

CAP Theorem: Revision of its Related Consistency Models

Francesc D. Muñoz-Escó¹, Rubén de Juan-Marín¹,
José-Ramón García-Escrivá¹, J. R. González de Mendivil²,
José M. Bernabéu-Aubán¹

¹Instituto Universitario Mixto Tecnológico de Informática
Universitat Politècnica de València
46022 Valencia (Spain)

²Depto. de Ingeniería Matemática e Informática
Universidad Pública de Navarra
31006 Pamplona (Spain)

{fmunoz,rjuan,rgarcia,josep}@iti.upv.es, mendivil@unavarra.es

Technical Report TR-IUMTI-SIDI-2017/002

CAP Theorem: Revision of its Related Consistency Models

Francesc D. Muñoz-Escó¹, Rubén de Juan-Marín¹,
José-Ramón García-Escrivá¹, J. R. González de Mendivil²,
José M. Bernabéu-Aubán¹

¹Instituto Universitario Mixto Tecnológico de Informática
Universitat Politècnica de València
46022 Valencia (Spain)

²Depto. de Ingeniería Matemática e Informática
Universidad Pública de Navarra
31006 Pamplona (Spain)

Technical Report TR-IUMTI-SIDI-2017/002

e-mail: {fmunyoz,rjuan,rgarcia,josep}@iti.upv.es, mendivil@unavarra.es

Revised edition, June 2018

Abstract

The CAP theorem states that only two of these properties can be simultaneously guaranteed in a distributed service: (i) consistency, (ii) availability, and (iii) network partition tolerance. This theorem was stated and proven assuming that “consistency” refers to atomic consistency. However, multiple consistency models exist and atomic consistency is located at the strongest edge of that spectrum.

Many distributed services deployed in cloud platforms should be highly available and scalable. Network partitions may arise in those deployments and should be tolerated. One way of dealing with CAP constraints consists in relaxing consistency. Therefore, it is interesting to explore the set of consistency models not supported in an available and partition-tolerant service (CAP-constrained models). Other weaker consistency models could be maintained when scalable services are deployed in partitionable systems (CAP-free models). Three contributions arise: (1) multiple other CAP-constrained models are identified, (2) a borderline between CAP-constrained and CAP-free models is set, and (3) a hierarchy of consistency models depending on their strength and convergence is built.

*NOTE: This technical report has been extended and improved in a subsequent edition accepted for its publication in *The Computer Journal* (Oxford University Press, United Kingdom). The DOI of that published article is: 10.1093/comjnl/bxy142. The authors recommend that readers download and use the published article version.*

1 Introduction

Scalable distributed services try to maintain their service continuity in all situations. When they are geo-replicated, a trade-off exists among three properties: replica consistency (C), service availability (A) and network partition tolerance (P). Only two of those three properties can be simultaneously guaranteed. Such trade-off was suggested long time ago (Davidson et al., 1985) [12], thoroughly explained by Fox and Brewer [16] in 1999 and proved by Gilbert and Lynch [17] in 2002. However, the compromise between strongly consistent actions, availability and tolerance to network partitions was already implicit in Johnson and Thomas (1975) [22] and justified by Birman and Friedman [7] in 1996.

Service availability and network partition tolerance are dichotomies. They are either respected or not. Service availability means that *every client request* that reaches a service instance should be answered. When a network partition arises, the instances of a service may be spread among multiple disjoint node subgroups. Network partition tolerance means that every service instance subgroup goes on while the network remains partitioned.

On the other hand, service replica consistency admits a gradation of consistency levels. In spite of this, when we simply refer to “consistency” we understand that it means atomic consistency [27]; i.e., that all instances are able to maintain the same values for each variable at the same time, providing a behaviour equivalent to that of a single copy. This led to assume that kind of consistency in the original proofs of the CAP theorem [17].

With the advent of cloud computing, it is easy to develop and deploy highly scalable distributed services [34]. Those applications usually provide world-wide services: they are deployed in multiple datacentres and this implies that network partition tolerance is a must for those services. Thus, those services regularly prioritise availability when they should deal with the constraints of the CAP theorem, and consistency is the property being sacrificed. However, that sacrifice should not be complete. Brewer [8] explains that network partitions are rare, even for world-wide geo-replicated services. If services demand partition tolerance and availability, their consistency may be still quite strong most of the time, relaxing it when any temporary network partition arises.

Therefore, it seems interesting to explore which levels of consistency are strong enough to be directly implied by the CAP constraints; i.e., those *CAP-constrained* models are not supported when the network becomes partitioned. On the other hand, there are several relaxed models that remain available when a network partition arises. They constitute the *CAP-free* set of models and there is a (not yet completely known) frontier between *CAP-free* and *CAP-constrained* models. Two questions arise in this scope: (1) Does CAP affect only to atomic consistency or are there any other “CAP-constrained” models? (2) If there were any other models, what would the CAP-constrained vs. CAP-free frontier be? Although some partial answers to these questions have been given in previous papers [32, 36, 4], let us provide a new and revised answer to them in the following sections.

2 System Model

A distributed system $S = (P, O)$ is assumed. The real-time domain is represented by set T . S is partially synchronous and consists of: (1) a set of processes P connected by a network where processes communicate through message passing, and (2) a set of objects O , with their states and methods. Processes in P may fail. Scalable distributed services may be deployed in S . Those services consist of a set of objects O . Objects are replicated in order to improve their availability. Their instances are deployed in P using a replication protocol and respecting some replica consistency model.

Function $Connect : P \times P \times T \rightarrow \{false, true\}$, used as $Connect(p_1, p_2, t)$, returns *true* when processes p_1 and p_2 are connected at time t , and *false* otherwise. Communication may fail when a temporary network partition occurs, defined as follows.

Definition 1 (Network partition). *When a network partition $NP = (S, K, it, et)$ occurs in a system $S = (P, O)$ from some initial time $it \in T$ to an end time $et \in T$ ($it < et$), S becomes partitioned in a set K of network components, with $|K| > 1$, such that:*

1. $\bigcup_{i \in K} S_i \subseteq S$, where $S_i = (P_i, O)$
2. $\bigcup_{i \in K} P_i \subseteq P$
3. $\forall i, j \in K, i \neq j : P_i \cap P_j = \emptyset$
4. $\forall i, j \in K, i \neq j, \forall p_m \in P_i, \forall p_n \in P_j, \forall t \in T, it \leq t \leq et : Connect(p_m, p_n, t) = false$
5. $\forall i \in K, \forall p_m, p_n \in P_i, \forall t \in T, it \leq t \leq et : Connect(p_m, p_n, t) = true$

Processes in different components cannot communicate with each other. Processes in the same component intercommunicate without problems. A *partitionable* system model is assumed in regard to process behaviour.

Proposition 1 (Partitionable system). *When a network partition $NP = (S, K, it, et)$ occurs in $S = (P, O)$, every operation from every process $p_i \in P$ is able to start and/or finish in a regular way in the (it, et) interval, independently on the connectivity of p_i with each other process $p_j \in P$.*

According to Prop. 1, no operation gets indefinitely blocked while a network partition lasts in S . Considering the CAP constraints, availability and network partition tolerance are respected, while consistency compliance may be sacrificed.

3 Basic Specification

Viotti and Vukolić propose a framework for specifying distributed (non-transactional) data consistency models in [43], based on that presented in [10, 9]. Since the CAP theorem involves software services deployed in distributed systems, it makes sense to consider those models in this scope. That framework may be summarised as follows.

3.1 Specification Framework

Services consist of processes and objects. Object values belong to set V . Processes interact with objects invoking their operations, whose types belong to set OT .

Tuples $(proc, type, obj, ival, oval, st, rt)$ represent operations, where:

- $proc \in P$ is the identifier of the process that invokes the operation.
- $type \in OT$ is the operation type; e.g., wr for writes and rd for reads.
- $obj \in O$ is the identifier of the invoked object.
- $ival \in V \cup \{\sqcup\}$ is the operation input value, or \sqcup in case of a read operation.
- $oval \in V \cup \{\sqcup, \nabla, \Theta\}$ is the operation output value, or \sqcup in case of a write or ∇ if the operation does not return or Θ when a write completes in $proc$ but not in other subsets of P .
- $st \in T$ is the operation invocation (i.e., start) time.
- $rt \in T$ is the operation return time.

In a tuple $T = (e_1, \dots, e_n)$, $T.e_i$ refers to element e_i in that tuple.

A history H is a set of operations. A history contains all operations invoked in an execution E of S . $H|_{wr}$ (respectively, $H|_{rd}$) denotes the set of write (respectively, read) operations in a history H . Formally, $H|_{wr} = \{op \in H : op.type = wr\}$.

The following relations are needed: (1) rb (*returns-before*) is a partial order on H based on real-time precedence: $rb \equiv \{(a, b) : a, b \in H \wedge a.rt < b.st\}$, (2) ss (*same-session*) is an equivalence relation on H that groups the operations invoked by the same process: $ss \equiv \{(a, b) : a, b \in H \wedge a.proc = b.proc\}$, (3) so (*session order*) is a partial order defined as: $so \equiv rb \cap ss$, (4) ob (*same-object*) is an equivalence relation on H that groups the operations invoked on the same object: $ob \equiv \{(a, b) : a, b \in H \wedge a.obj = b.obj\}$, and (5) $concur$ is a symmetric binary relation that includes all pairs of real-time concurrent operations invoked on the same object: $concur \equiv ob \setminus rb$.

Moreover, there are other specification aspects to be considered. To begin with, the $concur$ relation is complemented with a function $Concur : H \rightarrow 2^H$ that denotes the set of write operations concurrent with a given operation: $Concur(a) \equiv \{b \in H|_{wr} : (a, b) \in concur\}$. A binary relation rel over A is a subset $rel \subseteq A \times A$. The projection $rel|_{wr \rightarrow rd}$ identifies all pairs of operations in relation rel that consist of a write and a read operation. H / \approx_{rel} denotes the set of equivalence classes determined by relation rel , rel^{-1} denotes the inverse relation of rel and $rel(a) = \{b \in A : (a, b) \in rel\}$. Note that $rel(a)$ is a set, since there may be many elements related transitively to a .

An execution is defined as $E = (H, vis, ar)$ and is built on a history H , complemented with two relations vis and ar on elements of H , where: (1) vis (*visibility*) is an acyclic partial order that accounts for the

propagation of write operations; two write operations are *invisible* to each other when they are not ordered by *vis*, and (2) *ar* (*arbitration*) is a total order on operations of the history that specifies how conflicts due to invisible operations are resolved in E in order to respect its consistency models.

The *happens-before* (*hb*) partial order is defined as the transitive closure of the union of *so* and *vis*; i.e., $hb \equiv (so \cup vis)^+$.

Some extensions to [43] are needed in order to deal with partitionable networks. Those extensions are specified hereafter.

\mathcal{E} is the set of executions in S . \mathcal{E}_P is the subset of \mathcal{E} that contains all executions in which the conditions of Def. 1 are met, i.e., their network becomes temporarily partitioned. Besides, \mathcal{E}_C is the complementary subset of \mathcal{E}_P in which no network partition has occurred. Thus, $\mathcal{E} = \mathcal{E}_P \cup \mathcal{E}_C$ and $\mathcal{E}_P \cap \mathcal{E}_C = \emptyset$.

The context C of an operation op in execution E is defined as: $C_{op} = cxt(E, op) \equiv (E.vis^{-1}(op), E.vis|_{C_{op}.H}, E.ar|_{C_{op}.H})$, i.e., a projection of E that only keeps in its history those operations in $vis^{-1}(op)$. For each data type, function \mathcal{F} specifies the set of intended return values of op in relation to its context: $\mathcal{F}(op, cxt(E, op))$. With \mathcal{F} , the *return value consistency* is defined as: $RVAL(\mathcal{F}) \equiv \forall op \in E.H : op.oval \in \mathcal{F}(op, cxt(E, op))$. In this scope, we use by default a *register* data type (\mathcal{F}_{reg}). Let us explain how $op.oval$ is chosen from C_{op} in \mathcal{F}_{reg} . From $vis^{-1}(op)$, only those $op_2 \in C_{op}.H : op_2.oval \neq \nabla \wedge op_2.obj = op.obj$ are considered. Multiple candidates may arise. If so, only those operations without *vis*-successors in $C_{op}.H$ are assessed. From that subset, with operations invisible to each other, the read value is that of the latest operation in *ar* order. If no candidate exists, then $op.oval$ is a special value \perp .

Table 1: Definition of basic consistency predicates.

Predicate	Definition
RVAL(\mathcal{F})	$\forall op \in H : op.oval \in \mathcal{F}(op, cxt(E, op))$
PRAM	$so \subseteq vis$
SINGLEORDER	$\exists H' \subseteq \{op \in H : op.oval = \nabla\} : vis = ar \setminus (H' \times H)$
LAZYSINGLEORDER	$\exists H' \subseteq \{op \in H : op.oval \in \{\nabla, \Theta\}\} : vis = ar \setminus (H' \times H)$
REALTIME	$rb \subseteq ar$
REALTIMEWRITES	$rb _{wr \rightarrow op} \subseteq ar$
SEQRVAL(\mathcal{F})	$\forall op \in H : Concur(op) = \emptyset \Rightarrow op.oval \in \mathcal{F}(op, cxt(E, op))$
EVENTUALVISIBILITY	$\forall a \in H, \forall [f] \in H / \approx_{ss} : \{b \in [f] : (a, b) \in rb \wedge (a, b) \notin vis\} < \infty$
NOCIRCULARCAUSALITY	$acyclic(hb)$
STRONGCONVERGENCE	$\forall a, b \in H _{rd} : vis^{-1}(a) _{wr} = vis^{-1}(b) _{wr} \Rightarrow a.oval = b.oval$
CAUSALVISIBILITY	$hb \subseteq vis$
CAUSALARBITRATION	$hb \subseteq ar$
TIMEDVISIBILITY(Δ)	$\forall a \in H _{wr}, \forall b \in H, \forall t \in T : a.rt = t \wedge b.st = t + \Delta \Rightarrow (a, b) \in vis$
REALTIMEWW	$rb _{wr \rightarrow wr} \subseteq ar$
CONCURRVAL(\mathcal{F})	$\forall op \in H : op.oval \in \mathcal{F}(op, cxt(E, op) \cup Concur(op))$
K-REALTIMEREADS(K)	$\forall a \in H _{wr}, \forall b \in H _{rd}, \forall PW \subseteq H _{wr}, \forall pw \in PW : PW < K \wedge (a, pw) \in ar \wedge (pw, b) \in rb \wedge (a, b) \in rb \Rightarrow (a, b) \in ar$
NOJOIN	$\forall a_i, b_i, a_j, b_j \in H : a_i \not\approx_{ss} a_j \wedge (a_i, a_j) \in ar \setminus vis \wedge a_i \preceq_{so} b_i \wedge a_j \preceq_{so} b_j \Rightarrow (b_i, b_j), (b_j, b_i) \notin vis$
ATMOSTONEJOIN	$\forall a_i, a_j \in H : a_i \not\approx_{ss} a_j \wedge (a_i, a_j) \in ar \setminus vis \Rightarrow \{b_i \in H : a_i \preceq_{so} b_i \wedge (\exists b_j \in H : a_j \preceq_{so} b_j \wedge (b_i, b_j) \in vis)\} \leq 1 \wedge \{b_j \in H : a_j \preceq_{so} b_j \wedge (\exists b_i \in H : a_i \preceq_{so} b_i \wedge (b_j, b_i) \in vis)\} \leq 1$
PEROBJECTPRAM	$(so \cap ob) \subseteq vis$
PEROBJECTSINGLEORDER	$\exists H' \subseteq \{op \in H : op.oval = \nabla\} : vis \cap ob = ar \cap ob \setminus (H' \times H)$

Let us use an execution E_x for explaining the specification aspects presented in previous paragraphs. Let S be $(\{p_1, p_2\}, \{x\})$ and $E_x = (\{o_1 = (p_1, wr, x, 1, \perp, 0, 1), o_2 = (p_2, wr, x, 2, \perp, 0, 1), o_3 = (p_1, rd, x, \perp, 1, 1, 2), o_4 = (p_2, rd, x, \perp, 2, 1, 2), o_5 = (p_1, rd, x, \perp, 2, 3, 4), o_6 = (p_2, rd, x, \perp, 1, 3, 4)\}, \{(o_1, o_3), (o_3, o_5), (o_2, o_4), (o_4, o_6), (o_2, o_5), (o_1, o_6)\}, \{(o_4, o_1), (o_1, o_6), (o_6, o_3), (o_3, o_2), (o_2, o_5)\})$. Local execution order introduces $(o_1, o_3), (o_3, o_5), (o_2, o_4)$ and (o_4, o_6) in *vis*. Values written in o_1 and o_2 are propagated to the other process, so (o_2, o_5) and (o_1, o_6) are in *vis*. Since *ar* is a total order, it sets this ordering in E_x : $o_4 < o_1 < o_6 < o_3 < o_2 < o_5$. There are four reads: o_3, o_4, o_5 and o_6 , with these context histories: $C_{o_3}.H = \{o_1\}$, $C_{o_4}.H = \{o_2\}$, $C_{o_5}.H = \{\underline{o_1}, o_3, o_2\}$, $C_{o_6}.H = \{\underline{o_2}, o_4, o_1\}$. In each $C_i.H$, the underlined

operations are discarded when $\text{RVAL}(\mathcal{F})$ is applied, since they have subsequent operations in vis that are also in $C_i.H$. From the remaining subsets, any potential conflict is resolved according to ar . This explains the read values.

3.2 Distributed Consistency Models

Viotti and Vukolić [43] distinguish ten groups of consistency models: (1) linearisable and other strong models, (2) weak and eventual consistency, (3) PRAM and sequential consistency, (4) session guarantees, (5) causal models, (6) staleness-based models, (7) fork-based models, (8) composite and tunable models, (9) per-object models, and (10) synchronised models. Synchronised models are described in [43] for completeness; they make sense in multiprocessor computers but not in general distributed systems. The models in the eighth group cannot be specified with the proposed consistency predicates. Therefore, no relation with the models contained in other groups can be set for them. Those two groups are not considered hereafter. Table 1 shows a set of consistency predicates. With those predicates, consistency models may be specified as shown in Table 2.

Consistency models are also known as *consistency conditions*. Both terms are synonyms, but generate two different kinds of names. Conditions use nouns (e.g., linearisability [20]) while models use adjectives (e.g., atomic, regular and safe [27]). In order to be uniform, this paper uses models and adjectives in order to refer to consistency in all cases.

An execution E satisfies a consistency model M built as a conjunction of multiple consistency predicates ($M \equiv \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n$) iff E satisfies all those predicates. Formally: $E \models M \Leftrightarrow E \models \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n$.

In regard to the consistency models specified in Table 2, $\text{PREFIXSEQUENTIAL}(\mathcal{F})$ is derived from the “prefix consistency” proposed in Bayou [40]. Bayou manages a partitionable system. To this end, write operations have two states: tentative (Θ) and committed (\sqcup). That management is specified using LAZYSINGLEORDER in Table 2. In Bayou, a write operation op returns control once it reaches a single server p_i . At that time, op is still tentative (i.e., $op.oval = \Theta$). To be committed, p_i propagates op to a primary manager. The primary manager for $op.obj$ chooses a commit order (that conditions the ar relation in that execution) for all new writes on that object and that chosen sequence is kept in a log and lazily communicated to every other process. Disconnected nodes should eventually contact the primary manager to learn that commit order. At that time, those previously disconnected processes communicate their tentative writes to the primary (to be ordered on the next commit) and apply the already committed writes on their local replicas. This means that tentative writes may be undone and reapplied in their correct sequence position when they had been initially applied in a disconnected node. When a write op is applied onto the replica of object $op.obj$ in a process p_j in the commit order, $op.oval$ becomes \sqcup in $H|_{p_j}$.

3.3 CAP-related Definitions

Let assume that the executions in system S are driven by a consistency model $CM \equiv \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n$. All executions in \mathcal{E}_C comply always with the definition of CM . However, that behaviour may vary when network partitions arise. That fact originates the following definitions.

Definition 2 (CAP-free consistency model). *CM is CAP-free if every execution E in \mathcal{E}_P respects all consistency predicates that define CM.*

Formally: $\forall E \in \mathcal{E}_P : E \models \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n$.

This means that a CAP-free model is respected when the network is partitioned. On the other hand, a CAP-constrained model is not respected by every execution in case of network partitions:

Definition 3 (CAP-constrained consistency model). *CM is CAP-constrained if there is an execution E in \mathcal{E}_P that does not fulfil all consistency predicates that define CM.*

Formally: $\exists E \in \mathcal{E}_P : E \not\models \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n$.

Def. 3 cannot consider every execution in \mathcal{E}_P since there may be executions in a partitioned system that still comply with all CM predicates. For instance, let consider execution $E_1 \in \mathcal{E}_P$ with $S = (\{p_1, p_2\}, \{x, y\})$ and a partition with $P_1 = \{p_1\}$, $P_2 = \{p_2\}$, $it = 4$ and $et = 10$: $E_1 = (\{o_1 = (p_1, wr, x, 2, \sqcup, 0, 1), o_2 =$

Table 2: Definition of basic consistency models.

Model	Ref.	Definition
<i>1.- Linearisable and other strong models</i>		
LINEARISABLE(\mathcal{F})	[20]	SINGLEORDER \wedge REALTIME \wedge RVAL(\mathcal{F})
REGULAR(\mathcal{F})	[27]	SINGLEORDER \wedge REALTIMEWRITES \wedge RVAL(\mathcal{F})
SAFE(\mathcal{F})	[27]	SINGLEORDER \wedge REALTIMEWRITES \wedge SEQRVAL(\mathcal{F})
<i>2.- Weak and eventual consistency</i>		
WEAK		No requirement
EVENTUAL(\mathcal{F})	[13]	EVENTUALVISIBILITY \wedge NOCIRCULARCAUSALITY \wedge RVAL(\mathcal{F})
STRONGEVENTUAL(\mathcal{F})	[38]	EVENTUAL(\mathcal{F}) \wedge STRONGCONVERGENCE
<i>3.- PRAM and sequential consistency</i>		
PRAM(\mathcal{F})	[30]	PRAM \wedge RVAL(\mathcal{F})
SEQUENTIAL(\mathcal{F})	[26]	SINGLEORDER \wedge PRAM(\mathcal{F})
<i>4.- Session guarantees</i>		
MONOTONICREADS	[39]	$\forall a \in H, \forall b, c \in H \mid_{rd}: (a, b) \in vis \wedge (b, c) \in so \Rightarrow (a, c) \in vis$
READYOURWRITES	[39]	$\forall a \in H \mid_{wr}, \forall b \in H \mid_{rd}: (a, b) \in so \Rightarrow (a, b) \in vis$
MONOTONICWRITES	[39]	$\forall a, b \in H \mid_{wr}: (a, b) \in so \Rightarrow (a, b) \in ar$
WRITESFOLLOWREADS	[39]	$\forall a, c \in H \mid_{wr}, \forall b \in H \mid_{rd}: (a, b) \in vis \wedge (b, c) \in so \Rightarrow (a, c) \in ar$
<i>5.- Causal models</i>		
CAUSAL(\mathcal{F})	[1]	CAUSALVISIBILITY \wedge CAUSALARBITRATION \wedge RVAL(\mathcal{F})
CAUSAL+(\mathcal{F})	[31]	STRONGCONVERGENCE \wedge CAUSAL(\mathcal{F})
REALTIMECAUSAL(\mathcal{F})	[32]	REALTIME \wedge CAUSAL(\mathcal{F})
<i>6.- Staleness-based models</i>		
PREFIXSEQUENTIAL(\mathcal{F})	[40]	LAZYSINGLEORDER \wedge MONOTONICWRITES \wedge RVAL(\mathcal{F})
TIMEDCAUSAL(\mathcal{F}, Δ)	[42]	CAUSAL(\mathcal{F}) \wedge TIMEDVISIBILITY(Δ)
TIMEDSERIAL(\mathcal{F}, Δ)	[41]	SINGLEORDER \wedge TIMEDVISIBILITY(Δ) \wedge RVAL(\mathcal{F})
K-LINEARISABLE(\mathcal{F}, K)	[2]	SINGLEORDER \wedge K-REALTIMEREADS(K) \wedge REALTIMEWW \wedge RVAL(\mathcal{F})
K-REGULAR(\mathcal{F}, K)	[2]	SINGLEORDER \wedge K-REALTIMEREADS(K) \wedge REALTIMEWW \wedge CONCURRVAL(\mathcal{F})
K-SAFE(\mathcal{F}, K)	[2]	SINGLEORDER \wedge K-REALTIMEREADS(K) \wedge REALTIMEWW \wedge SEQRVAL(\mathcal{F})
<i>7.- Fork-based models</i>		
FORKLINEARISABLE(\mathcal{F})	[33]	REALTIME \wedge NOJOIN \wedge PRAM(\mathcal{F})
FORKSEQUENTIAL(\mathcal{F})	[35]	NOJOIN \wedge PRAM(\mathcal{F})
FORK*(\mathcal{F})	[29]	READYOURWRITES \wedge REALTIME \wedge RVAL(\mathcal{F}) \wedge ATMOSTONEJOIN
<i>8.- Per-object models</i>		
SLOW(\mathcal{F})	[21]	PEROBJECTPRAM \wedge RVAL(\mathcal{F})
CACHE(\mathcal{F})	[19]	PEROBJECTSINGLEORDER \wedge SLOW(\mathcal{F})
PROCESSOR(\mathcal{F})	[19]	PEROBJECTSINGLEORDER \wedge PRAM(\mathcal{F})

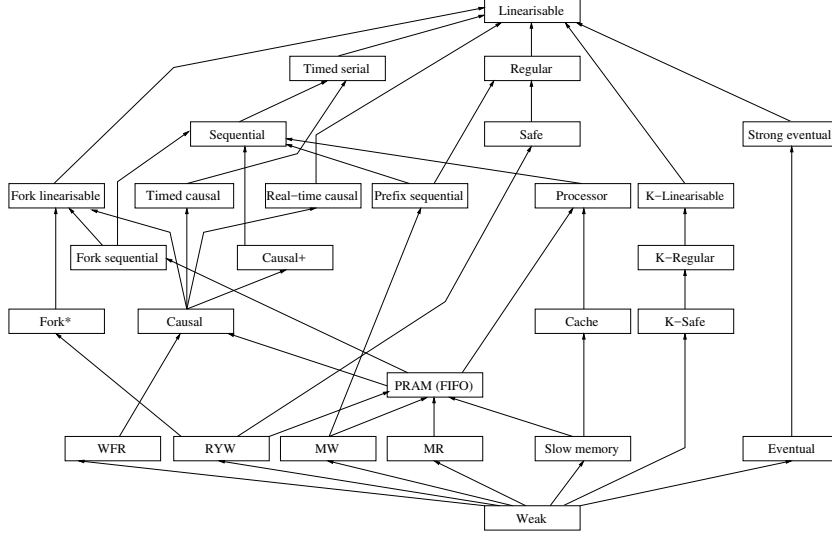


Figure 1: Strength-based partial ordering of consistency models.

$(p_2, rd, x, \perp, 2, 2, 3)$, $o_3 = (p_2, wr, x, 3, \perp, 3, 4)$, $o_4 = (p_1, wr, y, \perp, 2, 4, 5)$, $o_5 = (p_2, wr, x, 6, \perp, 5, 6)$, $o_6 = (p_1, rd, y, \perp, 2, 7, 8)$, $o_7 = (p_1, rd, x, \perp, 6, 11, 12)$, $o_8 = (p_2, rd, y, \perp, 2, 13, 14)\}$, $\{(o_1, o_2), (o_2, o_3), (o_3, o_4), (o_4, o_5), (o_5, o_6), (o_6, o_7), (o_7, o_8)\}$, $E_1.vis$).

Processes p_1 and p_2 have remained active in the partition interval (4, 10). Availability is maintained. E_1 complies with all predicates that define the $\text{LINEARISABLE}(\mathcal{F})$ model. As it will be stated in Theorem 1, $\text{LINEARISABLE}(\mathcal{F})$ is CAP-constrained. This shows that there are executions of CAP-constrained models in partitioned systems that still respect the predicates that define their consistency model.

In order to compare consistency models in S , let use \mathcal{E}^{CM} as the set of executions admitted in S by a consistency model CM . This allows the following definition:

Definition 4 (Weaker than (\rightarrow) relation on models). A model $A \equiv \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$ is weaker than another model $B \equiv \mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$, i.e., $A \rightarrow B$, when the set of executions admitted by B is a proper subset of the set of executions admitted by A .

Formally: $A \rightarrow B \equiv \mathcal{E}^B \subset \mathcal{E}^A$.

Alternatively: $A \rightarrow B \equiv \mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m} \Rightarrow \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$.

When $A \rightarrow B$, model B is stronger than A . Besides, when $A \not\rightarrow B \wedge B \not\rightarrow A$ then A and B are incomparable. Figure 1 depicts those relations. This initial model hierarchy is adapted from [43]. From Def. 4, combined with Def. 2 and 3, the following propositions can be stated.

Proposition 2 (Freedom of weaker CAP-free models). Given two consistency models $A \equiv \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$ and $B \equiv \mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$, if $A \rightarrow B$ and B is CAP-free, then A is also CAP-free.

Proof. By Def. 4: (1) $\mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m} \Rightarrow \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$. Additionally, since B is CAP-free, according to Def. 2: (2) $\forall E \in \mathcal{E}_P^B : E \models \mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$.

Let assume that A was CAP-constrained. In that case, according to Def. 3: $\exists E_1 \in \mathcal{E}_P : E_1 \not\models \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$. This would imply that $\mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$ is false in some executions when network partitions arise. If so, due to (1), $\mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$ is neither generally true in case of partitions. So, an execution E_2 could be found such that $E_2 \in \mathcal{E}_P : E_2 \not\models \mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$ and this implies that B is CAP-constrained. This sets a contradiction with (2). Thus, A is CAP-free. \square

Proposition 3 (Constriction of stronger CAP-constrained models). Given two consistency models $A \equiv \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$ and $B \equiv \mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$, if $A \rightarrow B$ and A is CAP-constrained, then B is also CAP-constrained.

Proof. By Def. 4: (1) $\mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m} \Rightarrow \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$. Additionally, since A is CAP-constrained, according to Def. 3: (2) $\exists E_1 \in \mathcal{E}_P : E_1 \not\models \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$.

Let assume that B was CAP-free. In that case, according to Def. 2: $\forall E \in \mathcal{E}_P^B : E \models \mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$. This would imply that $\mathcal{P}_{B,1} \wedge \dots \wedge \mathcal{P}_{B,m}$ is true when network partitions arise. If so, due to (1), $\mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$ is also true in case of partitions. Thus, $\forall E \in \mathcal{E}_P^A : E \models \mathcal{P}_{A,1} \wedge \dots \wedge \mathcal{P}_{A,n}$ and therefore A is CAP-free. This sets a contradiction with (2). So, B is CAP-constrained. \square

4 Finding a Consistency Border

Let us take the specifications of all consistency models as a base for analysing which requires any agreement that a network partition will break. Those models are *CAP-constrained*. Section 4.1 presents that analysis. Section 4.2 goes on in this analysis looking for other conditions that are not related with consensus and cannot be either attained in an available and partitioned system. Finally, Section 4.3 looks for convergence-based inter-model relations.

Most predicates in Table 1 strengthen the definition of the consistency models that include them. However, there are several exceptions: K-REALTImEREADS(K), NOJOIN, ATMOSTONEJOIN and SEQRVAL(\mathcal{F}). The first one breaks read determinism. Instead of returning the last received write, each process may read any of the K latest writes. The second and third ones impose divergence among processes once a failure happens. The last one is a weaker variant for RVAL(\mathcal{F}), since it drops the determinism of read operations in case of concurrent writes. Because of this, all consistency models that include them should be analysed with care.

4.1 Starting Point: The Linearisable Model

Gilbert and Lynch [17] proved the CAP theorem assuming *linearisable* [20] consistency. Let us start revising its definition, given in Table 2, with the goal of identifying which predicates cannot be respected in a partitioned and available system:

$$\text{LINEARISABLE}(\mathcal{F}) \equiv \text{SINGLEORDER} \wedge \text{REALTIME} \wedge \text{RVAL}(\mathcal{F})$$

RVAL(\mathcal{F}) defines the appropriateness of the return value in an operation based on its execution context. Its compliance is assumed in the following discussions. The other predicates mean the following (Table 1):

- **REALTIME:** $rb \subseteq ar$. This predicate states that all operations ordered by the returns-before (rb) relation are considered in the arbitration relation (ar). Since rb considers real time, it is able to order the operations executed by different processes, even when they are placed in different network components in a partition.

Relation ar totally orders the operations in a history H . Considered in isolation, REALTIME may be respected in a partitioned system. However, it leads to contradictions when the vis relation assumed in other predicates requires communication among isolated network components.

As a result, the inclusion of REALTIME in a consistency model may endanger the compliance with other vis -related predicates, e.g., SINGLEORDER.

- **SINGLEORDER:** $\exists H' \subseteq \{op \in H : op.oval = \nabla\} : vis = ar \setminus (H' \times H)$. This predicate states that the visibility relation (vis) coincides with the arbitration relation (ar), discarding incomplete operations. Let us prove that SINGLEORDER cannot be respected while a temporary network partition lasts in a system.

Lemma 1 (SINGLEORDER \wedge RVAL(\mathcal{F}) unattainability). *In a system $S = (P, O)$ that uses a consistency model that includes SINGLEORDER, while a network partition NP arises, there is some execution $E \in \mathcal{E}_P$ in which SINGLEORDER \wedge RVAL(\mathcal{F}) is false.*

Proof. Without loss of generality, let us assume the following two conditions:

1. $\exists E \in \mathcal{E}_P : E \models \text{SINGLEORDER} \wedge \text{RVAL}(\mathcal{F})$.

2. Partition NP has happened in the interval (it, et) generating two different separate network components P_1 and P_2 ; i.e., $NP = (S, \{1, 2\}, it, et)$. $\exists p_1, p_2 \in P : p_1 \in P_1, p_2 \in P_2$.

Process p_1 has executed $op_{1,1} = (p_1, wr, x, v, \sqcup, st, st + 1)$. Besides, p_2 has also run $op_{2,1} = (p_2, wr, x, v', \sqcup, st', st' + 1)$ with both $st, st' \in (it, et - 1)$. Process p_1 also runs $op_{1,2} = (p_1, rd, x, \sqcup, v, st'', st'' + 1)$ with $st'' \in (it, et - 1)$ and $st < st' < st''$.

Since $op_{1,1}$ and $op_{1,2}$ have happened both in p_1 , they are trivially related by vis : $(op_{1,1}, op_{1,2}) \in vis$. This justifies the read value in $op_{1,2}$ according to $RVAL(\mathcal{F})$.

By definition, ar should order all the operations in E . Since $RVAL(\mathcal{F})$ is respected in E , the order set by ar should allow that $op_{1,2}$ reads value v . This may only happen when $E.ar$ is either $ar_1 = \{(op_{2,1}, op_{1,1}), (op_{1,1}, op_{1,2})\}$ or $ar_2 = \{(op_{1,1}, op_{1,2}), (op_{1,2}, op_{2,1})\}$. Let us assume that $E.ar = ar_1$. Condition (1) above states that $SINGLEORDER$ is true in E . Thus, $E.vis = E.ar$.

Condition (2), above, states that S remained partitioned while E was run. Because of condition 4 from Def. 1, p_1 and p_2 cannot exchange messages in (it, et) . This means that $(op_{1,*}, op_{2,1}) \notin vis \wedge (op_{2,1}, op_{1,*}) \notin vis$; i.e., p_1 and p_2 operations cannot be directly or transitively related in vis . So, $E.vis \neq E.ar$. Therefore, this sets a contradiction with our assumed condition (1); i.e., execution E cannot respect $SINGLEORDER \wedge RVAL(\mathcal{F})$. \square

Lemma 1 assumes a single object x in its proof. So, this result applies to other predicates that constrain the $vis = ar$ equality to each object considered in isolation, e.g., to the $PEROBJECTSINGLEORDER$ predicate.

These predicates have been used in the definition of several consistency models. Models based on $SINGLEORDER$ or $PEROBJECTSINGLEORDER$ will be CAP -constrained when no relaxing predicate is used in their definition. On the other hand, those using $SINGLEORDER$ and any relaxing predicate, and those based on $REALTIME$ should be further assessed. Let us revise which they are:

- $SINGLEORDER$ with no relaxing predicate: *linearisable, regular, sequential, processor, cache and timed serial*.

Let us refer to these models as *strong-SINGLEORDER* models. SSO is the set of those models. All they are CAP -constrained, as stated in the following theorem.

Theorem 1 (Strong-SINGLEORDER models are CAP -constrained). *The $LINERISABLE(\mathcal{F})$, $REGULAR(\mathcal{F})$, $SEQUENTIAL(\mathcal{F})$, $PROCESSOR(\mathcal{F})$, $CACHE(\mathcal{F})$ and $TIMEDSERIAL(\mathcal{F}, \Delta)$ models are CAP -constrained.*

Proof. All models in SSO include the $SINGLEORDER \wedge RVAL(\mathcal{F})$ conjunction. They do not include any other predicate that relaxes what is required in that conjunction. Formally: $\forall M_j \in SSO, M_j \equiv \mathcal{P}_{j,1} \wedge \dots \wedge \mathcal{P}_{j,n_j}$. $\exists i, 1 \leq i \leq n_j : \mathcal{P}_{j,i} = SINGLEORDER \wedge RVAL(\mathcal{F})$.

Let us assume a system S with model M_j . Then, predicate $\mathcal{P}_{j,i}$ allows the application of Lemma 1. As a result: $\exists E \in \mathcal{E}_P, E \not\models \mathcal{P}_{j,1} \wedge \dots \wedge \mathcal{P}_{j,n_j}$, since $\mathcal{P}_{j,i}$ may be false in those executions. Thus, by Def. 3, each M_j is CAP -constrained. \square

- $SINGLEORDER$ with relaxing predicates: *safe, k-linearisable, k-regular and k-safe*.

Let us start our analysis with the *safe* model. It relaxes $SINGLEORDER$ with the conjunction of the $SEQRVAL(\mathcal{F})$ predicate. The latter only requires determinism in a read operation op_r when there are no other concurrent write operations with op_r . When there are concurrent write operations, op_r may return any value. Considering that behaviour, let us prove that the $SAFE(\mathcal{F})$ model is CAP -constrained.

Theorem 2 ($SAFE(\mathcal{F})$ is CAP -constrained). *The $SAFE(\mathcal{F})$ consistency model is CAP -constrained.*

Proof. Let us assume an execution $E_s \in \mathcal{E}_P$ with model $\text{SAFE}(\mathcal{F})$. Let us consider that there are two network components P_1 and P_2 in E_s , each one consisting of a single process, p_0 and p_1 , respectively. Let us imagine that in E_s there are no read events in process p_i concurrent with write events in process p_{1-i} . This may easily happen if the set of objects being managed by processes p_0 and p_1 is large, since each time one process is writing on a given object o_k the other process may be reading other objects. In this scenario, $\text{SEQRVAL}(\mathcal{F})$ is equivalent to $\text{RVAL}(\mathcal{F})$. As a result, $E_s \not\models \text{SAFE}(\mathcal{F})$, since Lemma 1 may then be applied onto E_s . Because of this, considering Def. 3, $\text{SAFE}(\mathcal{F})$ is CAP-constrained. □

The remaining models (*k-linearisable*, *k-regular* and *k-safe*) share $\text{K-REALTIMEREADS}(K)$ as their relaxing predicate. Let us refer to them as *k-staleness* models. Like $\text{SEQRVAL}(\mathcal{F})$, $\text{K-REALTIMEREADS}(K)$ relaxes read determinism. In this case, each read operation may return any of the K latest written values in S . Let us consider this predicate in the following lemma.

Lemma 2 ($\text{K-REALTIMEREADS}(K) \wedge \text{SINGLEORDER}$ unattainability). *In a system $S = (P, O)$ that uses a consistency model with $\text{K-REALTIMEREADS}(K) \wedge \text{SINGLEORDER}$, while a network partition NP arises in a time interval (it, et) , there is some execution $E \in \mathcal{E}_P$ in which $\text{K-REALTIMEREADS}(K) \wedge \text{SINGLEORDER}$ is false.*

Proof. Let us assume that there are at least two network components in S , being P_1 and P_2 two of those components, with at least one process in each component, p_1 and p_2 , respectively. E_{rt} is an execution in \mathcal{E}_P . In E_{rt} , process p_1 has written K different values on object x after time it , in operations $op_{1,1} \dots op_{1,K}$, respectively. Later, p_2 reads x in operation $op_{2,1}$. Process p_2 had not written any value on x since time it . All those operations have happened before time et .

According to $\text{K-REALTIMEREADS}(K)$, these K latest writes from p_1 precede $op_{2,1}$ in the *ar* relation. Thus, conceptually, p_2 reads any of those values in $op_{2,1}$. However, SINGLEORDER requires that all completed operations in *ar* are also in *vis*. This means that all effects from $op_{1,1} \dots op_{1,K}$ must be already delivered to p_2 when $op_{2,1}$ is started. Since p_1 and p_2 remain disconnected in the (it, et) interval, according to condition 4 from Def. 1, the values written by p_1 are not visible to $op_{2,1}$. So, $vis \neq ar$ in E_{rt} , i.e., SINGLEORDER is not true in E_{rt} .

Therefore, there are executions that do not comply with $\text{K-REALTIMEREADS}(K) \wedge \text{SINGLEORDER}$ in S . □

With that result, the following theorem may be proven.

Theorem 3 (*k-staleness* models are CAP-constrained). $\text{K-REGULAR}(\mathcal{F}, K)$, $\text{K-SAFE}(\mathcal{F}, K)$ and $\text{K-LINEARISABLE}(\mathcal{F}, K)$ are CAP-constrained.

Proof. Immediate from Lemma 2 and Def. 3. □

- **REALTIME** (or its variants **REALTIMEWRITES**, **REALTIMEWW**): *linearisable*, *regular*, *safe*, *k-linearisable*, *fork linearisable*, and *fork**. From these models, *fork linearisable* and *fork** have not been considered yet in the analysis, since the others include SINGLEORDER in their predicates. Besides, *fork linearisable* and *fork** contain in their definitions, respectively, the **NOJOIN** and **ATMOSTONEJOIN** relaxing predicates that have not been assessed yet.

It is worth noting that fork-based models were proposed by Mazières and Shasha in [33] with the aim of dealing with Byzantine failures in the management of networked file systems. When a failure of that kind arises, some users may create a new version of a given file without considering the latest version kept in the servers. This generates a new branch (i.e., a fork) of versions thereafter. If multiple failures exist, the resulting set of versions defines a tree of divergent branches and those branches cannot join again. The specification shown in Tables 1 and 2 is general and does not consider specific failure models. Our discussion considers those generalised consistencies. In them,

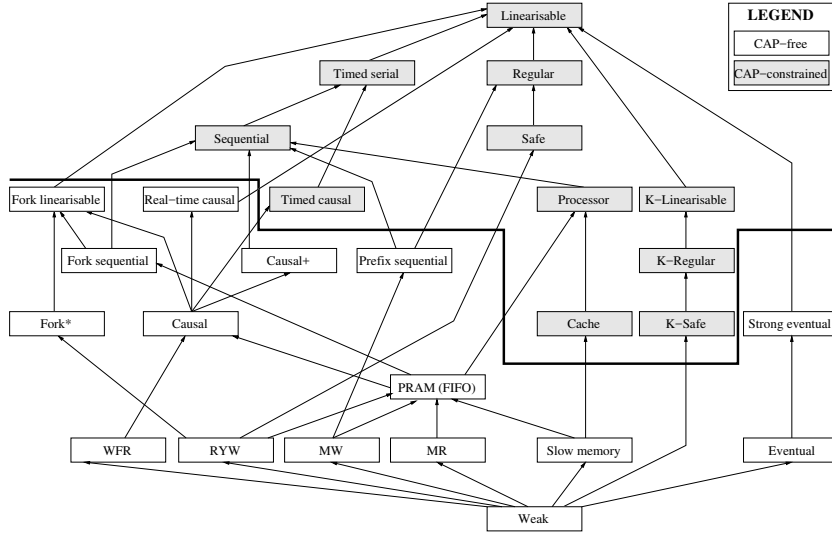


Figure 3: CAP-constrained vs CAP-free borderline.

4.2 Exploring the Frontier

Considering Figure 2, let us continue our analysis revising which are the predicates that define the set CCC (*CAP-constrained candidates*) of strongest models that do not need the `SINGLEORDER` or `REALTIME` conditions in their specifications. In this scope, “strongest” refers to those models that: (1) are not tagged yet as CAP-constrained, (2) are directly related as “weaker than” a CAP-constrained model, and (3) do not have any other CAP-free model stronger than them. According to Figure 2, these CCC models are: *strong eventual*, *causal+*, *timed causal*, *real-time causal* and *prefix sequential*. The *fork linearisable* model also complies with those three conditions, but Theorem 4 has already proven that it is CAP-free.

If any of these models was also CAP-constrained, we would continue our analysis with the new candidates generated by its inclusion in the CAP-constrained set. On the other hand, if no CCC model is identified as CAP-constrained in this stage, other models more relaxed than those in CCC will be also CAP-free, according to Prop. 2. In that case, our analysis should end there.

Part of these CCC models have been proven CAP-free either in the papers that proposed them or in other recent works. That is the case of `STRONGEVENTUAL(\mathcal{F})` [38], `CAUSAL+(\mathcal{F})` [31], `REALTIME-CAUSAL(\mathcal{F})` [32] and `PREFIXSEQUENTIAL(\mathcal{F})` [18]. Therefore, they are CAP-free by definition.

Let us consider now the other model: *timed causal*.

Theorem 5. *The `TIMEDCAUSAL(\mathcal{F}, Δ)` consistency model is CAP-constrained.*

Proof. Let us assume an execution $E \in \mathcal{E}_C : E \models \text{TIMEDCAUSAL}(\mathcal{F}, \Delta)$. Let a partition occur in S with $et - it > \Delta$. Let $E' \in \mathcal{E}_P : E'.H = E.H$. There may be (a, b) pairs in $E'.vis$ generated at network re-connection time with $a.rt = it$ (i.e., $t = it$) and $b.st > et$. Since $et - it > \Delta$, then $b.st > t + \Delta$. Thus, E' makes `TIMEDVISIBILITY(Δ)` false and this means that $E' \not\models \text{TIMEDCAUSAL}(\mathcal{F}, \Delta)$. Therefore, by Def. 3, `TIMEDCAUSAL(\mathcal{F}, Δ)` is CAP-constrained. \square

This analysis has shown that the *timed causal* model is CAP-constrained. Figure 2 shows that *timed causal* is directly stronger than *causal* and there is no other weaker model directly related with *timed causal*. The *causal* model is weaker than *causal+* and the latter is CAP-free. Thus, according to Prop. 2, `CAUSAL(\mathcal{F})` is CAP-free. This means that no other CAP-constrained model may be found. Therefore, the borderline between CAP-constrained and CAP-free models is finally set as depicted in Figure 3.

4.3 Revising Inter-Model Relationships

Previous figures show some “weaker than” relations between models. However, there are some other inter-model relations that have not been depicted yet. Those relations are explained hereafter. Section 4.3.1

discusses configurable models. Later, Section 4.3.2 describes relations that deal with state convergence.

4.3.1 Configurable Models

The family of *probabilistic bounded staleness* (PBS) [6] models consists of three models: PBS k -staleness, PBS t -visibility and PBS (k,t) -staleness. They are intended for quorum-based eventually consistent datastores. The first describes a probabilistic model that restricts the staleness of read values. The second limits probabilistically the time needed by written values to become visible. The third one combines the other two. PBS k -staleness, depending on its parameter k , may be weaker than or equivalent to k -linearisable, while PBS t -visibility is a probabilistic weakening of $\text{TIMEDVISIBILITY}(\Delta)$. Their place around the CAP-constrained vs CAP-free frontier depends on the values of their parameters, and this avoids their inclusion in Figure 4. They are generally configured as eventually consistent models, and this implies CAP-free models. However, PBS k -staleness when configured as non-stale becomes k -linearisable and, as such, is CAP-constrained.

As Brewer states in [8] “because partitions are rare, there is little reason to forfeit C (consistency) or A (availability) when the system is not partitioned”. This means that a strong model is needed while the network shows no connectivity problem and such consistency should be only relaxed when a network partition arises. Dynamically configurable models that may relax their consistency, as those proposed in the PBS family, seem to be an adequate solution to this problem. There have been several other models (e.g., [5, 25, 44, 24, 37, 28, 14]) of this kind, that are analysed by Viotti and Vukolić [43] in their eighth group: *composable and tunable* models. We refer the reader to [43] for a short description and comparison of them. Many of those models admit configurations in both parts of our identified frontier.

4.3.2 Convergence-based Relations

Eventual consistency was defined with the goal of breaking the constraints of the CAP theorem [4]. Eventual models that reach state convergence once their processes have seen the same set of write operations, like $\text{STRONGEVENTUAL}(\mathcal{F})$ [38], may be weaker than most CAP-constrained models. Indeed, according to Prop. 2, $\text{STRONGEVENTUAL}(\mathcal{F})$ cannot be stronger than any CAP-constrained model, since it is CAP-free [38]. Let us proceed in this analysis comparing several families of protocols:

- $\text{CACHE}(\mathcal{F})$ and stronger models.

If $\text{STRONGEVENTUAL}(\mathcal{F})$ was weaker than $\text{CACHE}(\mathcal{F})$, then $\text{CACHE}(\mathcal{F})$ predicates would imply those that define $\text{STRONGEVENTUAL}(\mathcal{F})$. Let us see whether that is true. Several lemmas are needed to this end.

Lemma 3. $\text{PEROBJECTSINGLEORDER} \wedge \text{RVAL}(\mathcal{F}) \Rightarrow \text{EVENTUALVISIBILITY} \wedge \text{RVAL}(\mathcal{F})$.

Proof. Without loss of generality, let assume a system $S = (\{p_1, p_2\}, \{x\})$ where all its executions should respect $\text{PEROBJECTSINGLEORDER}$. Let us consider executions in which all operations have returned. $\text{PEROBJECTSINGLEORDER}$ implies: (1) $\forall E \in \mathcal{E}, E.\text{vis} \cap \text{ob} = E.\text{ar} \cap \text{ob}$.

Let us now assume that $\text{EVENTUALVISIBILITY}$ did not hold in S . This means that: (2) $\forall a \in E.H, \forall [f] \in E.H / \approx_{ss}: b \in [f] \wedge (a, b) \in \text{rb} \Rightarrow (a, b) \notin E.\text{vis}$, i.e., all rb -ordered operations that do not belong to the same process are not in vis . In that scenario, execution E_3 would match $\neg\text{EVENTUALVISIBILITY}$ in S : $E_3 = (\{o_1 = (p_1, \text{wr}, x, 3, \sqcup, 1, 2), o_2 = (p_2, \text{wr}, x, 2, \sqcup, 1, 2), o_3 = (p_1, \text{rd}, x, \sqcup, 3, 3, 4), o_4 = (p_2, \text{rd}, x, \sqcup, 2, 4, 5), o_5 = (p_1, \text{wr}, x, 5, \sqcup, 6, 7), o_6 = (p_1, \text{rd}, x, \sqcup, 5, 8, 9)\}, \{(o_1, o_3), (o_2, o_4), (o_3, o_5), (o_5, o_6)\}, \{(o_1, o_3), (o_3, o_5), (o_5, o_6), (o_6, o_2), (o_2, o_4)\})$. But now, E_3 violates condition (1), since $E_3.\text{vis} \neq E_3.\text{ar}$. Thus, a contradiction is reached. So, $\text{EVENTUALVISIBILITY}$ must hold in S . Therefore: $\text{PEROBJECTSINGLEORDER} \wedge \text{RVAL}(\mathcal{F}) \Rightarrow \text{EVENTUALVISIBILITY} \wedge \text{RVAL}(\mathcal{F})$. □

Lemma 4. *The $\text{NOCIRCULARCAUSALITY}$ predicate is true in message-based inter-process communication systems.*

Proof. By definition, $hb \equiv (so \cup vis)^+$, vis is acyclic, $so \equiv rb \cap ss$ and $rb \equiv \{(a, b) : a, b \in H \wedge a.rt < b.st\}$. Because of rb , so is also acyclic, since the operations that compose the rb elements are ordered by real time. Thus, to close a cycle, vis needs elements that transitively relate two write operations in a reverse real-time order. This contradicts the vis definition, since vis represents value propagation among processes. Section 2 has stated that such value propagation is message-based. Therefore, when two operations are related by vis , their relation order is consistent with real time. As a result, hb is acyclic. \square

According to Burckhardt, NOCIRCULARCAUSALITY was introduced in his specification of *basic eventual consistency* in order to avoid the *circular causality* [9] (also known as *thin air* [10]) anomaly that may be generated by some compiler optimisations for shared memory multiprocessors (since vis is not acyclic in that context). However, in scenarios where those optimisations cannot be applied, as it is assumed in our paper, such an anomaly cannot happen.

Lemma 5. $PEROBJECTSINGLEORDER \wedge RVAL(\mathcal{F}) \Rightarrow STRONGCONVERGENCE \wedge RVAL(\mathcal{F})$.

Proof. STRONGCONVERGENCE requires that when two read operations ($\forall a, b \in H \mid_{rd}$, on two different processes) have been preceded by the same set of write operations ($vis^{-1}(a) \mid_{wr} = vis^{-1}(b) \mid_{wr}$), then both reads return the same value ($a.oval = b.oval$). PEROBJECTSINGLEORDER implies that every process in S sees the same sequence of write operations on each object. Therefore, every pair of processes that respect PEROBJECTSINGLEORDER have seen the same sequence of values. In that case, when two different processes compare the output value of a read in each process preceded by the same set of write operations, they will certainly obtain the same value. Thus, $PEROBJECTSINGLEORDER \wedge RVAL(\mathcal{F}) \Rightarrow STRONGCONVERGENCE \wedge RVAL(\mathcal{F})$. \square

Theorem 6. $STRONGEVENTUAL(\mathcal{F}) \rightarrow CACHE(\mathcal{F})$.

Proof. Lemmas 3, 4 and 5 imply that: $PEROBJECTSINGLEORDER \wedge RVAL(\mathcal{F}) \Rightarrow STRONGCONVERGENCE \wedge EVENTUALVISIBILITY \wedge NOCIRCULARCAUSALITY \wedge RVAL(\mathcal{F})$. PEROBJECTPRAM is not a relaxing predicate in this scope. Therefore, according to Def. 4, $STRONGEVENTUAL(\mathcal{F}) \rightarrow CACHE(\mathcal{F})$. \square

Corollary 2. $STRONGEVENTUAL(\mathcal{F})$ is weaker than $PROCESSOR(\mathcal{F})$, $SEQUENTIAL(\mathcal{F})$, $TIMEDSERIAL(\mathcal{F})$ and $LINEARISABLE(\mathcal{F})$.

Proof. All mentioned models are stronger than $CACHE(\mathcal{F})$. Relation “weaker than”, according to Def. 4, is transitive. So, because of Theorem 6, $STRONGEVENTUAL(\mathcal{F})$ is weaker than each one of them. \square

- PREFIXSEQUENTIAL(\mathcal{F}) and stronger models.

PREFIXSEQUENTIAL(\mathcal{F}) ensures eventual convergence because it uses a primary manager per set of objects that imposes a common commit write order among all processes. Therefore, $STRONGEVENTUAL(\mathcal{F})$ is weaker than PREFIXSEQUENTIAL(\mathcal{F}), as stated in the following theorem.

Theorem 7. $STRONGEVENTUAL(\mathcal{F}) \rightarrow PREFIXSEQUENTIAL(\mathcal{F})$.

Proof. In LAZYSINGLEORDER, each time a write op becomes committed in an execution E , op is conceptually complete and it becomes visible to other processes. Then, it is included in both $E.vis$ and $E.ar$ relations, in a way that ensures that $E.vis = E.ar$. Thus, $LAZYSINGLEORDER \Rightarrow PEROBJECTSINGLEORDER$. Therefore, Lemmas 3, 4 and 5 imply that: $LAZYSINGLEORDER \wedge RVAL(\mathcal{F}) \Rightarrow STRONGCONVERGENCE \wedge EVENTUALVISIBILITY \wedge NOCIRCULARCAUSALITY \wedge RVAL(\mathcal{F})$. MONOTONICWRITES is not a relaxing predicate in this scope, since it constrains how to build the ar total order in each accepted execution. Therefore, according to Def. 4, $STRONGEVENTUAL(\mathcal{F}) \rightarrow PREFIXSEQUENTIAL(\mathcal{F})$. \square

Corollary 3. $\text{STRONGEVENTUAL}(\mathcal{F}) \rightarrow \text{REGULAR}(\mathcal{F})$.

Proof. $\text{REGULAR}(\mathcal{F})$ is stronger than $\text{PREFIXSEQUENTIAL}(\mathcal{F})$. Relation “weaker than”, according to Def. 4, is transitive. So, because of Theorem 7, $\text{STRONGEVENTUAL}(\mathcal{F})$ is weaker than $\text{REGULAR}(\mathcal{F})$. \square

- $\text{SAFE}(\mathcal{F})$.

In the $\text{SAFE}(\mathcal{F})$ model, a read operation o_r returns the value written in the latest write if there is no write concurrent with o_r . If any concurrent write exists, $o_r.oval$ may be any value. This provides the base for the following theorem.

Theorem 8. $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{SAFE}(\mathcal{F})$.

Proof. Let us assume a system $S = (\{p_1, p_2, p_3\}, \{x\})$. In order to prove that a model A is not weaker than another model B , we should find an execution E_{nw} that complies with B and does not respect A , since this implies that $\mathcal{E}^B \not\subseteq \mathcal{E}^A$. So, $E_4 = (\{o_1 = (p_1, wr, x, 2, \sqcup, 0, 1), o_2 = (p_1, wr, x, 1, \sqcup, 2, 5), o_3 = (p_2, rd, x, \sqcup, 15, 3, 6), o_4 = (p_3, rd, x, \sqcup, 20, 4, 7)\}, \{(o_1, o_2), (o_2, o_3), (o_3, o_4)\}, \{(o_1, o_2), (o_2, o_3), (o_3, o_4)\})$ satisfies those requirements. In that execution, p_1 has written values 2 and 1 on x , in that order. That means that value 2 was known (i.e., visible) to p_1 when it wrote 1. The replication protocol forwarded value 1 to process p_2 while it was written, and p_2 also propagated that value 1 to p_3 . These actions explain the three pairs in the $E_4.vis$ relation.

E_4 complies with all predicates that define $\text{SAFE}(\mathcal{F})$:

1. SINGLEORDER needs that $E_4.vis = E_4.ar$, and that is true,
2. REALTIMEWRITES ($rb \mid_{wr \rightarrow op} \subseteq ar$) means that, in E_4 , (o_1, o_2) , (o_1, o_3) and (o_1, o_4) must be in ar . Transitively, those pairs of operations are in $E_4.ar$.
3. $\text{SEQRVAL}(\mathcal{F})$ allows that both o_3 and o_4 return any value as a result of the read operation, since o_2 is a write concurrent with both of them.

However, E_4 does not respect STRONGCONVERGENCE , since $E_4.vis^{-1}(o_3) \mid_{wr=} E_4.vis^{-1}(o_4) \mid_{wr=} \{o_1, o_2\}$, but $o_3.oval \neq o_4.oval$.

Therefore, $\mathcal{E}^{Safe} \not\subseteq \mathcal{E}^{StrongEventual}$. Then, according to Def. 4, $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{SAFE}(\mathcal{F})$. \square

Let us find out which models are weaker than $\text{SAFE}(\mathcal{F})$ in order to adequately show as many weaker-than relations as possible in Figure 4. To this end, we need to prove a property that was previously conjectured:

Proposition 4. $\text{K-REALTImEREADS}(K)$ is a weakening predicate.

Proof. According to its definition, $\forall a \in H \mid_{wr}, \forall b \in H \mid_{rd}, \forall PW \subseteq H \mid_{wr}, \forall pw \in PW : \mid PW \mid < K \wedge (a, pw) \in ar \wedge (pw, b) \in rb \wedge (a, b) \in rb \Rightarrow (a, b) \in ar$, when b is a read operation, it may get as its result any of the latest K written values.

Let us consider a consistency model M that is built as the conjunction of a set of consistency predicates: $M \equiv \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n$. We may build another consistency model M' with this definition: $M' \equiv \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n \wedge \text{K-REALTImEREADS}(K)$.

Let us assume a generic execution $E \in \mathcal{E}^M$ that consists of at least K write operations ($o_{w1} \dots o_{wK}$, placed in that order in $E.ar$) and with o_r as its first read operation after the latest of those K writes. According to the $\text{RVAL}(\mathcal{F})$ or $\text{SEQRVAL}(\mathcal{F})$ contained in M , $o_r.oval = val$.

$\text{K-REALTImEREADS}(K)$ introduces indeterminism in the output values of read operations. However, according to the definition of $\text{K-REALTImEREADS}(K)$, E still complies with all predicates that define M' . Therefore, $\mathcal{E}^M \subseteq \mathcal{E}^{M'}$. This implies that $E \in \mathcal{E}^{M'}$.

Let us check now whether \mathcal{E}^M is a strict subset of $\mathcal{E}^{M'}$ or both sets are equal. To this end, let us replay E using M' as the consistency model in S . Each one of those replays may generate a result different to val for $o_r.oval$. Now $o_r.oval$ is a value val' , with val' taken from any of the $o_{wx}.ival$ (with $1 \leq x \leq K$) values. All those replays of E with $val' \neq val$ will belong to $\mathcal{E}^{M'}$, but will not be acceptable in \mathcal{E}^M . Therefore, this shows that $\mathcal{E}^M \subset \mathcal{E}^{M'}$ and, according to Def. 4, this proves that $M' \rightarrow M$. \square

Then, the following corollary may be stated.

Corollary 4. $\text{K-SAFE}(\mathcal{F}, K) \rightarrow \text{SAFE}(\mathcal{F})$

$\text{K-REGULAR}(\mathcal{F}, K) \rightarrow \text{REGULAR}(\mathcal{F})$

$\text{K-LINEARISABLE}(\mathcal{F}, K) \rightarrow \text{LINEARISABLE}(\mathcal{F})$

Proof. Every statement in this corollary is a direct consequence of Prop. 4. \square

- **K-REALTIMEReads(K)-based models.**

The following proposition is needed for proving that all models in this set are incomparable to $\text{STRONGEVENTUAL}(\mathcal{F})$.

Proposition 5. $\text{K-REALTIMEReads}(K) \not\rightarrow \text{STRONGCONVERGENCE}$.

Proof. According to the $\text{K-REALTIMEReads}(K)$ definition, $\forall a \in H \mid_{wr}, \forall b \in H \mid_{rd}, \forall PW \subseteq H \mid_{wr}, \forall pw \in PW : |PW| < K \wedge (a, pw) \in ar \wedge (pw, b) \in rb \wedge (a, b) \in rb \Rightarrow (a, b) \in ar$, when b is a read operation, it may get as its result any of the latest K written values.

Let S be a system where no communication problem exists. Let E be an execution that consists of K write operations ($o_{w1} \dots o_{wK}$, placed in that order in $E.ar$) each one writing a different value on object x and with o_{r1} and o_{r2} as two read operations after the latest of those K writes. Since there are no communication problems, $vis^{-1}(o_{r1}) = \{o_{w1} \dots o_{wK}\}$ and $vis^{-1}(o_{r2}) = vis^{-1}(o_{r1})$. However, $o_{r1}.oval \neq o_{r2}.oval$, since $o_{r1}.oval$ may be $o_{w1}.ival$ and $o_{r2}.oval$ be $o_{wK}.ival$ and those values are different. By definition of STRONGCONVERGENCE , STRONGCONVERGENCE becomes false in E . \square

Corollary 5. $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{K-SAFE}(\mathcal{F}, K)$.

Proof. By Prop. 5, $\exists E \in \mathcal{E}, E \models \text{K-SAFE}(\mathcal{F}, K) \wedge E \not\models \text{STRONGEVENTUAL}(\mathcal{F})$. Thus, by Def. 4, $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{K-SAFE}(\mathcal{F}, K)$. \square

Corollary 6. $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{K-REGULAR}(\mathcal{F}, K)$.

Proof. By Prop. 5, $\exists E \in \mathcal{E}, E \models \text{K-REGULAR}(\mathcal{F}, K) \wedge E \not\models \text{STRONGEVENTUAL}(\mathcal{F})$. Thus, by Def. 4, $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{K-REGULAR}(\mathcal{F}, K)$. \square

Corollary 7. $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{K-LINEARISABLE}(\mathcal{F}, K)$.

Proof. Due to Prop. 5, $\exists E \in \mathcal{E}, E \models \text{K-LINEARISABLE}(\mathcal{F}, K) \wedge E \not\models \text{STRONGEVENTUAL}(\mathcal{F})$. Thus, because of Def. 4, $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{K-LINEARISABLE}(\mathcal{F}, K)$. \square

- **Other models.**

A single CAP-constrained model remains unexplored in the CAP-constrained vs CAP-free frontier: $\text{TIMEDCAUSAL}(\mathcal{F}, \Delta)$. Let us assess it in this theorem.

Theorem 9. $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{TIMEDCAUSAL}(\mathcal{F}, \Delta)$.

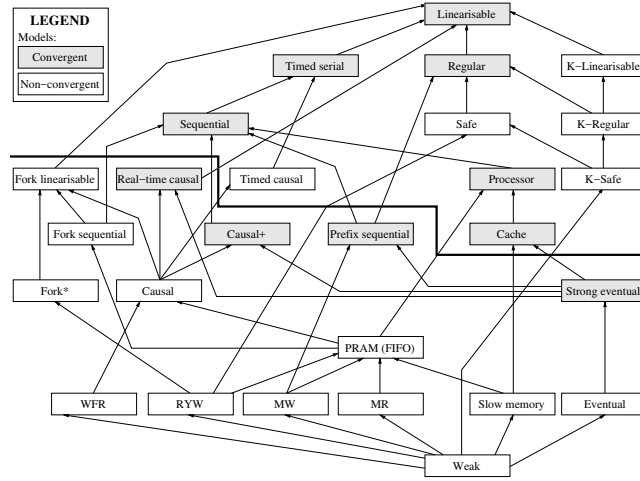


Figure 4: Revised “weaker than” (\rightarrow) relationships among models.

Proof. Let assume the following system and execution: $S = (\{p_1, p_2\}, \{x\})$, $\Delta = 2$, $\exists E_6 \in \mathcal{E}^{TimedCausal}$: $E_6 = (\{o_1 = (p_1, wr, x, 1, \perp, 0, 1), o_2 = (p_2, wr, x, 2, \perp, 0, 1), o_3 = (p_1, rd, x, \perp, 1, 1, 2), o_4 = (p_2, rd, x, \perp, 2, 1, 2), o_5 = (p_1, rd, x, \perp, 2, 3, 4), o_6 = (p_2, rd, x, \perp, 1, 3, 4)\}, \{(o_1, o_3), (o_3, o_5), (o_2, o_4), (o_4, o_6), (o_2, o_5), (o_1, o_6)\}, \{(o_4, o_1), (o_1, o_6), (o_6, o_3), (o_3, o_2), (o_2, o_5)\})$.

$E_6 \models \text{TIMEDCAUSAL}(\mathcal{F}, 2)$, and $vis^{-1}(o_5) \upharpoonright_{wr} = vis^{-1}(o_6) \upharpoonright_{wr} = \{o_1, o_2\}$. But, $o_5.oval \neq o_6.oval$. Therefore, $E_6 \not\models \text{STRONGEVENTUAL}(\mathcal{F})$. This means that, $\mathcal{E}^{TimedCausal} \not\subseteq \mathcal{E}^{StrongEventual}$. Thus, according to Def. 4, $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{TIMEDCAUSAL}(\mathcal{F}, \Delta)$. \square

The focus of this analysis has been set on CAP-constrained models, since their strength suggests that many of them should be convergent. Some notes may be also given about CAP-free models. In that area, Lloyd et al. [31] prove that $\text{CAUSAL+}(\mathcal{F})$ is stronger than $\text{STRONGEVENTUAL}(\mathcal{F})$ and Mahajan et al. [32] prove that $\text{REALTIMECAUSAL}(\mathcal{F})$ also implies $\text{STRONGEVENTUAL}(\mathcal{F})$. It is also known that $\text{CAUSAL}(\mathcal{F})$ is not convergent [31]. So, $\exists E_7 \models \text{CAUSAL}(\mathcal{F}) : E_7 \not\models \text{STRONGEVENTUAL}(\mathcal{F})$, then $\text{STRONGEVENTUAL}(\mathcal{F}) \not\rightarrow \text{CAUSAL}(\mathcal{F})$. By Def. 4, every model M_w weaker than $\text{CAUSAL}(\mathcal{F})$ will not be stronger than $\text{STRONGEVENTUAL}(\mathcal{F})$, since $E_7 \models M_w$ and $E_7 \not\models \text{STRONGEVENTUAL}(\mathcal{F})$. Therefore, the *causal*, *PRAM*, *RYW*, *MW*, *MR* and *slow* models are not stronger than *strong eventual*. This means that they are not inherently convergent. Fork-based models do not comply either with the $\text{STRONGEVENTUAL}(\mathcal{F})$ definition since their NOJOIN (or ATMOSTONEJOIN) prevents $\text{EVENTUALVISIBILITY}$ from becoming true.

Figure 4 shows the *weaker than* relations among $\text{STRONGEVENTUAL}(\mathcal{F})$ and other models. The bold line depicts the frontier presented in Figure 2. Models placed above that frontier are CAP-constrained, while the others are CAP-free. Inherently convergents models are shown in grey boxes, while the others are in white boxes.

As shown in Figure 4, our analysis distributes consistency models in these classes regarding convergence:

1. CAP-constrained and not necessarily convergent: $\text{TIMEDCAUSAL}(\mathcal{F}, \Delta)$, $\text{K-LINEARISABLE}(\mathcal{F}, K)$, $\text{K-REGULAR}(\mathcal{F}, K)$, $\text{SAFE}(\mathcal{F})$, $\text{K-SAFE}(\mathcal{F}, K)$, ... These models lose availability when a network partition arises and do not ensure convergence when no network partition has occurred. This set is the worst one regarding state convergence and CAP freedom.
2. CAP-constrained and convergent: $\text{LINEARISABLE}(\mathcal{F})$, $\text{TIMEDSERIAL}(\mathcal{F}, \Delta)$, $\text{SEQUENTIAL}(\mathcal{F})$, $\text{PROCESSOR}(\mathcal{F})$, $\text{CACHE}(\mathcal{F})$, ... These models lose availability when a network partition arises, but they keep state convergence among processes when no network partition occurs.

3. CAP-free and convergent: $\text{STRONGEVENTUAL}(\mathcal{F})$, $\text{CAUSAL+}(\mathcal{F})$, $\text{REALTIMECAUSAL}(\mathcal{F})$, $\text{PREFIXSEQUENTIAL}(\mathcal{F})$,... These models keep availability when a network partition arises and they are able to reach convergence among all processes when there is no connectivity problem or connectivity is recovered. This is one of the most interesting sets, since these models are able to overcome all CAP constraints without indefinitely losing state convergence.
4. CAP-free and not necessarily convergent: $\text{WEAK}(\mathcal{F})$, $\text{PRAM}(\mathcal{F})$, $\text{EVENTUAL}(\mathcal{F})$, $\text{CAUSAL}(\mathcal{F})$, ... These models keep availability in all processes when a network partition occurs, and this is a good characteristic, but to this end they have renounced to state convergence. They are generally considered too relaxed to implement and deploy scalable services.

5 Related Work

The identification of the set of consistency models affected by the CAP theorem has been implicitly undertaken by several recent papers that have looked for the strongest consistency to be supported in available and partition-tolerant systems [31, 32, 4]. Those papers have taken as a base causal consistency, adding some conditions in order to strengthen it, generating in that way the *causal+* [31], *real-time causal* [32] and *observable causal* [4] models. Those models are incomparable to each other and they define part of the strongest subset of models in the CAP-free set.

Another “classical” approach to implement available and partition-tolerant services is based on *eventual consistency*. The term *eventual consistency* was probably first used in the Clearinghouse system [13] and in the Lotus/Iris Notes project on computer-supported cooperative work [23], in 1987 and 1988, respectively. However, such kind of consistency was already explained and used in other previous papers, being the works from Johnson and Thomas [22, 11] (1975) and the commutative protocol variant of Alsberg and Day [3] (1976) the first ones we are aware of.

Thus, both causal and eventual consistencies belong to the CAP-free set of models. Causal consistency does not demand consensus on a common order of writes, while eventual consistency relaxes the recency of the values being read since it uses lazy write propagation. Intuitively, this suggests that CAP-constrained models are those requiring either consensus on a global write-order (impossible to attain in a partitionable system due to the FLP impossibility result [15]) or a close to immediate recency on the read values (broken when the latest values have been written in another network component while the network is partitioned).

A complete frontier is not easy to set. The consistency specification framework proposed by Burckhardt et al. [10, 9] provides an excellent basis for specifying consistency models. With it, it is easy to specify both safety and liveness conditions. Viotti and Vukolić [43] have used that framework for surveying distributed consistency models. Our work complements their survey looking for the CAP-constrained to CAP-free frontier considering value-order consensus and read recency criteria. Thus, this article extends other previous work focused on setting a partial frontier based on value-order consensus [36].

There are several composable and tunable consistency models [5, 25, 44, 24, 37, 6, 28, 14] whose implementation protocols support both CAP-constrained and CAP-free consistencies. That seems to be the best approach to overcome the limitations imposed by the CAP theorem on the consistency of highly available distributed services. Services that use those models may relax consistency or easily determine which service activities may remain blocked while partitions arise. Additionally, they provide quite a strong consistency while the network remains connected.

6 Conclusions

We have revised which distributed consistency models, besides the *linearisable* one, are directly affected by the CAP theorem constraints on consistency. This study has shown that there are many other models (e.g., *timed serial*, *regular*, *sequential*, *safe*, *timed causal*, *processor*, *cache*, *k-linearisable*,...) that cannot be ensured in available and partition-tolerant distributed services. Therefore, the strongest classical data-centric consistency to be supported by those services in case of network partitions is the *causal* one that may be complemented with other conditions in order to provide stronger semantics in a partitionable system, building in this way the *causal+*, *real-time causal* and *observable causal* models. Many other CAP-free

models have been also identified. Thus, a more precise frontier between CAP-constrained and CAP-free models has been set, providing a good basis for classifying any newly proposed models from now on.

References

- [1] Mustaque Ahamad, James E. Burns, Phillip W. Hutto, and Gil Neiger. Causal memory. In *5th International Workshop on Distributed Algorithms and Graphs (WDAG)*, pages 9–30, Delphi, Greece, October 1991.
- [2] Amitanand S. Aiyer, Lorenzo Alvisi, and Rida A. Bazzi. On the availability of non-strict quorum systems. In *19th International Conference on Distributed Computing (DISC)*, pages 48–62, Cracow, Poland, September 2005.
- [3] Peter Alsberg and John D. Day. A principle for resilient sharing of distributed resources. In *2nd International Conference on Software Engineering (ICSE)*, pages 562–570, San Francisco, CA, USA, October 1976.
- [4] Hagit Attiya, Faith Ellen, and Adam Morrison. Limitations of highly-available eventually-consistent data stores. *IEEE Trans. Parallel Distrib. Syst.*, 28(1):141–155, 2017.
- [5] Hagit Attiya and Roy Friedman. A correctness condition for high-performance multiprocessors. In *24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 679–690, Victoria, British Columbia, Canada, May 1992.
- [6] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. Probabilistically bounded staleness for practical partial quorums. *PVLDB*, 5(8):776–787, 2012.
- [7] Kenneth P. Birman and Roy Friedman. Trading consistency for availability in distributed systems. Technical report, 96-1579, Dept. of Comput. Sc., Cornell University, Ithaca, NY, USA, April 1996.
- [8] Eric A. Brewer. CAP twelve years later: How the “rules” have changed. *IEEE Computer*, 45(2):23–29, February 2012.
- [9] Sebastian Burckhardt. Principles of eventual consistency. *Foundations and Trends in Programming Languages*, 1(1-2):1–150, 2014.
- [10] Sebastian Burckhardt, Alexey Gotsman, Hongseok Yang, and Marek Zawirski. Replicated data types: specification, verification, optimality. In *41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 271–284, San Diego, CA, USA, January 2014.
- [11] B. S. Cosell, Paul R. Johnson, J. H. Malman, Richard E. Schantz, J. Sussman, Robert H. Thomas, and David C. Walden. An operational system for computer resource sharing. In *5th ACM Symposium on Operating System Principles (SOSP)*, pages 75–81, The University of Texas at Austin, Austin, Texas, USA, November 1975.
- [12] Susan B. Davidson, Héctor García-Molina, and Dale Skeen. Consistency in partitioned networks. *ACM Comput. Surv.*, 17(3):341–370, 1985.
- [13] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In *6th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, Vancouver, British Columbia, Canada, August 1987.
- [14] Dan Dobre, Paolo Viotti, and Marko Vukolić. Hybris: Robust hybrid cloud storage. In *ACM Symposium on Cloud Computing (SoCC)*, pages 12:1–12:14, Seattle, WA, USA, November 2014.
- [15] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
- [16] Armando Fox and Eric A. Brewer. Harvest, yield and scalable tolerant systems. In *7th Workshop on Hot Topics in Operating Systems (HotOS)*, pages 174–178, Rio Rico, Arizona, USA, March 1999.
- [17] Seth Gilbert and Nancy Lynch. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

- [18] Alain Girault, Gregor Göbller, Rachid Guerraoui, Jad Hamza, and Dragos-Adrian Seredinschi. Monotonic prefix consistency in distributed systems. In *38th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE)*, pages 41–57, Madrid, Spain, June 2018.
- [19] James R. Goodman. Cache consistency and sequential consistency. Technical report, Number 61, IEEE Scalable Coherent Interface Working Group, March 1989.
- [20] Maurice Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.
- [21] Phillip W. Hutto and Mustaque Ahamad. Slow memory: Weakening consistency to enhance concurrency in distributed shared memories. In *10th International Conference on Distributed Computing Systems (ICDCS)*, pages 302–309, Paris, France, May 1990.
- [22] Paul R. Johnson and Robert H. Thomas. The maintenance of duplicate databases. RFC 677, Network Working Group, Internet Engineering Task Force, January 1975.
- [23] Leonard Kawell, Jr., Steven Beckhardt, Timothy Halvorsen, Raymond Ozzie, and Irene Greif. Replicated document management in a group communication system. In *ACM Conference on Computer-Supported Cooperative Work (CSCW)*, pages 395–, Portland, Oregon, USA, 1988.
- [24] Sudha Krishnamurthy, William H. Sanders, and Michel Cukier. An adaptive framework for tunable consistency and timeliness using replication. In *International Conference on Dependable Systems and Networks (DSN)*, pages 17–26, Bethesda, MD, USA, June 2002.
- [25] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Trans. Comput. Syst.*, 10(4):360–391, November 1992.
- [26] Leslie Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE T. Comput.*, 28(9):690–691, 1979.
- [27] Leslie Lamport. On interprocess communication. Part II: algorithms. *Distrib. Comput.*, 1(2):86–101, 1986.
- [28] Cheng Li, Daniel Porto, Allen Clement, Johannes Gehrke, Nuno M. Preguiça, and Rodrigo Rodrigues. Making geo-replicated systems fast as possible, consistent when necessary. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–278, Hollywood, CA, USA, October 2012.
- [29] Jinyuan Li and David Mazières. Beyond one-third faulty replicas in byzantine fault tolerant systems. In *4th Symposium on Networked Systems Design and Implementation (NSDI)*, pages 131–144, Cambridge, Massachusetts, USA, April 2007.
- [30] Richard J. Lipton and Jonathan S. Sandberg. PRAM: A scalable shared memory. Technical report, CS-TR-180-88, Princeton University, USA, April 1988.
- [31] Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. Don’t settle for eventual: scalable causal consistency for wide-area storage with COPS. In *23rd ACM Symposium on Operating Systems Principles (SOSP)*, pages 401–416, Cascais, Portugal, October 2011.
- [32] Prince Mahajan, Lorenzo Alvisi, and Mike Dahlin. Consistency, availability and convergence. Technical report, UTCS TR-11-22, Dept. of Computer Science, The University of Texas at Austin, USA, 2011.
- [33] David Mazières and Dennis E. Shasha. Building secure file systems out of byzantine storage. In *21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 108–117, Monterrey, California, USA, July 2002.
- [34] Francesc D. Muñoz-Escóí and José M. Bernabéu-Aubán. A survey on elasticity management in PaaS systems. *Computing*, 99(7):617–656, 2017.
- [35] Alina Oprea and Michael K. Reiter. On consistency of encrypted files. In *20th International Symposium on Distributed Computing (DISC)*, pages 254–268, Stockholm, Sweden, September 2006.
- [36] Leticia Pascual-Miret, José Ramón González de Mendivil, José M. Bernabéu-Aubán, and Francesc D. Muñoz-Escóí. Widening CAP consistency. Technical report, IUMTI-SIDI-2015/03, Inst. Univ. Mixto Tecnológico de Informática, Univ. Politècnica de València, Spain, 2015.

- [37] Nuno Santos, Luís Veiga, and Paulo Ferreira. Vector-field consistency for ad-hoc gaming. In *ACM/IFIP/USENIX 8th International Middleware Conference*, pages 80–100, Newport Beach, CA, USA, November 2007.
- [38] Marc Shapiro, Nuno M. Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 386–400, Grenoble, France, October 2011.
- [39] Douglas B. Terry, Alan J. Demers, Karin Petersen, Mike Spreitzer, Marvin Theimer, and Brent B. Welch. Session guarantees for weakly consistent replicated data. In *3rd International Conference on Parallel and Distributed Information Systems (PDIS)*, pages 140–149, Austin, Texas, USA, September 1994.
- [40] Douglas B. Terry, Marvin Theimer, Karin Petersen, Alan J. Demers, Mike Spreitzer, and Carl Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *15th ACM Symposium on Operating System Principles (SOSP)*, pages 172–183, Copper Mountain Resort, Colorado, USA, December 1995.
- [41] Francisco J. Torres-Rojas, Mustaque Ahamad, and Michel Raynal. Timed consistency for shared distributed objects. In *18th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 163–172, Atlanta, Georgia, USA, May 1999.
- [42] Francisco J. Torres-Rojas and Esteban Meneses. Convergence through a weak consistency model: Timed causal consistency. *CLEI Electron. J.*, 8(2):2:1–2:10, 2005.
- [43] Paolo Viotti and Marko Vukolić. Consistency in non-transactional distributed storage systems. *ACM Comput. Surv.*, 49(1):19:1–19:34, 2016.
- [44] Haifeng Yu and Amin Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Trans. Comput. Syst.*, 20(3):239–282, 2002.