

复制数据类型

(CCF 优博论坛)

魏恒峰

南京大学软件所

2018 年 08 月 08 日



复制数据类型

① 研究背景

② 主要成果

复制数据类型

① 研究背景

② 主要成果

复制数据类型

Abstract Data Types (ADT) [Liskov@VHLL'74]
(单线程; 顺序语义)

Concurrent Data Types (CDT) [Herlihy@TOPLAS'90]
(多线程; 并发语义)

Replicated Data Types (RDT; \approx 2010 年) [Burckhardt@POPL'14]
(多副本; 复制语义)

Replicated Data Types: Specification, Verification, Optimality

Sebastian Burckhardt

Alexey Gotsman

Hongseok Yang

Marek Zawirski

新平台, 新机遇, 新挑战

大规模分布式应用



新浪微博社交应用¹:

- ▶ 日均用户近一亿名
- ▶ 日均消息近一亿条

¹2015 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¹:

- ▶ 日均用户近一亿名
- ▶ 日均消息近一亿条

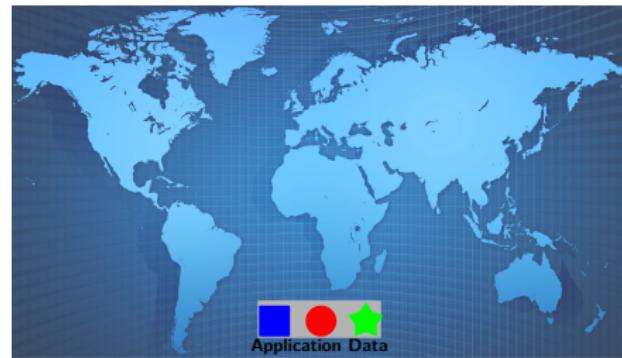
底层数据服务系统特性需求 (H^3L):

- ▶ 低延迟, 高可用性 (4 个 9²)
- ▶ 高容错性, 高可扩展性

¹ 2015 第三季度; 数据来自 China Internet Watch.

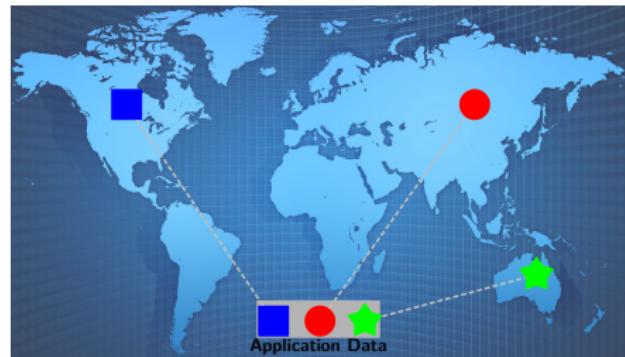
² 数据来自 InfoQ.

分布数据



分布数据 (distributed data):

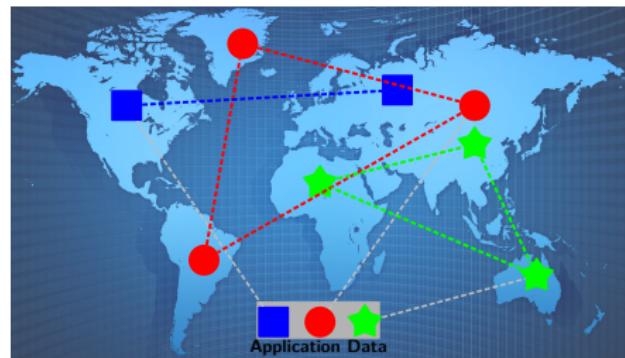
分布数据



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

复制数据类型

1 研究背景

2 主要成果

VPC 工作在技术框架中的位置

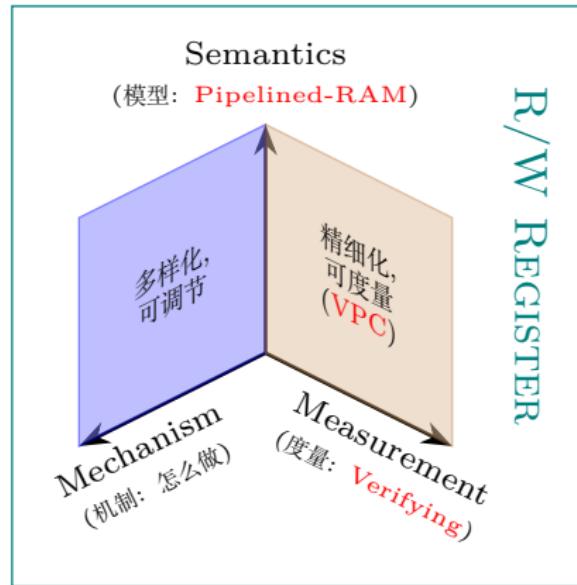


图: VPC — Pipelined-RAM (PRAM) 一致性验证.

研究动机

问题: 为什么要验证 PRAM 一致性?

验证: 测试/确认系统是否提供了其所声称的数据一致性

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

研究动机

问题: 为什么要验证 PRAM 一致性?

验证: 测试/确认系统是否提供了其所声称的数据一致性

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

PRAM: 包含存储系统常提供的“会话”一致性

[Terry@PDIS'94] [Brzeziński@PDP'04]

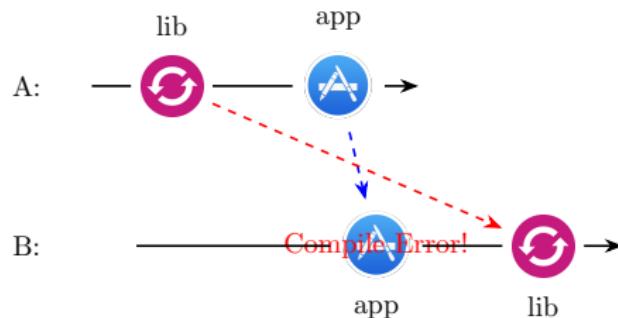


图: PRAM 保证“单调写”性质 [Terry@PDIS'94] [Brzeziński@PDP'04].

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})?

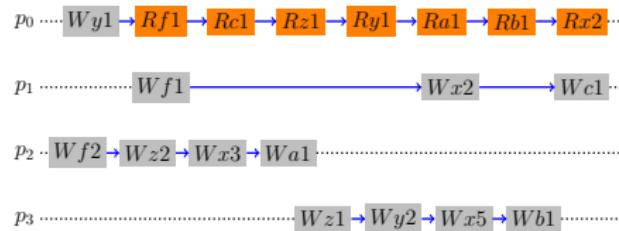
$$e \in \mathcal{C} \Rightarrow \{0, 1\}?$$

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



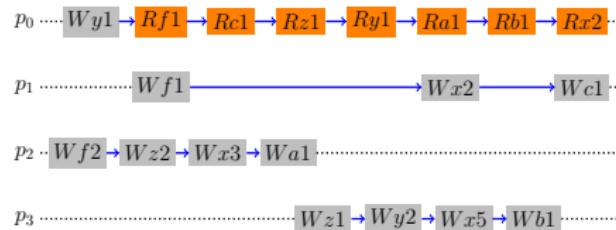
VPC 问题定义

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.



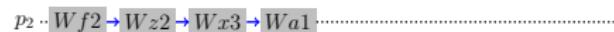
VPC 问题定义

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.



$p_0 : W f 2 \ W f 1 \ W z 2 \ W z 1 \ W y 2 \ W y 1 \ R f 1 \ W x 5 \ W x 3 \ W x 2 \ W c 1 \ R c l$
 $R z 1 \ R y 1 \ W a 1 \ R a 1 \ W b 1 \ R b 1 \ R x 2$

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 (NP-complete) [*]	VPC-MD (NP-complete) [*]
<i>write (U)nique value</i>	VPC-SU (P) [Golab@PODC'11]	VPC-MU (P) [*]

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 (NP-complete) [*]	VPC-MD (NP-complete) [*]
<i>write (U)nique value</i>	VPC-SU (P) [Golab@PODC'11]	VPC-MU (P) [*]

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.

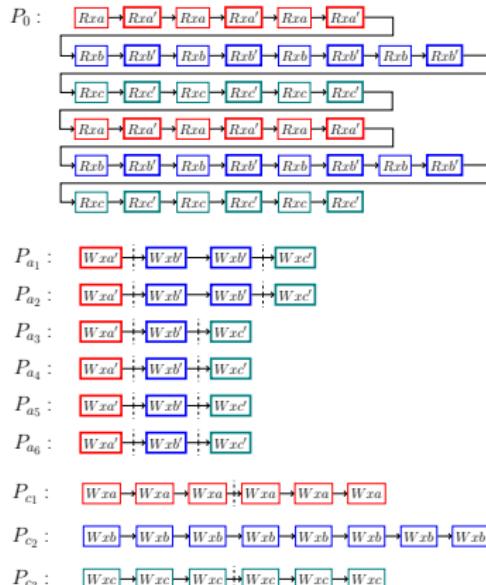


图: 对应于 UNARY 3-PARTITION 实例 $A = \{2, 2, 1, 1, 1, 1\}$, $m = 2, B = 4$ 的 VPC 执行.

VPC-MU 的多项式算法 RW-CLOSURE

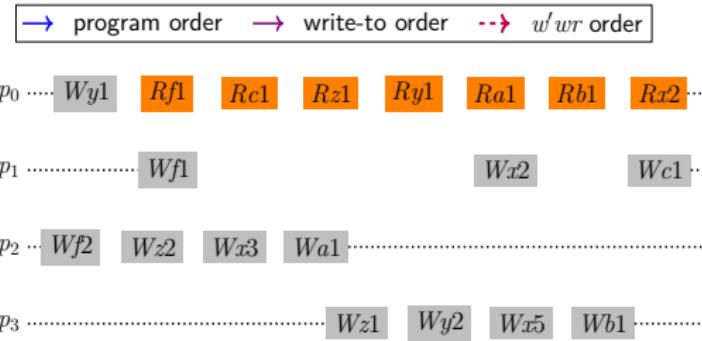


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w'wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

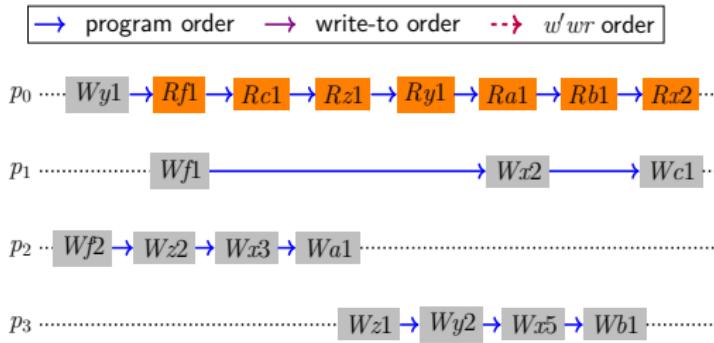


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w'wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

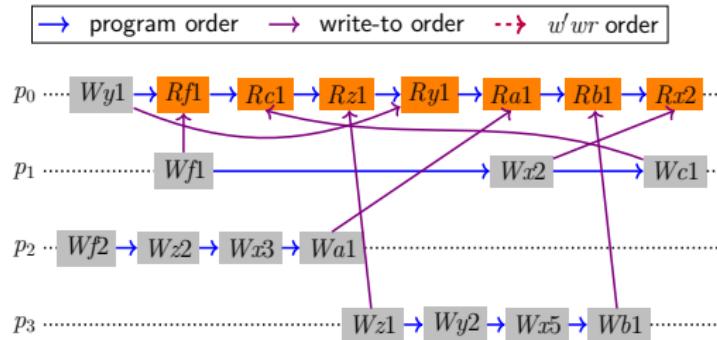


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w' wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

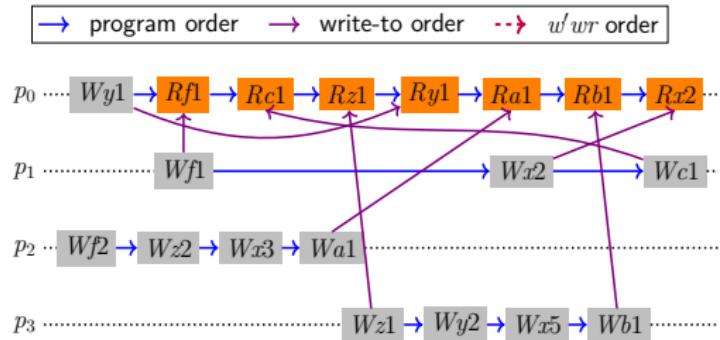


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w'wr$ 规则.

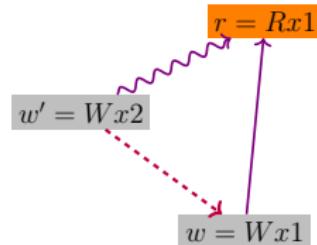


图: $w'wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

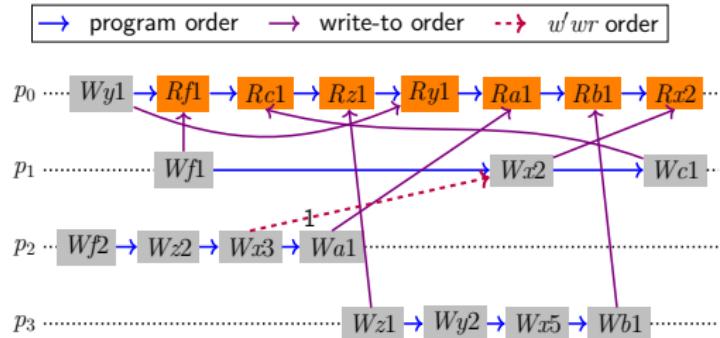


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w'wr$ 规则.

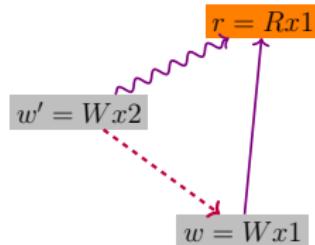


图: $w'wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

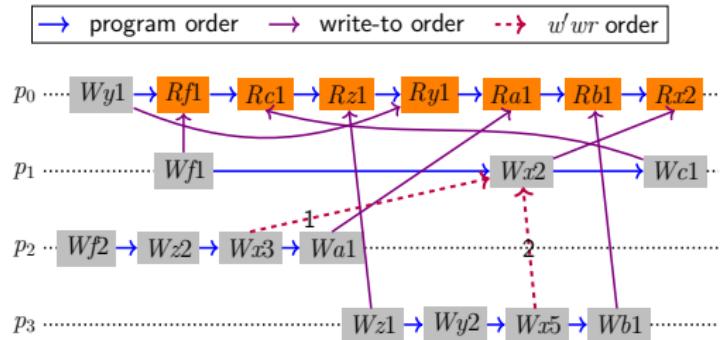


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w'wr$ 规则.

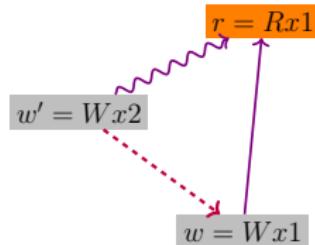


图: $w'wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

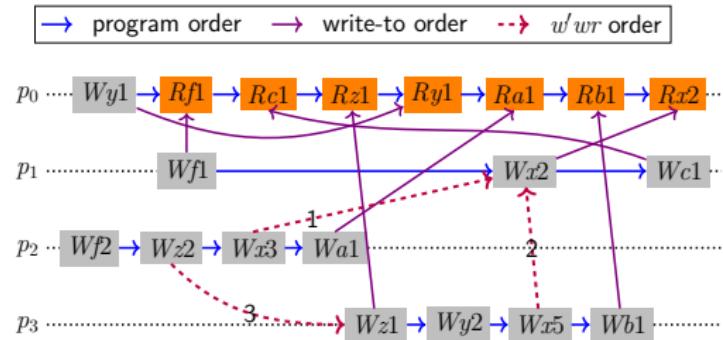


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w' wr$ 规则.

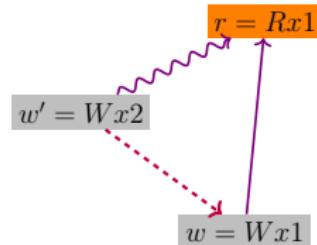


图: $w'wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

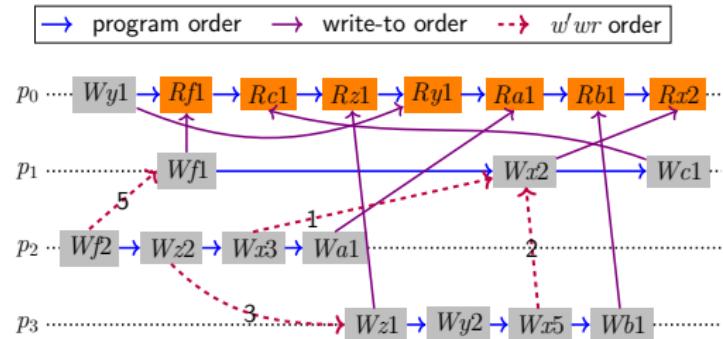


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w' wr$ 规则.

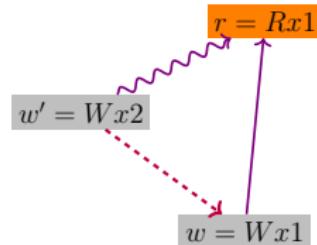


图: $w'wr$ 规则.

VPC-MU 的多项式算法 RW-CLOSURE

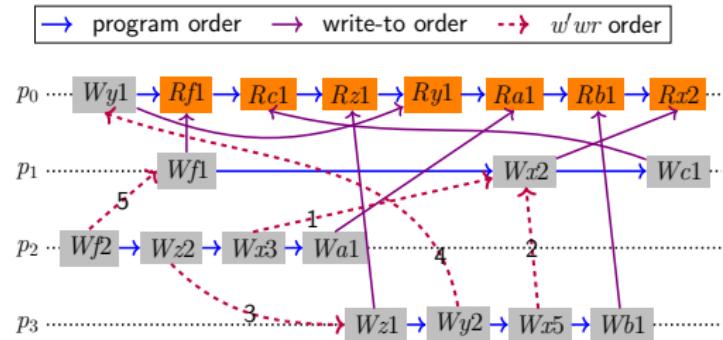


图: RW-CLOSURE 算法示例: 在传递闭包之上迭代应用 $w'wr$ 规则.

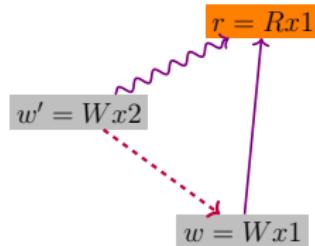


图: $w'wr$ 规则.

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



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

“ \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

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

定理 (READ-CENTRIC 算法正确性)

$VPC-MU$ 实例满足 $PRAM$ 一致性



READ-CENTRIC 算法所得图是 DAG 图.

VPC-MU 的多项式算法 READ-CENTRIC

定理 (READ-CENTRIC 算法正确性)

VPC-MU 实例满足 PRAM 一致性



READ-CENTRIC 算法所得图是 DAG 图.

证明

READ-CENTRIC $\xleftrightarrow{\text{Reachability}}$ RW-CLOSURE

难点: $\#w'wr_{\text{READ-CENTRIC}} \leq \#w'wr_{\text{RW-CLOSURE}}$

VPC-MU 的多项式算法 READ-CENTRIC

READ-CENTRIC 算法复杂度:

$$\underbrace{O(n)}_{\text{iterations}} \cdot \underbrace{O(n \cdot n^2)}_{\text{TOPO-SCHEDULE}} = O(n^4)$$

引理 (TOPO-SCHEDULE 的非迭代性)

设 TOPO-SCHEDULE 正在处理读操作 r ,
则局部子图中的每个写操作最多只有一次机会
在满足规则 $w'wr$ 的三元组中扮演 “ w' 角色”.

实验评估

实验目的¹：

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

¹机器配置: Intel Core i7 3.40GHZ, 4GB RAM.

实验评估

实验目的¹：

1. 考察 READ-CENTRIC 算法的实际效率 (*vs.* 渐近时间复杂度)
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)

¹机器配置: Intel Core i7 3.40GHZ, 4GB RAM.

实验评估

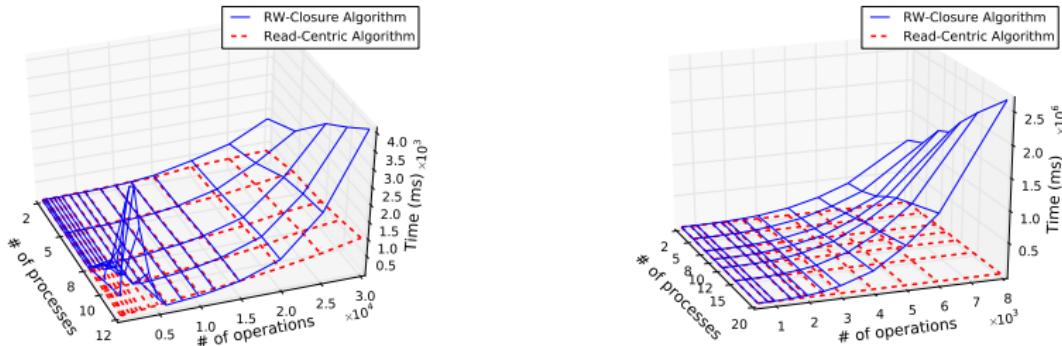


图: RW-CLOSURE 算法与 READ-CENTRIC 算法在 (左) 随机生成的执行及 (右) 满足 PRAM 一致性的执行上的运行时间。

实验评估

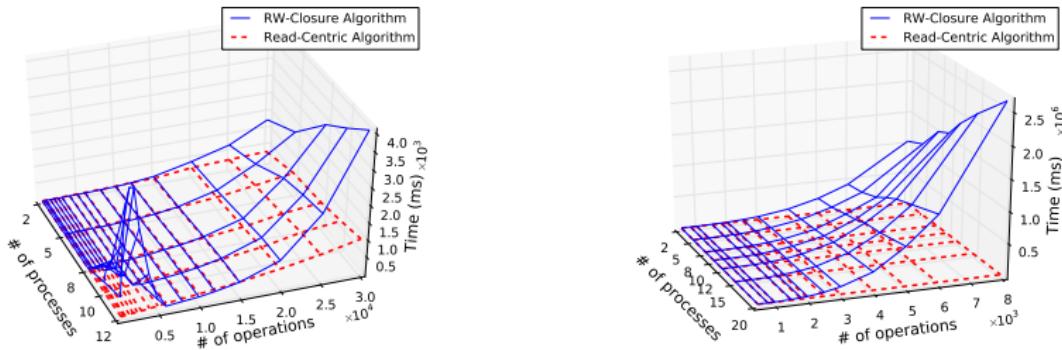


图: RW-CLOSURE 算法与 READ-CENTRIC 算法在 (左) 随机生成的执行及 (右) 满足 PRAM 一致性的执行上的运行时间。

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

实验评估

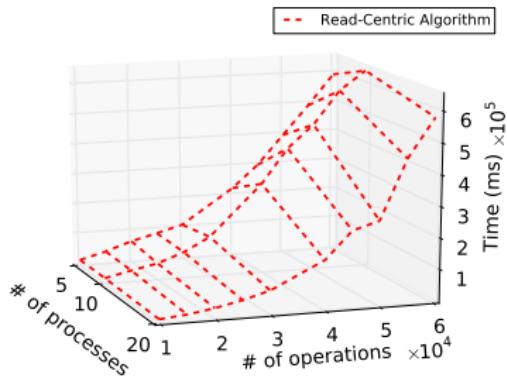


图: READ-CENTRIC 算法在满足 PRAM 一致性的执行上的运行时间.

READ-CENTRIC: 20 个进程、60,000 个操作 < 600s ¹

RW-CLOSURE: 20 个进程、8,000 个操作 > 3,000s

¹ 用于测试，规模可用。

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

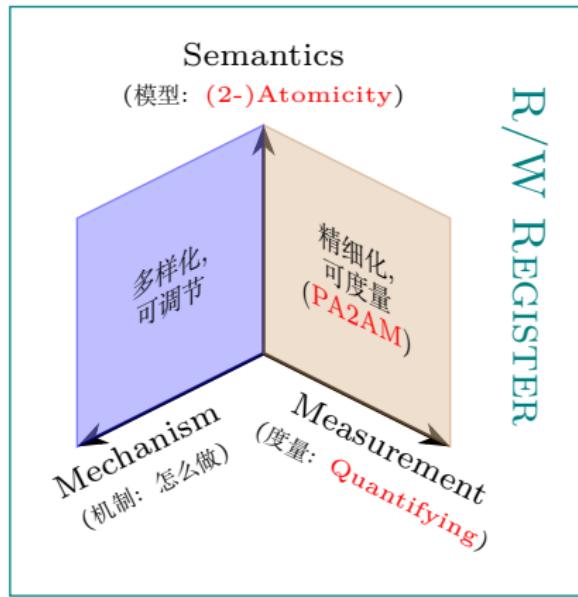


图: PA2AM — (2-)Atomicity 一致性维护与量化.

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

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 $k \leq 2$

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:

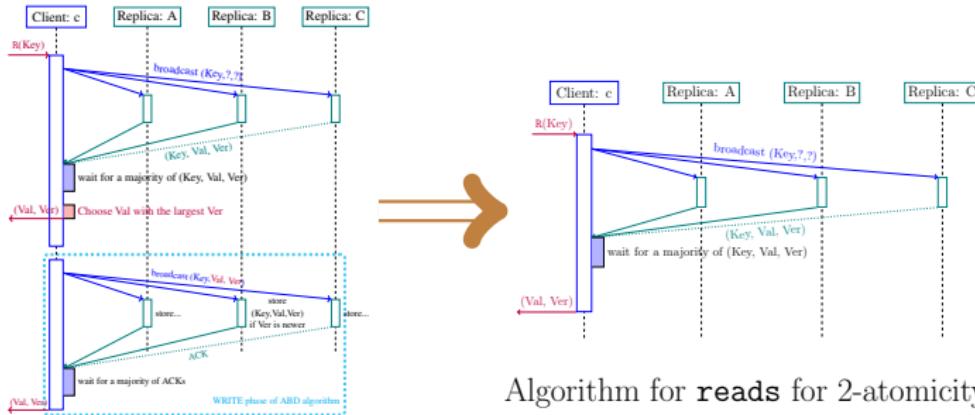
场景: 出租车实时更新位置信息 (副本)

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?

PA2AM 量化分析

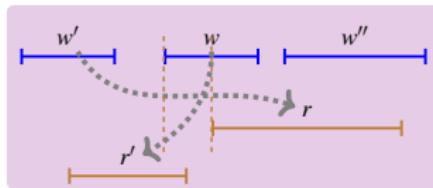
问题: PA2AM 维护算法在多大程度上违反了 atomicity?

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. $w' \text{ immediately precedes } w$;
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 $\mathbb{P}(\text{RWP}|\text{CP})$

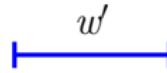
$$\begin{aligned}
 \mathbb{P}\{\text{CP} \mid R' = m\} &= \mathbb{P}(E_{N-1,m}) \\
 &= \sum_{k=0}^{N-2} \binom{N-1}{k} \binom{m-1}{N-k-2} p_0^k r^{N-k-1} s^m \\
 &\quad \mathbb{P}\{\text{RWP} \mid R' = m\} \\
 &\leq \mathbb{P}\{r \neq R(w)\} \times \left(1 - \mathbb{P}\{r' \neq R(w) \mid r \neq R(w)\}^m\right)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

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

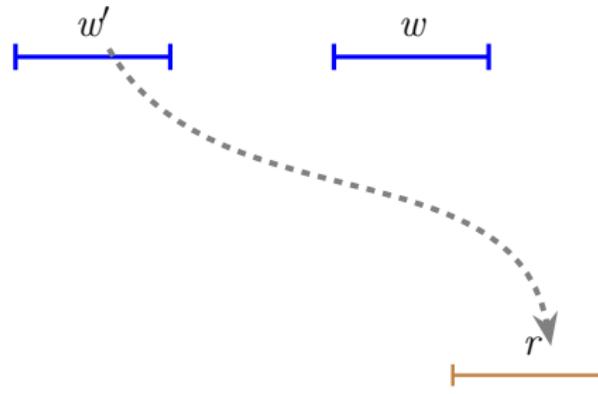
带时间的球盒模型, 计算 $\mathbb{P}(\text{RWP}|\text{CP})$:



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: $\mathbb{P}\{R(r) = w'\} \ (R(r) \neq w)$

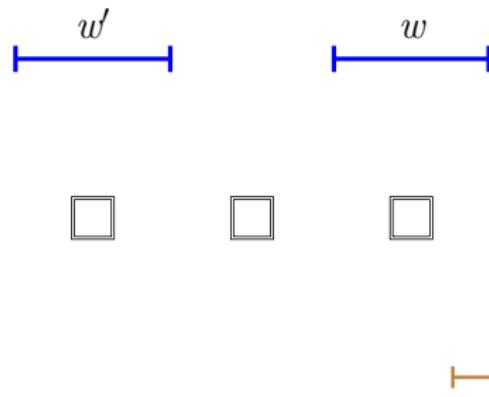


PA2AM 量化分析

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

建模目的: $\mathbb{P}\{R(r) = w'\} \ (R(r) \neq w)$

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



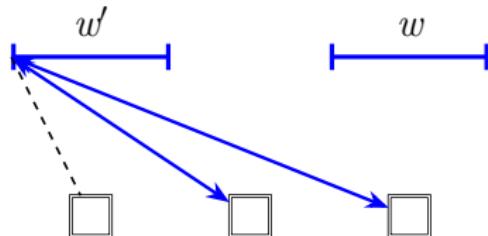
PA2AM 量化分析

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

建模目的: $\mathbb{P}\{R(r) = w'\}$ ($R(r) \neq w$)

系统协议: n 副本节点; “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

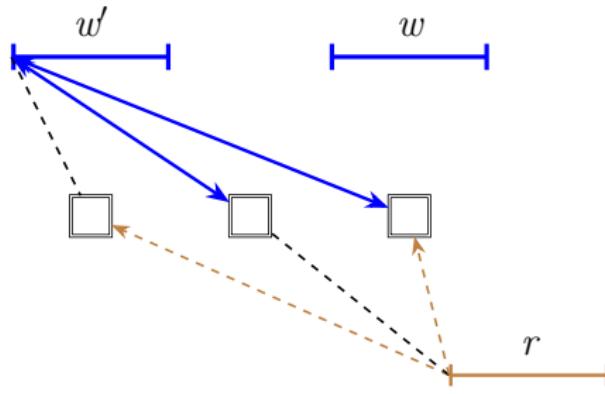
带时间的球盒模型, 计算 $\mathbb{P}(\text{RWP}|\text{CP})$:

建模目的: $\mathbb{P}\{R(r) = w'\}$ ($R(r) \neq w$)

系统协议: n 副本节点; “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 $\mathbb{P}(\text{RWP}|\text{CP})$:

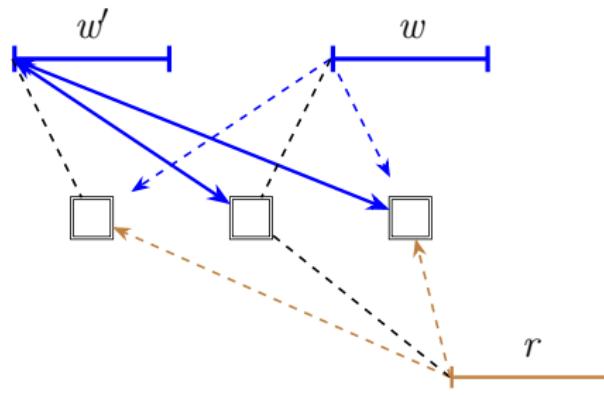
建模目的: $\mathbb{P}\{R(r) = w'\}$ ($R(r) \neq w$)

系统协议: n 副本节点; “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

$R(r) \neq w$: 操作 w 相应 M 个球尚在途中



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: $\mathbb{P}\{R(r) = w'\}$ ($R(r) \neq w$)

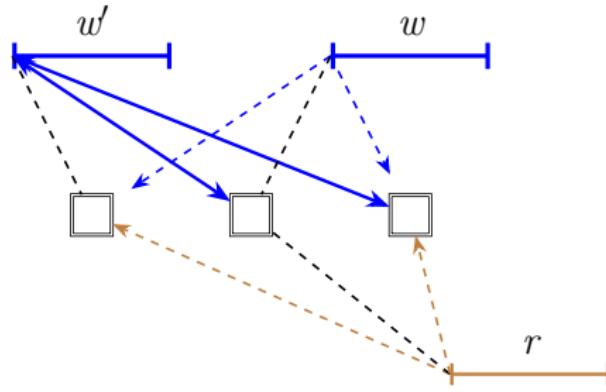
系统协议: n 副本节点; “过半数” ($M = \lfloor n/2 \rfloor + 1$) 通信规则

球盒模型: 操作产生 n 个球, 发送到 n 个副本盒子

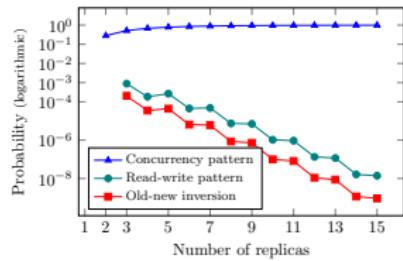
时刻 t : 操作 r 恰好前 M 个球到达相应盒子

$R(r) \neq w$: 操作 w 相应 M 个球尚在途中

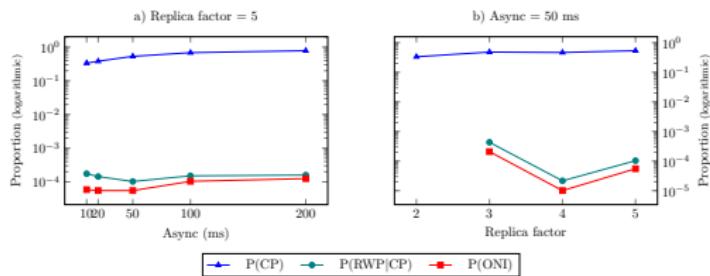
带时间: 操作起始时间 + 消息延迟



PA2AM 量化分析

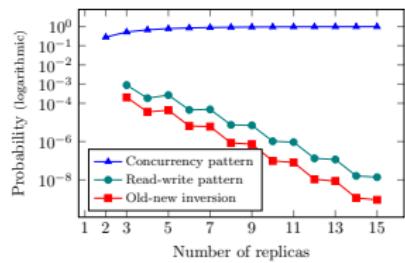


(a) 数值结果.

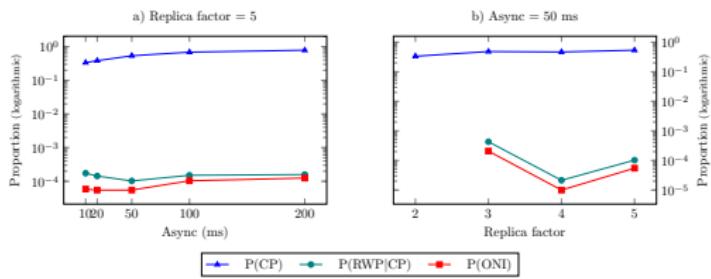


(b) 实验结果.

PA2AM 量化分析



(a) 数值结果.

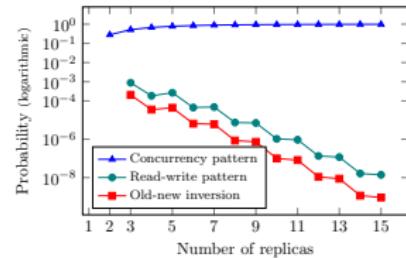


(b) 实验结果.

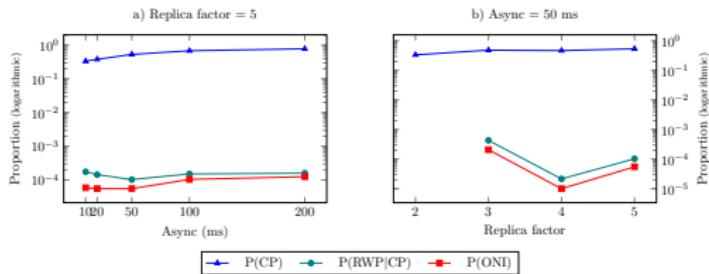
观察 (观察一)

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.

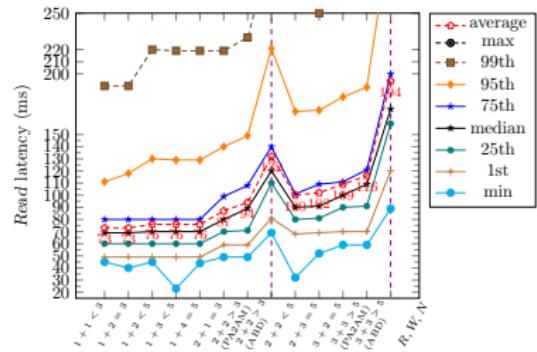
观察 (观察一)

从概率角度讲, PA2AM 算法极少违反 *atomicity* 一致性.

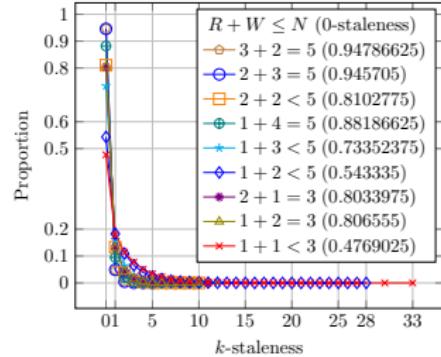
观察 (观察二)

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-Maj 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)						
	R, W, N	$1 + 1 < 3$	$1 + 2 = 3$	$2 + 1 = 3$	$2 + 2 > 3$ (PA2AM)	$1 + 2 < 5$	$1 + 3 < 5$	$1 + 4 = 5$	$2 + 2 < 5$	$2 + 3 = 5$	$3 + 2 = 5$
max k	6	4	2	1	2	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 (在大多数时候) 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 (在大多数时候) 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@PVLDB'12]
2. 度量违反 atomicity (vs. regularity [Lee@DC'05] [Bailis@PVLDB'12]) 的程度