PNUTS notes

Similar motivation to Dynamo and Megastore:

Multiple data centers
Variety of user-oriented applications (e.g., user account mgt, status updates)
High availability
Scalability
Low latency reads and writes

First, a quick overview of their system, then a comparison to Dynamo and MegaStore.

Consistency only wrt an individual record (e.g., user account info). Record is a bag of bits – no updates allowed, only complete replacement. No transactional support across records.

By contrast to Megastore (serializable) or Dynamo (eventually consistent), PNUTS has a middle approach:

"Timeline-based consistency"

Think of a set of updates by various nodes. With serializability, both reads and writes appear in order, but that means all reads/writes need to be totally ordered.

With eventual consistency, updates can be applied immediately, before being propagated to other data centers. So if two concurrent updates, W1 and W2, one DC can see W1 and the other DC can see W2.

To be eventually consistent, these need to be applied in the same order everywhere – eventually, everyone should see either W1 or W2 (or in Dynamo's case, a reconciliation of W1 and W2). But in the meantime, some reads might see W0, W1 or W2, depending on which copy they read.

This might be a problem if, say, the two updates are related, e.g., change privacy settings and upload a sensitive photo.

With timeline based consistency, updates are constrained to occur in the same order everywhere. They accomplish this by sending all updates to a single location (different for every record), which orders writes. But the write can finish early – it doesn't need to be sent to every data center before returning.

So can think of it as each record going through a series of versions, v0, v1, v2, v3….

API allows you to constrain reads. Eg.,

read any old version

read a version at least as recent as x

read latest version

and you can constrain writes

write regardless of current version

or

write a version if the current version is x (compare and update)

The latter is similar to OCC – only commit if no intervening write since prior read.

Thus, reads see some old version of the data, not necessarily the latest, and so can be fast.

Manager can migrate on each write – so it can move to the data center where the data is being actively modified (where the user is), and if that changes, it can move again. Thus, writes are usually local.

All writes go to the manager, read any can read any version, read recent reads locally if its nearly up to date, or otherwise goes to the manager.

Data centers can cache the manager location for each record, or consult a manager manager who keeps track

Question: easier to use than eventually consistent?

Implementation using reliable asynchronous message passing. Each update to a record is a reliable (exactly once) message, that can be delivered asynchronously to the other data centers. The message queue essentially serves as the commit log.

On to the comparison:

	PNUTS	Dynamo	Megastore	WWYD
Record update: Consistency model				
Cross-record update consistency model				
Intra-record update allowed?				
Record lookup procedure				
Read (best case)				
Write (best case)				
Read (worst case)				
Write (worst case)				
Data store failure				
Local manager failure				
Data center failure				
Partial network failure				
Network partition				