System Design: почему нельзя все и сразу (CAP и PACELC)

Компромиссы в техническом дизайне

Любое техническое решение включает в себя ряд условностей и компромиссов, которые важно понимать самим и уметь доносить до команды и менеджмента.

Говоря о дизайне систем, обычно обсуждают такие факторы, как стоимость разработки и поддержки, отказоустойчивость, доступность, масштабируемость и другие.

Как нетрудно догадаться, очень сложно, а иногда и вовсе невозможно, добиться всех этих условий одновременно.

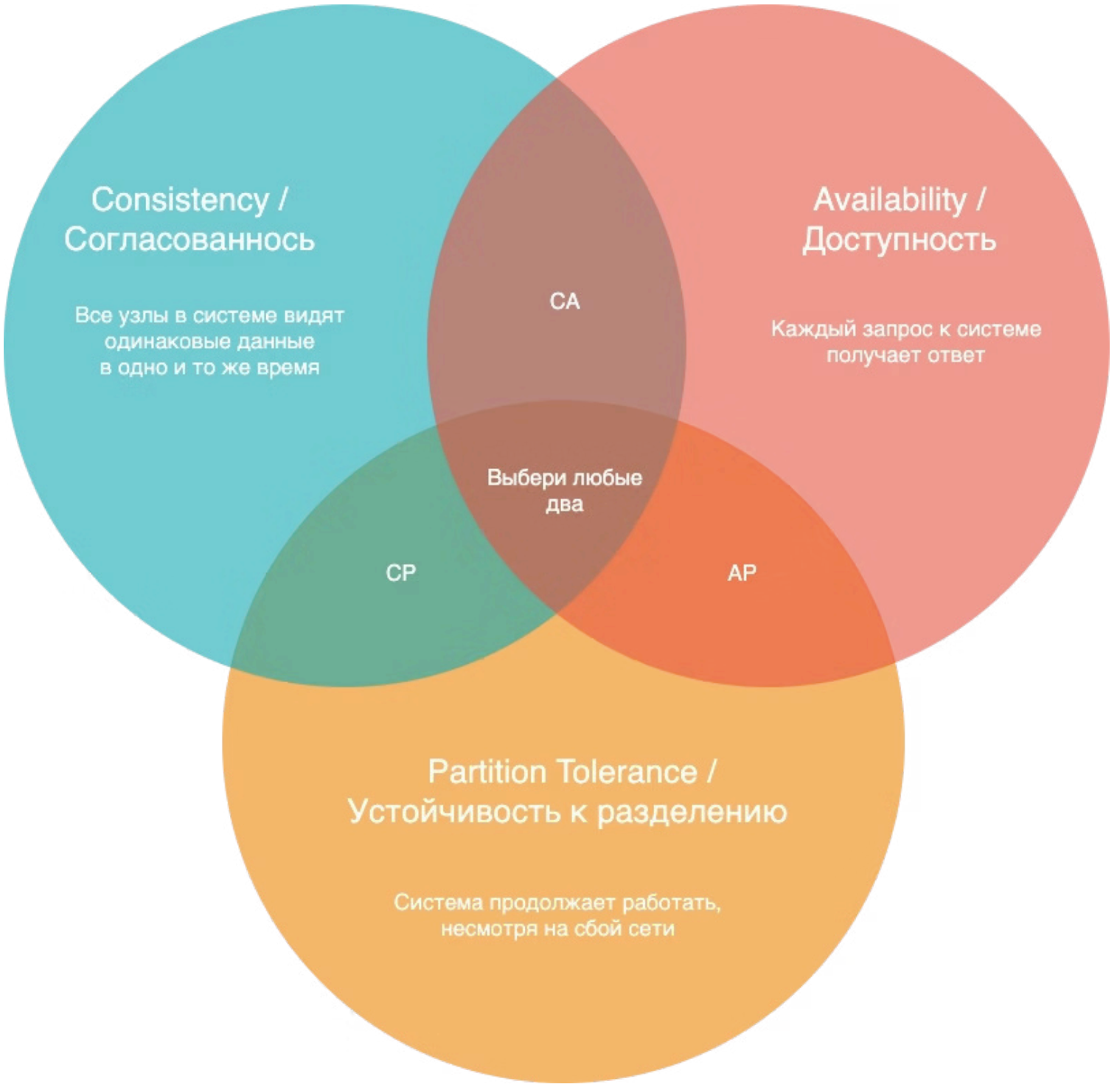
Собственно, **CAP-теорема** и её расширение **PACELC** — это простой и наглядный пример такого компромисса.

CAP теорема

Обратимся к определению из Википедии.

CAP-теорема (также известна как теорема Брюэра) — эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств:

- **Согласованность данных (consistency)** — во всех вычислительных узлах в один и тот же момент времени данные не противоречат друг другу.
- **Доступность (availability)** — любой запрос к распределённой системе завершается откликом, однако без гарантии, что ответы всех узлов совпадают.
- **Устойчивость к фрагментации (partition tolerance)** — расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из них.



Вопросы после определения

На мой взгляд, после такого определения у нормального человека возникает только больше вопросов:

- [ ] Почему только два из трёх? А почему нельзя всё сразу?
- [ ] А что значит "согласованность"? Это как в ACID?
- [ ] Что такое "разделение сети"? Это типа интернет отвалился?
- [ ] Как разработчику понять, что важнее: C, A или P? Есть какие-то советы?

Давайте попробуем разобраться.

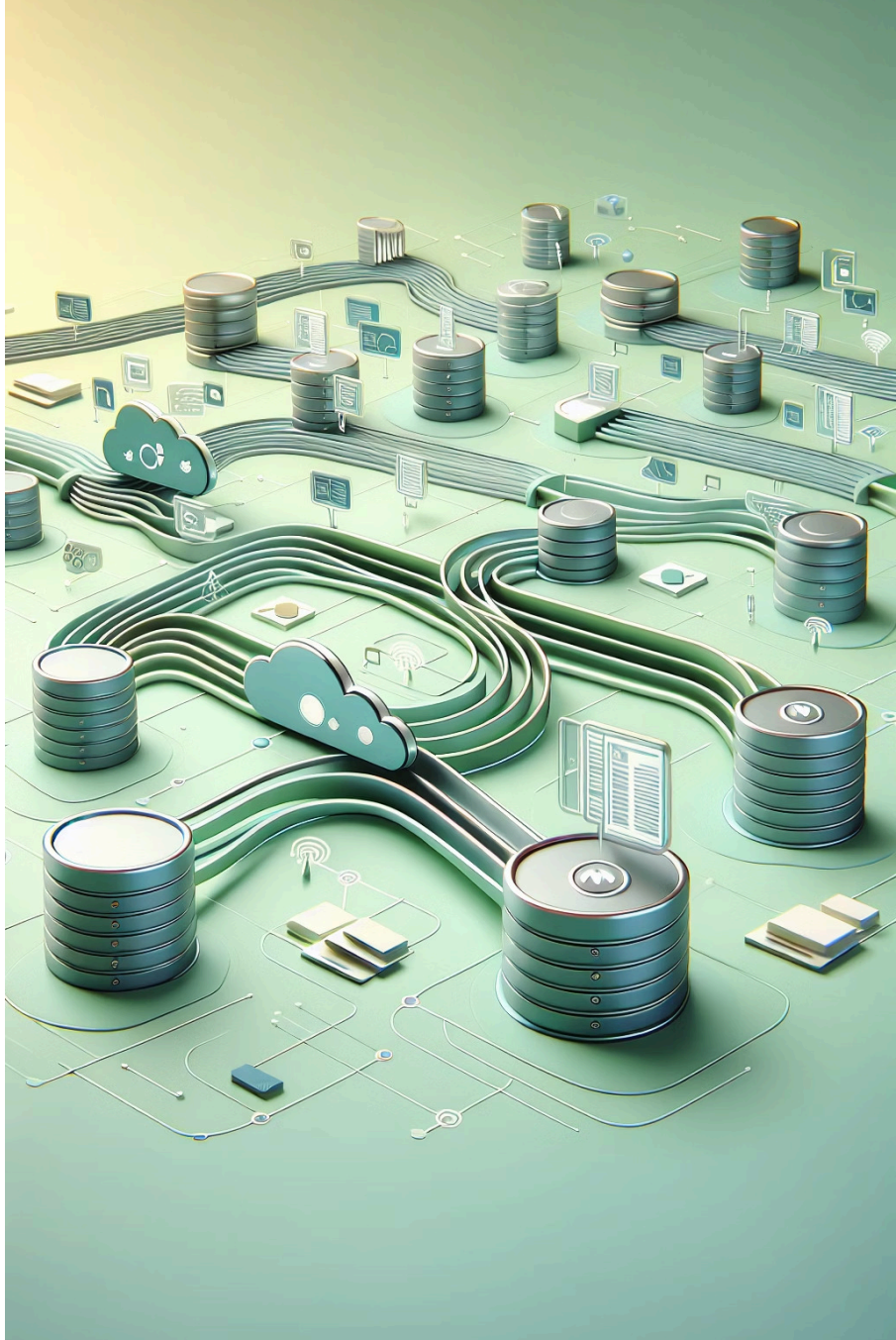
> CAP-теорему часто неправильно трактуют как требование всегда отказываться от одной из трёх гарантий. На самом деле выбор между согласованностью и доступностью встаёт **только в случае сетевого разделения или сбоя**.

Что это означает на практике?

CP системы: сильная согласованность

Если система **CP** (*сильная согласованность / Strong Consistency*), то в случае ошибок сети вы выбираете согласованность данных. Если одна из двух нод упала, то часть пользователей может не достучаться до вашего приложения — зато целостность данных не нарушится.

> Под **целостностью данных** здесь понимается следующее: если вы записали что-то в базу, то именно это и будет там храниться. Более того, после восстановления сети не возникнет конфликтов в данных — например, не получится так, что на одной ноде у аккаунта баланс 100, а на другой — 200.



AP системы: высокая доступность

Если же система **AP** (*высокая доступность / High Availability*), то в случае сетевых проблем приоритет отдаётся доступности — в ущерб согласованности.

На практике это означает, что вы **всегда** сможете что-то записать или прочитать, но:

- прочитанные данные могут быть устаревшими;
- записанные вами изменения могут не сразу отобразиться другим пользователям.

Существует множество "слабых" моделей согласованности, каждая со своими нюансами — но мы сейчас их рассматривать не будем.

Как уже было сказано, всё это имеет смысл **только при наличии сетевых сбоев или разделений**.

Во все остальные моменты компромисс следует рассматривать уже с точки зрения **PACELC-теоремы**.

PACELC

Теорема PACELC — это более тонкая и детализированная версия теоремы CAP.

Она утверждает, что:

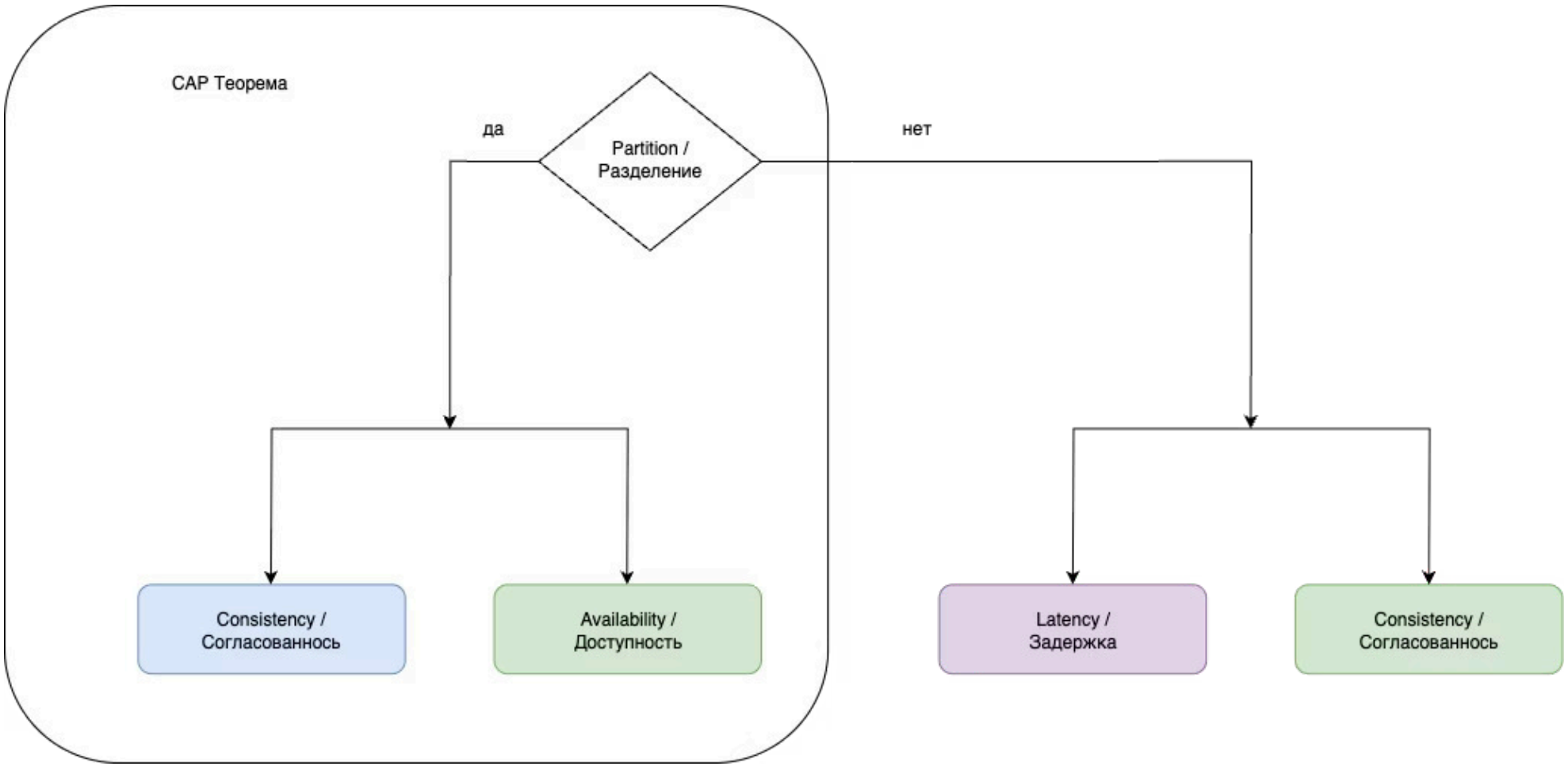
В случае сетевого разделения (Partition)

в распределённой системе необходимо выбирать между **доступностью (Availability)** и **согласованностью (Consistency)** — как это и описано в классической теореме CAP;

В остальных случаях

— то есть когда система работает нормально и разделений нет — приходится делать выбор между **задержкой (Latency)** и **согласованностью (Consistency)**.

Такой компромисс возникает естественным образом: для обеспечения устойчивости к сбоям и разделениям данные и сервисы реплицируются, зачастую — между дата-центрами или географически удалёнными узлами. Это, в свою очередь, приводит к необходимости выбирать между уровнем согласованности и связанной с этим задержкой



Синхронная vs асинхронная репликация

Если говорить простыми словами, то при стремлении обеспечить **сильную согласованность** системе необходимо использовать **синхронную репликацию**, гарантируя, что все реплики содержат актуальные данные.

Например, если у нас есть три сервера, и на один из них приходит запрос на увеличение баланса пользователя на $100, то:

01

данные сначала записываются на этот сервер;

02

затем аналогичный запрос отправляется двум другим репликам;

03

система дожидается подтверждения записи от всех трёх узлов;

04

и **только после этого** считает операцию завершённой.

Это естественным образом увеличивает **задержку (latency)**, пропорционально количеству реплик.

С другой стороны, если система использует **асинхронную репликацию** и не дожидается ответа от других нод, то обеспечить строгую согласованность **в принципе невозможно**.

В таких случаях система поддерживает так называемую **eventual consistency** — то есть **со временем** все реплики приходят к актуальному состоянию и отражают последнюю версию данных.

Заключение

Теоремы **CAP** и **PACELC** — это важные концепции при проектировании распределённых систем. Они задают рамки для понимания **неизбежных компромиссов**, с которыми сталкиваются инженеры при создании высокодоступных систем с требованиями к согласованности данных.