**技巧:** 对读操作作数学归纳, 构造合法调度

# VPC-MU 的多项式算法 RW-CLOSURE

定理 (RW-CLOSURE 算法正确性)

$$\begin{array}{c} \text{VPC-MU 实例满足 PRAM 一致性} \\ \iff \\ \text{RW-CLOSURE 算法所得图是 DAG 图.} \end{array}$$

证明

“ $\Rightarrow$ ” 反证法.

“ $\Leftarrow$ ” **难点:** DAG 图蕴含着多个全序  
**技巧:** 对读操作作数学归纳, 构造合法调度

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

两类负载:

1. 随机生成的系统执行
2. 满足 PRAM 一致性的系统执行 ( $\approx$  最坏情况输入)

# 实验评估

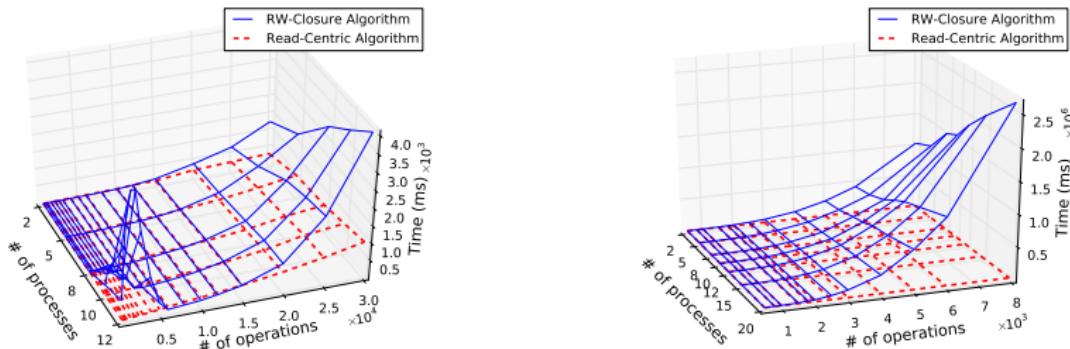


图: RW-CLOSURE 算法与 READ-CENTRIC 算法在 (左) 随机生成的执行及 (右) 满足 PRAM 一致性的执行上的运行时间。

# 实验评估

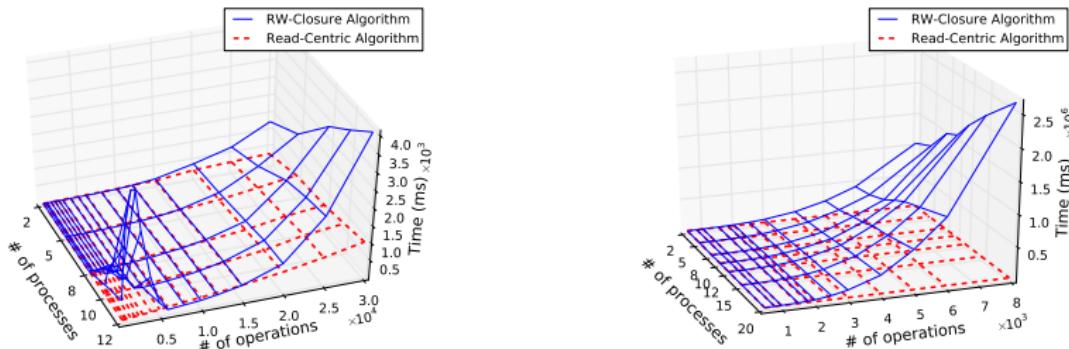


图: RW-CLOSURE 算法与 READ-CENTRIC 算法在 (左) 随机生成的执行及 (右) 满足 PRAM 一致性的执行上的运行时间。

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

# 实验评估

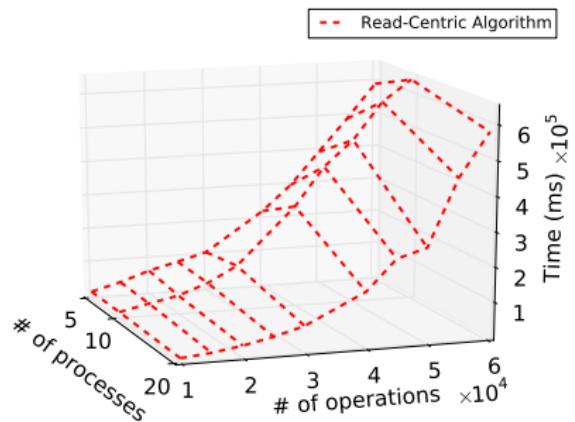


图: READ-CENTRIC 算法在满足 PRAM 一致性的执行上的运行时间.

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

# 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

## VPC 在相关工作中的意义

较早关注 (分布式系统领域) “弱一致性模型验证” 问题 (2013~):

**强一致性:** [Gibbons@SICOMP'97] [Cantin@TPDS'05] [Golab@PODC'11]

**弱一致性:** [Furbach@ACSD'14] [Bouajjani@arXiv'16]

# PA2AM 工作在技术框架中的位置

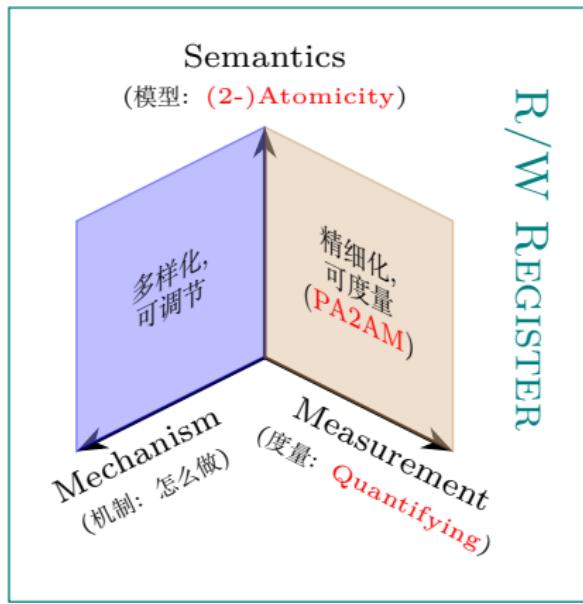


图: PA2AM — (2-)Atomicity 一致性维护与量化.

# 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 (Almost Strong Consistency):  
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 (Almost Strong Consistency):  
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 (Almost Strong Consistency):  
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 (Almost Strong Consistency):  
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 (Almost Strong Consistency):  
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 一致性

出租车实时位置查询一致性需求:

场景: 出租车实时更新位置信息 (副本)

权衡: 为了保证查询低延迟, 允许违反 atomicity

# PA2AM 一致性

出租车实时位置查询一致性需求:

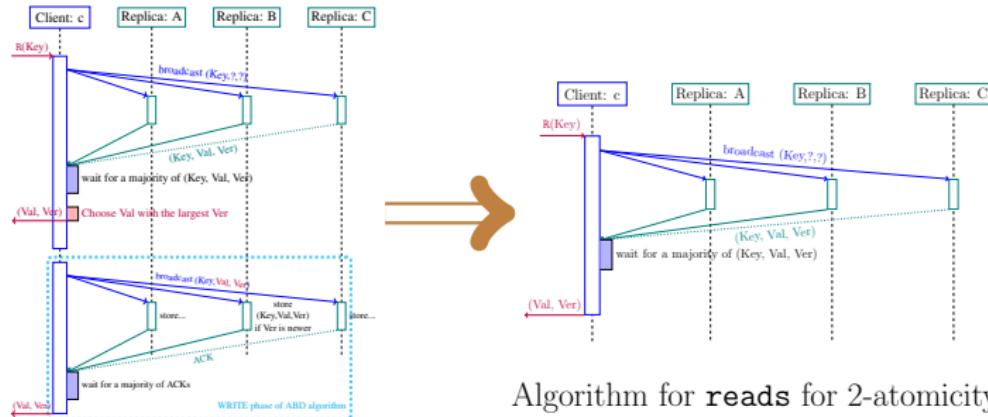
场景: 出租车实时更新位置信息 (副本)

权衡: 为了保证查询低延迟, 允许违反 atomicity

有界需求: 满足 2-atomicity

量化需求: 违反 atomicity 的读请求低于 1%

# PA2AM 维护算法



Algorithm for **reads** for atomicity.

Algorithm for **reads** for 2-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

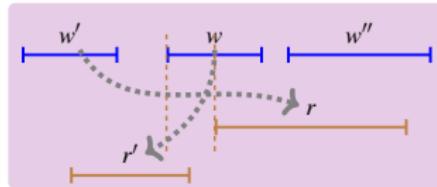
问题: PA2AM 维护算法在多大程度上违反了 atomicity?

定理 (充要条件)

PA2AM 维护算法违反 atomicity

$\iff$

存在 ONI (*old-new inversion*) [Attiya@JACM'95].



# PA2AM 量化分析

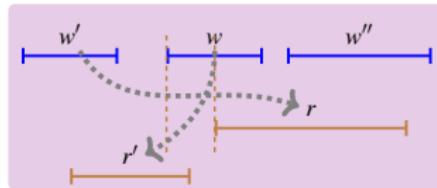
问题: PA2AM 维护算法在多大程度上违反了 atomicity?

## 定理 (充要条件)

*PA2AM 维护算法违反 atomicity*

$\iff$

*存在 ONI (old-new inversion) [Attiya@JACM'95].*



## 证明

分情况分析: 读写操作之间的偏序关系与 *reads-from* 关系.

# PA2AM 量化分析

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.  $r' = R(w)$ .

$$\text{ONI} \triangleq \text{CP} \cap \text{RWP}$$

# PA2AM 量化分析

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

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

# PA2AM 量化分析

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

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

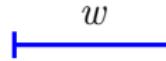
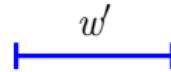
$$\begin{aligned}
 \mathbb{P}\{\text{CP} \mid R' = m\} &= \mathbb{P}(E_{N-1,m}) \\
 &= \sum_{k=0}^{N-2} \binom{N-1}{k} \binom{m-1}{N-k-2} p_0^k r^{N-k-1} s^m \\
 &\quad \mathbb{P}\{\text{RWP} \mid R' = m\} \\
 &\leq \mathbb{P}\{r \neq R(w)\} \times \left(1 - \mathbb{P}\{r' \neq R(w) \mid r \neq R(w)\}^m\right)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

# PA2AM 量化分析

带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :

# PA2AM 量化分析

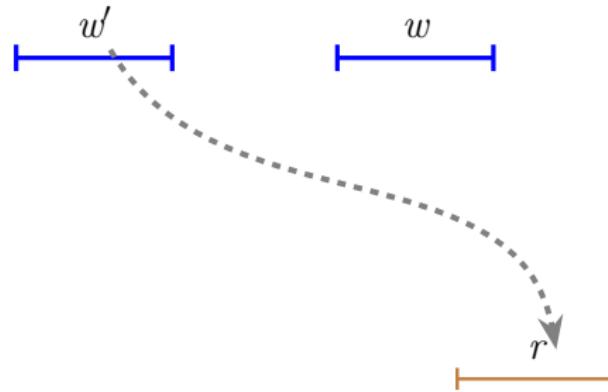
带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :



# PA2AM 量化分析

带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :

建模目的:  $R(r) = w' \ (R(r) \neq w)$

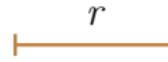


# PA2AM 量化分析

带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :

建模目的:  $R(r) = w' \ (R(r) \neq w)$

系统协议:  $n$  副本; “过半数” ( $M = \lfloor n/2 \rfloor + 1$ ) 通信规则



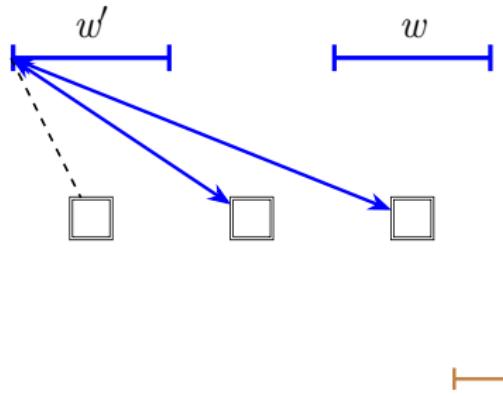
# PA2AM 量化分析

带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :

建模目的:  $R(r) = w' \ (R(r) \neq w)$

系统协议:  $n$  副本; “过半数” ( $M = \lfloor n/2 \rfloor + 1$ ) 通信规则

球盒模型: 操作产生  $n$  个球, 发送到  $n$  个副本盒子



# PA2AM 量化分析

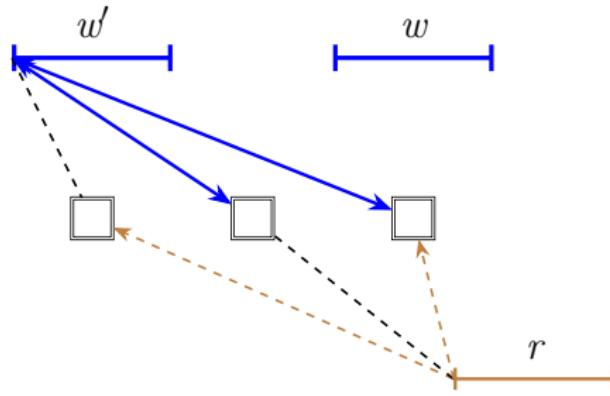
带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :

建模目的:  $R(r) = w'$  ( $R(r) \neq w$ )

系统协议:  $n$  副本; “过半数” ( $M = \lfloor n/2 \rfloor + 1$ ) 通信规则

球盒模型: 操作产生  $n$  个球, 发送到  $n$  个副本盒子

时刻  $t$ : 操作  $r$  恰好前  $M$  个球到达相应盒子



# PA2AM 量化分析

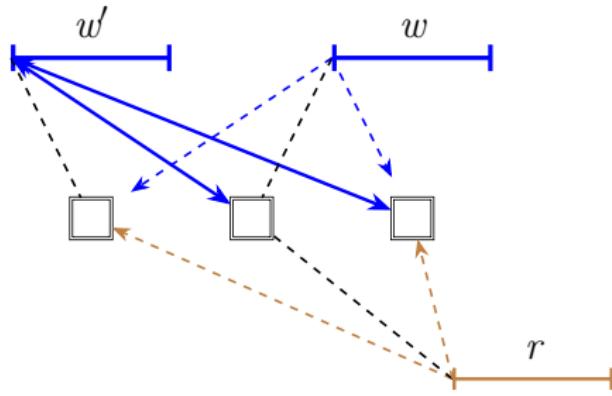
带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :

建模目的:  $R(r) = w'$  ( $R(r) \neq w$ )

系统协议:  $n$  副本; “过半数” ( $M = \lfloor n/2 \rfloor + 1$ ) 通信规则

球盒模型: 操作产生  $n$  个球, 发送到  $n$  个副本盒子

时刻  $t$ : 操作  $r$  恰好前  $M$  个球到达相应盒子



# PA2AM 量化分析

带时间的球盒模型, 计算  $\mathbb{P}(\text{RWP}|\text{CP})$ :

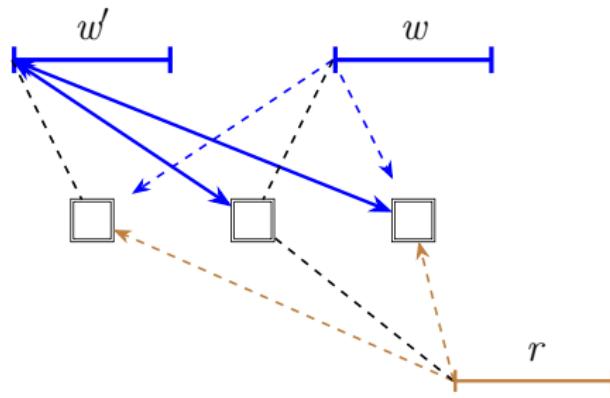
建模目的:  $R(r) = w' \ (R(r) \neq w)$

系统协议:  $n$  副本; “过半数” ( $M = \lfloor n/2 \rfloor + 1$ ) 通信规则

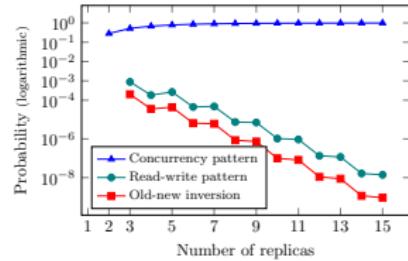
球盒模型: 操作产生  $n$  个球, 发送到  $n$  个副本盒子

时刻  $t$ : 操作  $r$  恰好前  $M$  个球到达相应盒子

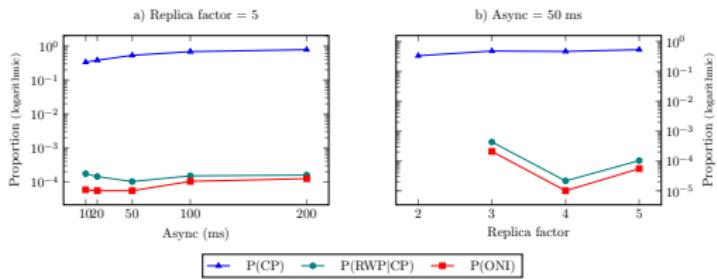
带时间: 操作起始时间 + 消息延迟



# PA2AM 量化分析

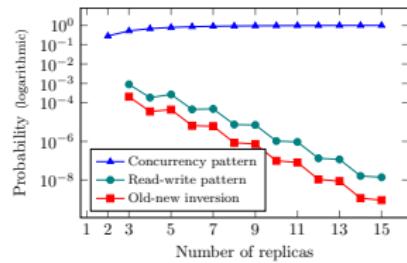


(a) 数值结果.

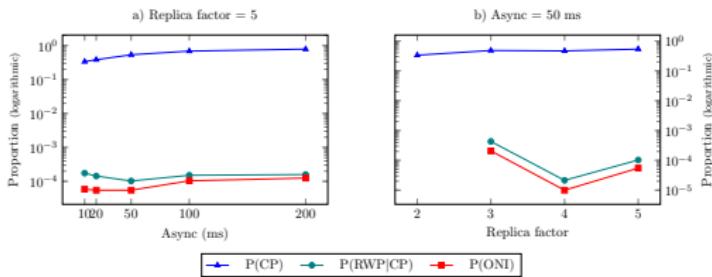


(b) 实验结果.

# PA2AM 量化分析



(a) 数值结果.

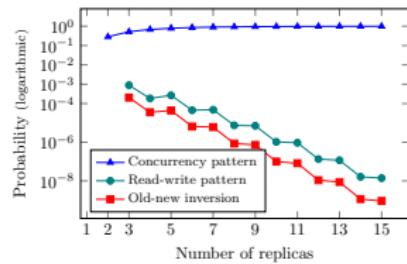


(b) 实验结果.

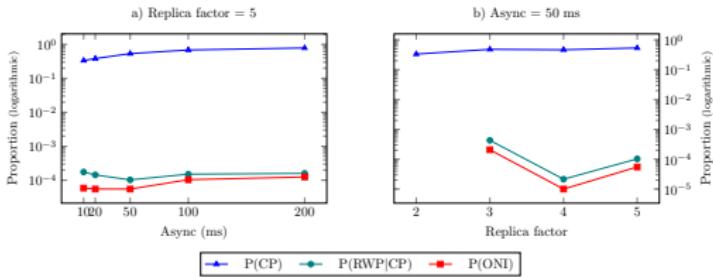
观察: (观察一)

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.

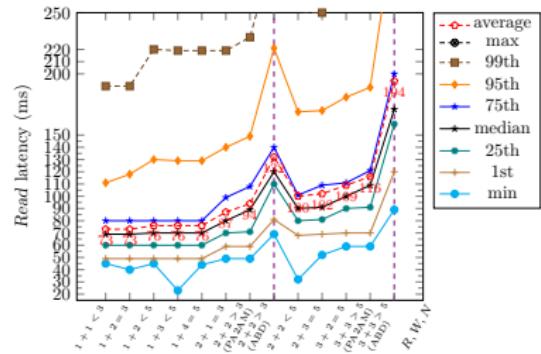
观察: (观察一)

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.

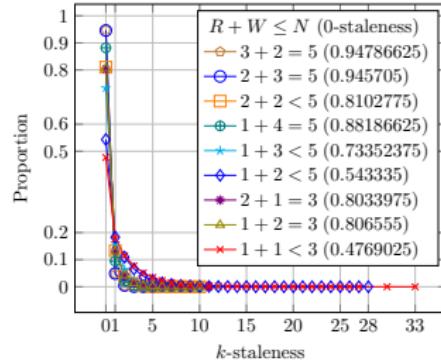
(b)  $k$ -陈旧度对比.

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	0.000085	0.00377875	0.002755	0.00406	0.0027225	0.0020275	0.002255	0.0003525

# PA2AM 的意义

PA2AM 可作为数据一致性/访问延迟权衡的一种可行选项

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

# PA2AM 的意义

## PA2AM 可作为数据一致性/访问延迟权衡的一种可行选项

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

## PA2AM 在相关工作中的意义

1. (AFAWK) 首次 “确定性陈旧度有界 + 概率”  
[Aiyer@DISC'05] [Lee@DC'05] [Bailis@VLDB'12]
2. 度量违反 atomicity (vs. regularity [Lee@DC'05] [Bailis@VLDB'12]) 的程度

# RVSI 工作在技术框架中的位置

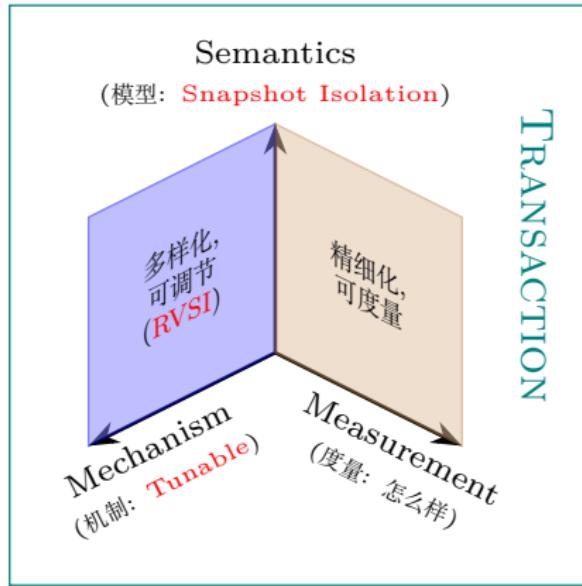


图: RVSI — Snapshot Isolation 一致性弱化与维护.

# 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:

SI [Berenson@SIGMOD'95]  $\xrightarrow{\text{relaxed}}$

PCSI [Elnekety@SRDS'05] 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: SI [Berenson@SIGMOD'95]  $\xrightarrow{\text{relaxed}}$   
PCSI [Elnekety@SRDS'05] 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]

# RVSI 定义

RVSI (Relaxed Version Snapshot Isolation) 定义原则:

- ▶ 参数  $k_1, k_2, k_3$  调节/控制相对于 SI 的“异常”程度

# RVSI 定义

RVSI (Relaxed Version Snapshot Isolation) 定义原则:

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

RC: Read Committed

SI: Snapshot Isolation

# RVSI 定义

SI: “**观察到事务开始 (2) 之前的 (1) 最新的系统 (3) 快照**”

定义 (RVSI (弱化 SI 的要点))

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

# RVSI 定义

在线书店应用:

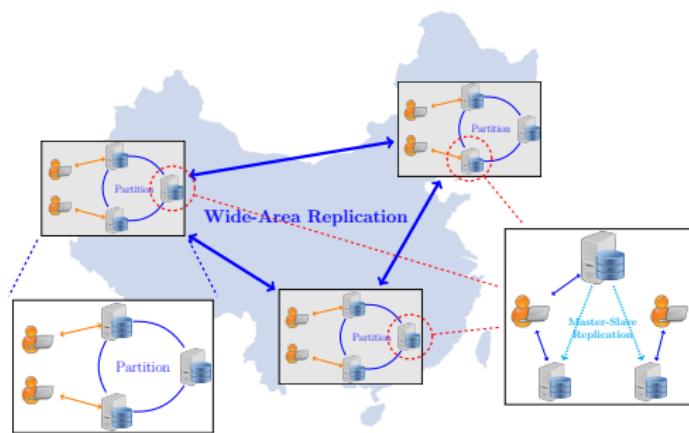
客户  $T_1$ : 查询某书信息  $\xrightarrow{k_1-BV}$  书评, 标签可以过时

职员  $T_2$ : 查询库存量  $\xrightarrow{k_2-FV}$  希望观察到并发的库存更新

分析师  $T_3$ : 销量 vs. 星级  $\xrightarrow{k_3-SV}$  不同系统快照

# CHAMELEON<sup>TKVS</sup><sup>1</sup>分布式事务键值存储原型系统设计

系统架构: 阿里云<sup>2</sup>多数据中心 ( $9 = 3 \times 3$ )



<sup>1</sup><https://github.com/hengxin/chameleon-transactional-kvstore>

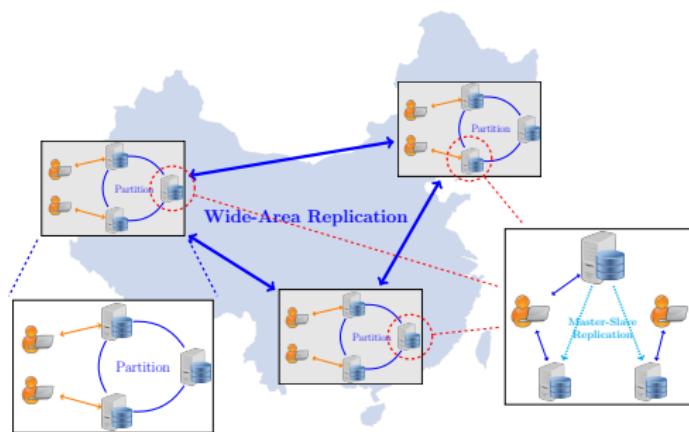
<sup>2</sup><https://www.aliyun.com/>

# CHAMELEON<sup>TKVS</sup><sup>1</sup>分布式事务键值存储原型系统设计

系统架构: 阿里云<sup>2</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>1</sup><https://github.com/hengxin/chameleon-transactional-kvstore>

<sup>2</sup><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

# RVSI 维护算法

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

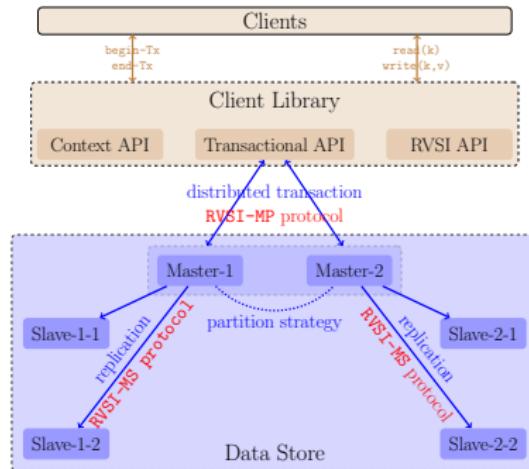
RVSI 维护算法:

- ▶ 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”

# RWSI 维护算法

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

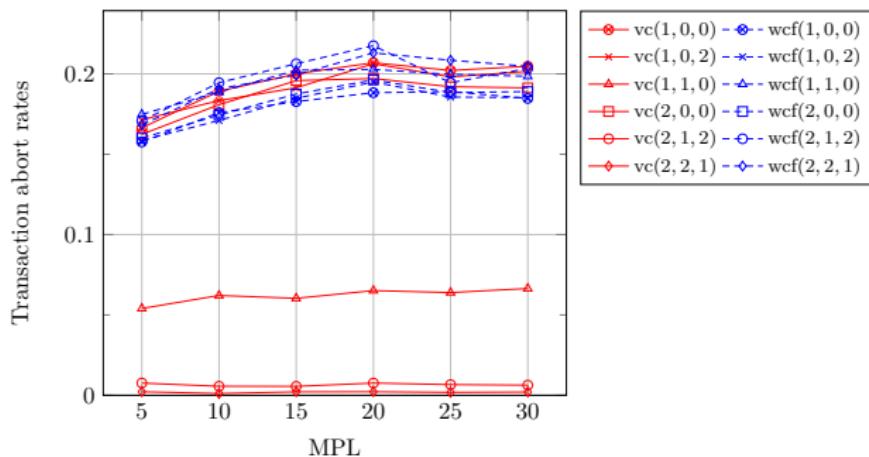


图: 读频繁 ( $rwRatio = 4:1$ ) 负载下 RVI 对事务中止率的影响.

# RVI 实验评估

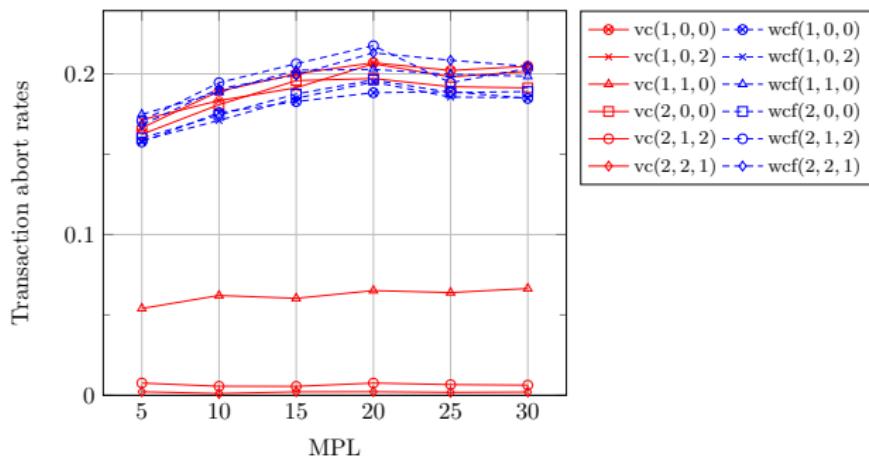


图: 读频繁 ( $rwRatio = 4:1$ ) 负载下 RVI 对事务中止率的影响.

wcf-aborted: 无显著变化 ( $mpl = 30$  时,  $wcf(1, 0, 0) = 0.184733$ )

# RVI 实验评估

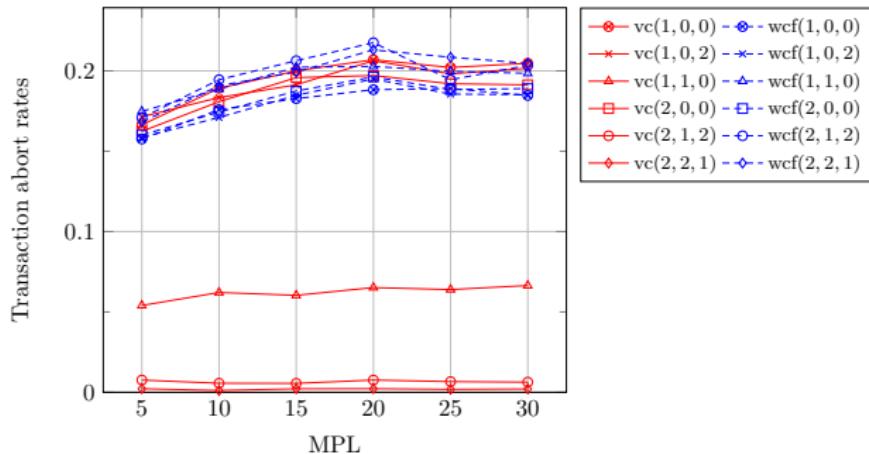


图: 读频繁 ( $rwRatio = 4:1$ ) 负载下 RVI 对事务中止率的影响.

wcf-aborted: 无显著变化 ( $mpl = 30$  时,  $wcf(1, 0, 0) = 0.184733$ )

vc-aborted: 显著减少 ( $mpl = 30$  时,  $vc(1, 0, 0) = 0.204733; vc(1, 1, 0) = 0.066433; vc(2, 2, 1) = 0.002033$ )

# RVI 实验评估

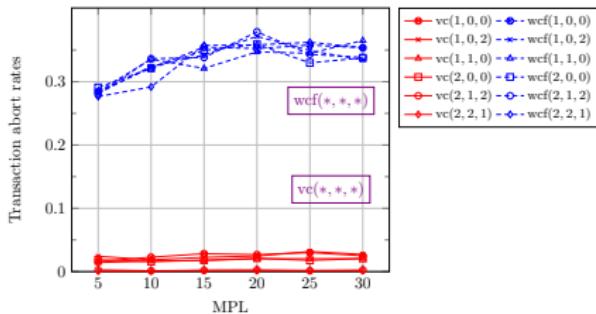


图: 写频繁 ( $rwRatio = 1:2$ ) 负载下 RVI 对事务中止率的影响.

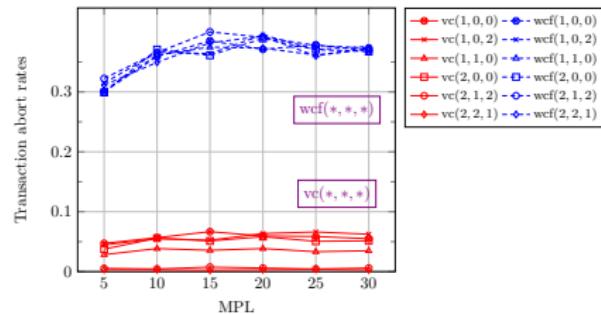


图: 读写相当 ( $rwRatio = 1:1$ ) 负载下 RVI 对事务中止率的影响.

wcf-aborted: 无显著变化

vc-aborted: 绝对数值小; 相对变化显著

# 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 主要成果

5 总结展望

# 工作总结

## 论文主要工作

# 工作总结

## 论文主要工作

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

# 工作总结

## 论文主要工作

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

框架: 提出包含“一个基础、三个维度”的技术框架

# 工作总结

## 论文主要工作

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

框架: 提出包含“一个基础、三个维度”的技术框架

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
- ▶ 论文评审与答辩老师
- ▶ 答辩秘书: 余萍副教授
- ▶ 软件所全体师生



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



# 分布数据一致性技术研究

6 附录: 参考文献

7 附录: 相关工作

# 参考文献 I

-  Daniel Abadi. "Consistency Tradeoffs in Modern Distributed Database System Design". In: *IEEE Computer* 45.2 (2012), pp. 37–42.
-  Sarita V. Adve and Kourosh Gharachorloo. "Shared Memory Consistency Models: A Tutorial". In: *Computer* 29.12 (Dec. 1996), pp. 66–76.
-  Atul Adya. "Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions". PhD thesis. Massachusetts Institute of Technology, 1999.
-  Amitanand Aiyer, Lorenzo Alvisi, and Rida A. Bazzi. "On the Availability of Non-strict Quorum Systems". In: *Proceedings of the 19th International Conference on Distributed Computing*. DISC '05. Springer-Verlag, 2005, pp. 48–62.

## 参考文献 II

-  Masoud Saeida Ardekani, Pierre Sutra, and Marc Shapiro. "Non-monotonic Snapshot Isolation: Scalable and Strong Consistency for Geo-replicated Transactional Systems". In: *Proceedings of the 32nd IEEE International Symposium on Reliable Distributed Systems*. SRDS '13. IEEE Computer Society, 2013, pp. 163–172.
-  Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. "Sharing Memory Robustly in Message-passing Systems". In: *J. ACM* 42.1 (Jan. 1995), pp. 124–142.
-  Hagit Attiya and Roy Friedman. "A Correctness Condition for High-Performance Multiprocessors". In: *SIAM Journal on Computing* 27.6 (1998), pp. 1637–1670.
-  Peter Bailis et al. "Highly Available Transactions: Virtues and Limitations". In: *Proc. VLDB Endow.* 7.3 (Nov. 2013), pp. 181–192.
-  Peter Bailis et al. "Probabilistically Bounded Staleness for Practical Partial Quorums". In: *Proc. VLDB Endow.* 5.8 (2012), pp. 776–787.

# 参考文献 III

-  Surender Baswana, Shashank K. Mehta, and Vishal Powar. "Implied Set Closure and Its Application to Memory Consistency Verification". In: *Proceedings of the 20th International Conference on Computer Aided Verification*. CAV '08. 2008, pp. 94–106.
-  Hal Berenson et al. "A Critique of ANSI SQL Isolation Levels". In: *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*. SIGMOD '95. ACM, 1995, pp. 1–10.
-  Philip A. Bernstein and Nathan Goodman. "Multiversion Concurrency Control — Theory and Algorithms". In: *ACM Trans. Database Syst.* 8.4 (Dec. 1983), pp. 465–483.
-  Philip A. Bernstein et al. "Relaxed-currency Serializability for Middle-tier Caching and Replication". In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. SIGMOD '06. ACM, 2006, pp. 599–610.

# 参考文献 IV

-  A. Bouajjani et al. "On Verifying Causal Consistency". In: *ArXiv e-prints* (Nov. 2016). arXiv: 1611.00580 [cs.LO].
-  Eric Brewer. "Towards robust distributed systems". In: *Proceedings of the annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (Invited Talk)*. PODC '00. July 2000.
-  Jake Brutlag. *Speed Matters for Google Web Search*. Accessed: 07-08-2016.
-  J Brzezinski, C Sobaniec, and D Wawrzyniak. "From session causality to causal consistency". In: *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing*. 2004, pp. 152–158.
-  Sebastian Burckhardt et al. "Replicated Data Types: Specification, Verification, Optimality". In: *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '14. ACM, 2014, pp. 271–284.

# 参考文献 V

-  Michael J. Cahill, Uwe Röhm, and Alan D. Fekete. "Serializable Isolation for Snapshot Databases". In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. ACM, 2008, pp. 729–738.
-  Jason F. Cantin, Mikko H. Lipasti, and James E. Smith. "The complexity of verifying memory coherence and consistency". In: *IEEE Transactions on Parallel and Distributed Systems* 16.7 (July 2005), pp. 663–671.
-  Giuseppe DeCandia et al. "Dynamo: Amazon's Highly Available Key-value Store". In: *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*. SOSP '07. ACM, 2007, pp. 205–220.
-  Michel Dubois, Christoph Scheurich, and Fayé A. Briggs. "Memory Access Buffering in Multiprocessors". In: *Proceedings of the 13th Annual International Symposium on Computer Architecture*. ISCA '86. IEEE Computer Society Press, 1986, pp. 434–442.

# 参考文献 VI

-  Partha Dutta et al. "How Fast Can a Distributed Atomic Read Be?" In: *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing*. 2004, pp. 236–245.
-  Sameh Elnikety, Willy Zwaenepoel, and Fernando Pedone. "Database Replication Using Generalized Snapshot Isolation". In: *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems*. SRDS '05. IEEE Computer Society, 2005, pp. 73–84.
-  Alan Fekete, Shirley N. Goldrei, and Jorge Pérez Asenjo. "Quantifying Isolation Anomalies". In: *Proc. VLDB Endow.* 2.1 (Aug. 2009), pp. 467–478.
-  Alan Fekete et al. "Making Snapshot Isolation Serializable". In: *ACM Trans. Database Syst.* 30.2 (June 2005), pp. 492–528.
-  F. Furbach et al. "Memory Model-Aware Testing - A Unified Complexity Analysis". In: *2014 14th International Conference on Application of Concurrency to System Design*. June 2014, pp. 92–101.

# 参考文献 VII

-  Phillip B. Gibbons and Ephraim Korach. "Testing shared memories". In: *SIAM J. Comput.* 26.4 (Aug. 1997), pp. 1208–1244.
-  Wojciech Golab, Jeremy Hurwitz, and Xiaozhou (Steve) Li. "On the k-Atomicity-Verification Problem". In: *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems*. ICDCS '13. IEEE Computer Society, 2013, pp. 591–600.
-  Wojciech Golab, Xiaozhou Li, and Mehul A. Shah. "Analyzing Consistency Properties for Fun and Profit". In: *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. PODC '11. ACM, 2011, pp. 197–206.
-  Wojciech Golab et al. "Client-Centric Benchmarking of Eventual Consistency for Cloud Storage Systems". In: *Proceedings of the 34th IEEE International Conference on Distributed Computing Systems*. ICDCS '14. IEEE Computer Society, 2014, pp. 493–502.

# 参考文献 VIII

-  Wojciech Golab et al. "Computing Weak Consistency in Polynomial Time: [Extended Abstract]". In: *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. PODC '15. ACM, 2015, pp. 395–404.
-  Vincent Gramoli and Michel Raynal. "Timed Quorum Systems for Large-scale and Dynamic Environments". In: *Proceedings of the 11th International Conference on Principles of Distributed Systems*. OPODIS '07. Springer-Verlag, 2007, pp. 429–442.
-  Rachid Guerraoui, Matej Pavlovic, and Dragos-Adrian Seredinschi. "Trade-offs in Replicated Systems". In: *IEEE Data Eng. Bull.* 39.1 (Mar. 2016), pp. 14–26.
-  Sudheendra Hangal et al. "TSOtool: A Program for Verifying Memory Systems Using the Memory Consistency Model". In: *Proceedings of the 31st Annual International Symposium on Computer Architecture*. ISCA '04. June 2004.
-  Ramakrishna Kotla et al. *Transactions with Consistency Choices on Geo-Replicated Cloud Storage*. Tech. rep. Microsoft, MSR-TR-2013-82, Sept. 2013.

# 参考文献 IX

-  Rivka Ladin et al. "Providing High Availability Using Lazy Replication". In: *ACM Trans. Comput. Syst.* 10.4 (Nov. 1992), pp. 360–391.
-  Avinash Lakshman and Prashant Malik. "Cassandra: A Decentralized Structured Storage System". In: *SIGOPS Oper. Syst. Rev.* 44.2 (Apr. 2010), pp. 35–40.
-  Leslie Lamport. "How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs". In: *IEEE Trans. Comput.* 28.9 (Sept. 1979), pp. 690–691.
-  Leslie Lamport. "On interprocess communication (Part II: algorithms)". In: *Distrib. Comput.* 1.2 (1986), pp. 86–101.
-  Hyunyoung Lee and Jennifer L. Welch. "Randomized Registers and Iterative Algorithms". In: *Distrib. Comput.* 17.3 (2005), pp. 209–221.

# 参考文献 X

-  Cheng Li et al. "Making Geo-replicated Systems Fast As Possible, Consistent when Necessary". In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*. OSDI '12. USENIX Association, 2012, pp. 265–278.
-  Chaiyasit Manovit and Sudheendra Hangal. "Efficient Algorithms for Verifying Memory Consistency". In: *Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA '05. ACM, 2005, pp. 245–252.
-  Christos H. Papadimitriou. "The Serializability of Concurrent Database Updates". In: *J. ACM* 26.4 (Oct. 1979), pp. 631–653.
-  Amitabha Roy et al. "Fast and generalized polynomial time memory consistency verification". In: *Proceedings of the 18th International Conference on Computer Aided Verification*. CAV '06. 2006, pp. 503–516.
-  Yasushi Saito and Marc Shapiro. "Optimistic Replication". In: *ACM Comput. Surv.* 37.1 (Mar. 2005), pp. 42–81.

# 参考文献 XI

-  Yair Sovran et al. "Transactional Storage for Geo-replicated Systems". In: *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*. SOSP '11. ACM, 2011, pp. 385–400.
-  Robert C. Steinke and Gary J. Nutt. "A Unified Theory of Shared Memory Consistency". In: *J. ACM* 51.5 (Sept. 2004), pp. 800–849.
-  Doug Terry. "Replicated Data Consistency Explained Through Baseball". In: *Commun. ACM* 56.12 (2013), pp. 82–89.
-  Douglas B. Terry et al. "Consistency-based Service Level Agreements for Cloud Storage". In: *Proceedings of the 24th ACM Symposium on Operating Systems Principles*. SOSP '13. ACM, 2013, pp. 309–324.
-  Douglas B. Terry et al. "Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System". In: *Proceedings of the 15th ACM Symposium on Operating Systems Principles*. SOSP '95. ACM, 1995, pp. 172–182.

# 参考文献 XII

-  Anand Tripathi and BhagavathiDhass Thirunavukarasu. "A Transaction Model for Management of Replicated Data with Multiple Consistency Levels". In: *Proceedings of the 2015 IEEE International Conference on Big Data*. BigData'15. IEEE Computer Society, 2015, pp. 470–477.
-  Mihalis Yannakakis. "Serializability by Locking". In: *J. ACM* 31.2 (Mar. 1984), pp. 227–244.
-  Haifeng Yu. "Overcoming the Majority Barrier in Large-Scale Systems". In: *Proceedings of the 17th International Conference on Distributed Computing*. DISC' 03. Springer Berlin Heidelberg, 2003, pp. 352–366.
-  Haifeng Yu and Amin Vahdat. "Design and Evaluation of a Conit-based Continuous Consistency Model for Replicated Services". In: *ACM Trans. Comp. Syst.* 20.3 (2002), pp. 239–282.

# 参考文献 XIII



Kamal Zellag and Bettina Kemme. "Real-time Quantification and Classification of Consistency Anomalies in Multi-tier Architectures". In: *Proceedings of the 27th IEEE International Conference on Data Engineering*. ICDE '11. IEEE Computer Society, 2011, pp. 613–624.

# 分布数据一致性技术研究

6 附录: 参考文献

7 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	分布式系统
“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	分布式系统
“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	
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	
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@TPDS'05]
- ▶ VTSO (Total Store Order) [Hangal@ISCA'04] [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@TPDS'05]
- ▶ VTSO (Total Store Order) [Hangal@ISCA'04] [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@TPDS'05]
- ▶ VTSO (Total Store Order) [Hangal@ISCA'04] [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		