CAP teorém

CAP teorém, známý také jako Brewerův teorém, je fundamentální princip v počítačové vědě, který definuje teoretické hranice při návrhu distribuovaných datových systémů. Koncept poprvé formuloval počítačový vědec Eric Brewer v roce 2000 a v roce 2002 jej matematicky dokázali výzkumníci Seth Gilbert a Nancy Lynch z MIT.

Základní myšlenka teorému spočívá v tom, že v jakémkoliv distribuovaném systému (systému, který běží na více vzájemně propojených serverech nebo uzlech) je fyzicky a logicky nemožné současně garantovat všechny tři následující vlastnosti.

Tři základní vlastnosti (C-A-P)

Název teorému je akronymem složeným z počátečních písmen anglických názvů těchto tří klíčových vlastností:

C (Consistency – Konzistence): Konzistence zaručuje, že každý požadavek na čtení dat vrátí buď nejnovější zapsanou hodnotu, nebo vrátí chybovou hlášku. To v praxi znamená, že jakmile dojde k zápisu dat na jednom uzlu, všechna následná čtení z jakéhokoliv jiného uzlu v síti musí tuto novou změnu reflektovat. Všichni uživatelé vidí stejná data ve stejný čas.

A (Availability – Dostupnost): Dostupnost znamená, že každý dotaz odeslaný na fungující uzel (server) v systému dostane platnou odpověď, aniž by došlo k chybě nebo timeoutu. Systém „vždy odpoví“, nicméně u této vlastnosti není garantováno, že vrácená odpověď obsahuje ta absolutně nejnovější data.

P (Partition Tolerance – Tolerance k rozdělení sítě): Tolerance k výpadkům sítě zajišťuje, že systém nadále funguje a plní své úkoly i přesto, že dojde ke zpoždění zpráv nebo k úplnému výpadku komunikace mezi některými uzly. Znamená to, že cluster je schopný operovat, i když je síťově rozdělen na několik izolovaných částí.

Realita cloudového věku: Výběr mezi CP a AP

Historicky se CAP teorém často učil jako pravidlo „vyberte si dvě vlastnosti ze tří“. V prostředí moderního internetu a cloudových služeb je však tento výklad nepřesný.

Vzhledem k tomu, že systémy komunikují přes nespolehlivou síť (internet), tolerance k rozdělení sítě (P) není volitelná, ale je to nutnost. Sítě mohou kdykoliv selhat kvůli poškozenému kabelu, vadnému routeru nebo přetíženým linkám. Protože je vlastnost „P“ daná, architekt systému se při výpadku spojení musí rozhodnout, kterou ze zbylých dvou vlastností systém obětuje. Databáze se tak dělí do dvou hlavních kategorií:

CP systémy (Konzistence a Tolerance k rozdělení)

Když dojde k výpadku sítě mezi servery, systém raději odmítne zpracovat požadavek (obětuje dostupnost), aby zabránil vrácení neaktuálních dat nebo vzniku konfliktů. Zajišťuje absolutní přesnost.

Příklad z praxe: Bankovní a finanční aplikace. Pokud bankomat ztratí spojení s centrální databází, neumožní vám vybrat peníze. Nedostupnost je v tomto případě bezpečnější varianta než riziko, že by vám bankomat vydal hotovost, kterou na účtu fyzicky nemáte.

Příklady databází: MongoDB, Redis, Apache HBase.

AP systémy (Dostupnost a Tolerance k rozdělení)

Při výpadku sítě systém nadále ochotně odpovídá na všechny dotazy uživatelů, ačkoliv si je vědom, že nedokáže zkontrolovat aktuálnost dat u ostatních izolovaných uzlů. Systém upřednostňuje plynulý chod a vrácení zastaralých informací před výpadkem služby. Jakmile je síť opravena, data se dodatečně synchronizují (tzv. konečná konzistence - Eventual Consistency).

Příklad z praxe: E-shopy nebo sociální sítě. Pokud si na e-shopu přidáte zboží do košíku, systém požadavek okamžitě přijme. Pokud má příspěvek na sociální síti o několik vteřin zpožděný počet „lajků“ kvůli synchronizaci, uživatele to nijak zásadně neovlivní.

Příklady databází: Amazon DynamoDB, Apache Cassandra, CouchDB.

CA systémy (Konzistence a Dostupnost)

Tato kategorie nemůže bezpečně fungovat na distribuované síti. Patří sem typicky relační databáze (SQL) běžící na jednom jediném fyzickém serveru, kde z logiky věci nemůže dojít k rozdělení sítě mezi uzly.

Příklady databází: Tradiční nasazení PostgreSQL, MySQL nebo Oracle.

Rozšíření PACELC

S rostoucí komplexností cloudových architektur přestal být samotný CAP teorém dostačující, protože řeší pouze chování při výpadku sítě (který nastává zřídka). V roce 2010 proto Daniel Abadi představil rozšířený model PACELC.

Tento model říká: Pokud dojde k výpadku sítě (Partition), systém volí mezi Availability a Consistency (to je standardní CAP). Else (Jinak - pokud síť běží normálně a bez výpadků), systém volí mezi Latency (Rychlostí, tedy snížením zpoždění) a Consistency (Konzistencí).

I při dokonale fungující síti totiž okamžitá synchronizace dat napříč kontinenty zabere fyzický čas. Architekti tak musí balancovat, zda systém uživateli odpoví bleskově (s rizikem starších dat), nebo si

vyžádá pár milisekund navíc, aby zaručil naprostou přesnost.

Související pojmy: Eric Brewer, Distribuované systémy, Databáze, NoSQL, Eventual Consistency, PACELC teorém, ACID, BASE, DynamoDB, Cassandra, MongoDB.