

PDV 08 2025/2026

Úvod do distribuovaných systémů

Michal Jakob

michal.jakob@fel.cvut.cz

Centrum umělé inteligence, katedra počítačů, FEL ČVUT





Paralelní
výpočty

Distribuované
výpočty

Plán

T	Přednáška
8.4.	Úvod do distribuovaných systémů (DS). Detekce selhání.
15.4.	Detekce selhání. Čas a uspořádání v DS.
22.4.	Globální stav a distribuovaný snapshot.
29.4.	Vyloučení procesů. Konsensus.
6.5.	Konsensus. Raft.
13.5.	-- volno --
20.5.	Volba lídra. Shrnutí předmětu.

Studijní materiály

Slidy

Hlavní kniha: Maarten van Steen, Andrew S. Tanenbaum: **Distributed Systems (4.03 Edition)**, 2025, k dispozici online: <https://www.distributed-systems.net/index.php/books/ds4/>

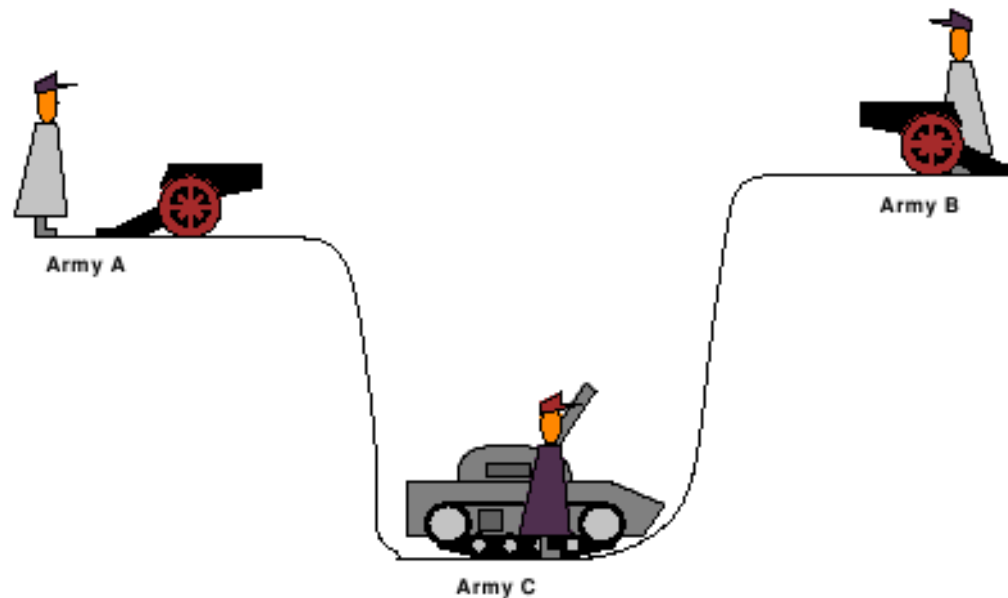
Sekundární kniha: George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair: **Distributed Systems: Concepts and Design (5th Edition)**, 2011

⚠ Cvičení a semestrální práce jsou v **Javě** (nainstalovat a zprovoznit IntelliJ IDEA)



Úvod do Distribuovaných Systemů

Dva generálové



Pouze **synchronizovaný útok uspěje** → generálové se potřebují dopředu **shodnout na čase**, kdy oba zaútočí.

Komunikují skrze **posílání zpráv**. Poslané zprávy se mohou **ztratit...**

Jak **zaručit**, že zaútočí ve stejný čas?

Dva generálové

Možné řešení:

- Generál A: zaútočit za úsvitu!
- Generál B: potvrzují, zaútočit za úsvitu!
- Generál A: potvrzují, potvrzují, zaútočit za úsvitu!
- Generál B: potvrzují, potvrzují, potvrzují, zaútočit za úsvitu!
- ...

Lze ukázat:

Řešení problému Dvou generálů za předpokladu nespolehlivého kanálu **neexistuje!**

Reality Check

Příklad: Výběr z bankomatu

- **vyberete** si z bankomatu v Praze 1000 Kč
- ze zůstatku vašeho účtu se v datovém centru **odečte** 1000 Kč

Problém **atomického commitu**: atomický commit je sada více operací, které jsou provedeny jako jedná operace.

- zúčastněné systémy musí zkoordinovat zda a kdy tyto operace budou provedeny

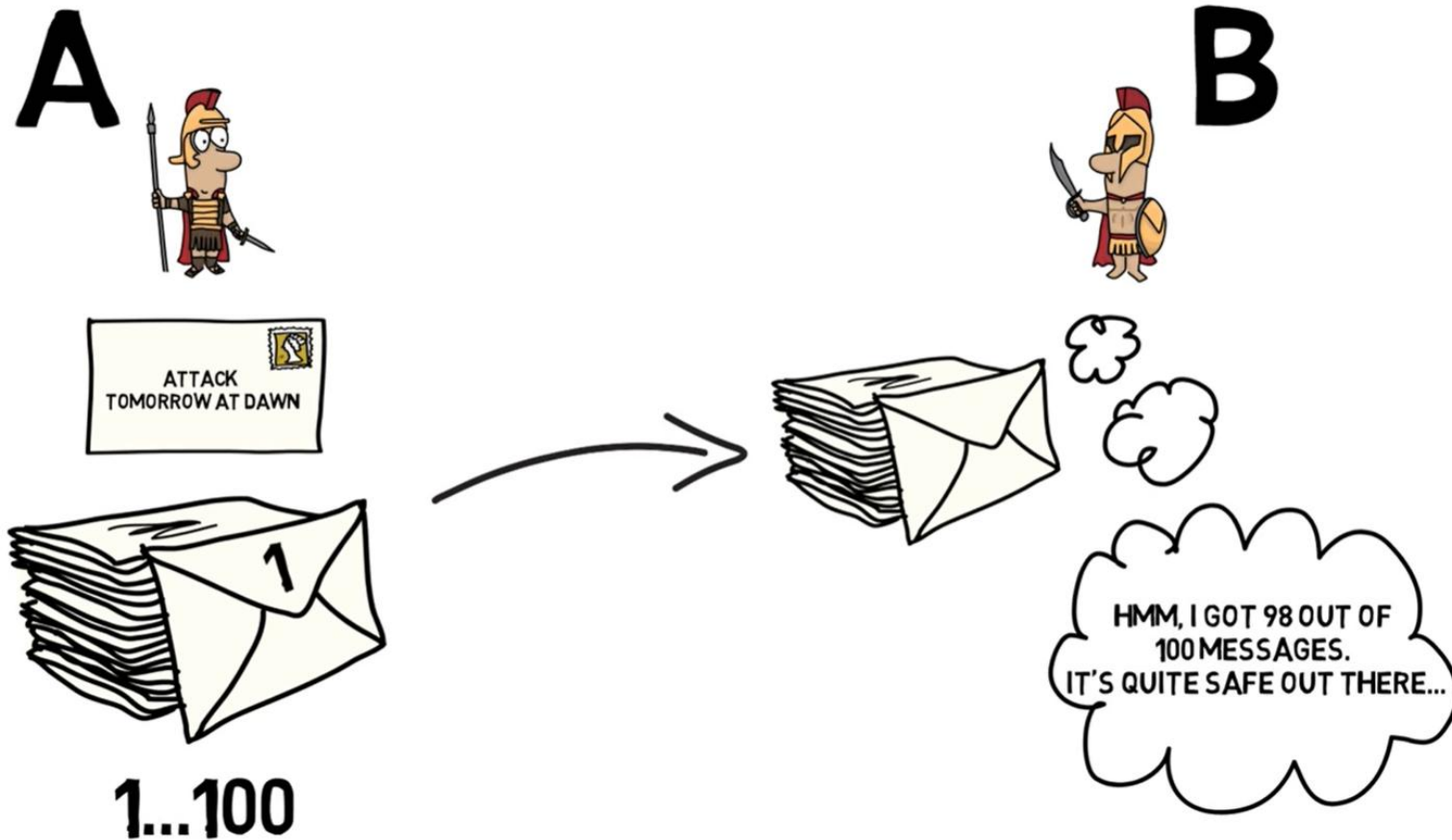
Pragmatické řešení problému Dvou Generálů



Musíme **akceptovat nespolehlivost** komunikačního kanálu a nesnažit se ji nejistotu eliminovat, ale **snížit na přijatelnou** mez

Předpokládejme, že **pravděpodobnost chycení** kurýra je p a že opakované zachycení kurýra jsou **nezávislé** jevy.

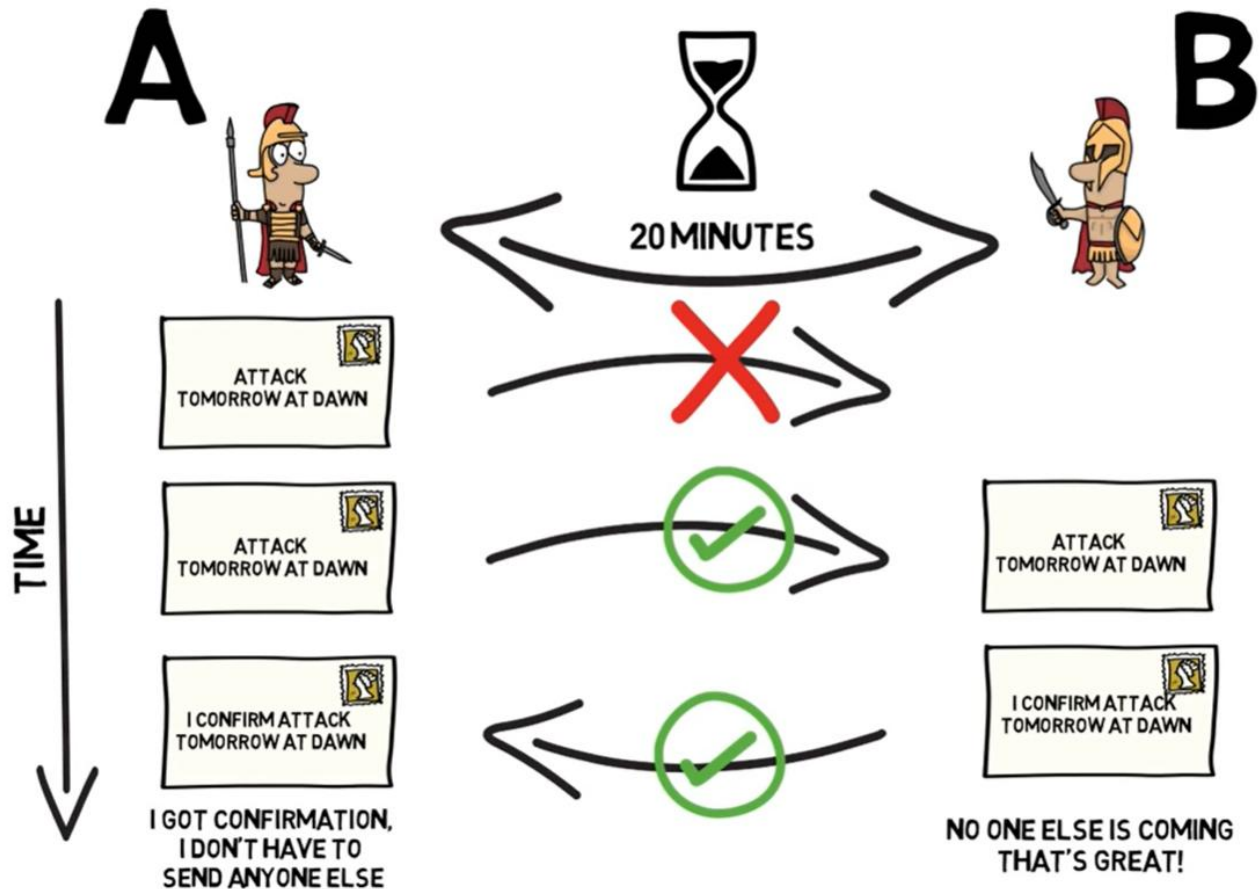
Pragmatické řešení



Drahé

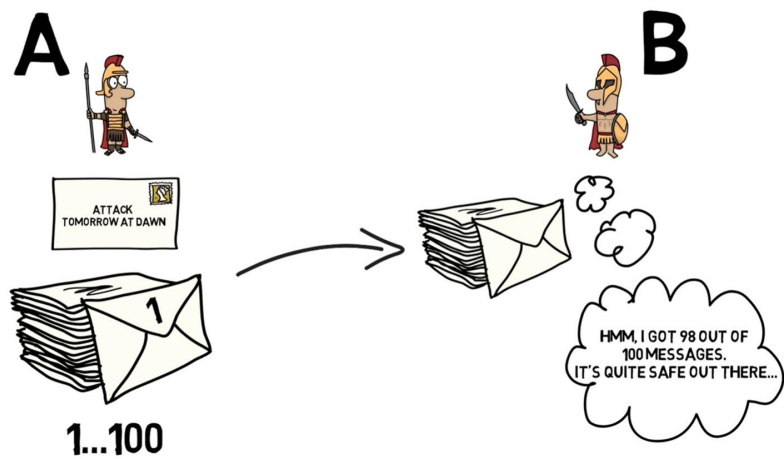
Alternativní přístup

WHAT IF WE DON'T WANT TO SACRIFICE SO MANY MESSENGERS?

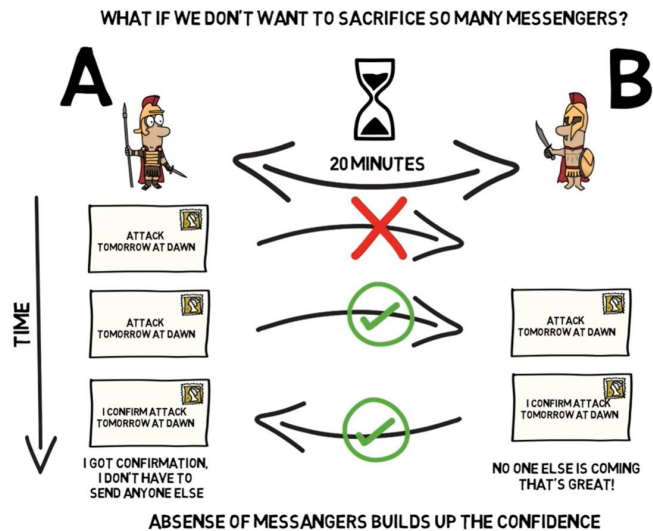


ABSENCE OF MESSENGERS BUILDS UP THE CONFIDENCE

Dva přístupy



drahé



pomalé

Obecně

Řešitelnost problémů v distribuovaných systémech

- Řada zdánlivě jednoduchých problémů nemá v distribuovaných systémech 100% řešení...
- ...ale mají pragmatická řešení, často ale s různými trade-offs
- Některé problémy 100% řešení mají...
- ... ale jen za určitých předpokladů.

Pochopit, které problémy jsou které, a jak se dají řešit (s ohledem na různé možné cíle) je cílem tohoto předmětu.



Co to je distribuovaný
systém?

Distribuované systémy jsou všude

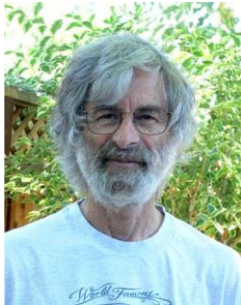
Od té doby, co existují počítačové sítě (natož Internet!) už prakticky **neexistují systémy**, které by **nebyly** aspoň částečně **distribuované**.

Co to je DS?



Definice (optimistická)

Soubor (*collection*) **autonomních výpočetních jednotek**, které se uživateli jeví jako **jeden koherentní systém**. [Andrew Tanenbaum]



Definice (pesimistická)

Systém, ve kterém selhání počítače, o které jste vůbec nevěděli, že existuje, učiní váš vlastní počítač nepoužitelný. [Leslie Lamport]

Definice (pragmatická)

Soubor (*collection*) nezávislých, **autonomních výpočetních jednotek** propojených **komunikační sítí**. Výpočetní elementy komunikují formou **posílání zpráv** za účelem určité formy **spolupráce**.

Příklady distribuovaných systémů



Mezibankovní
platby

Multi-player
games

Uber/Bolt

Sensorové
sítě

Blockchain
Sítě

LLMs a jejich
trénink

Řízení výrobní
linky

Datové
centrum

Internet

Charakteristiky DS

V distribuovaném systému:

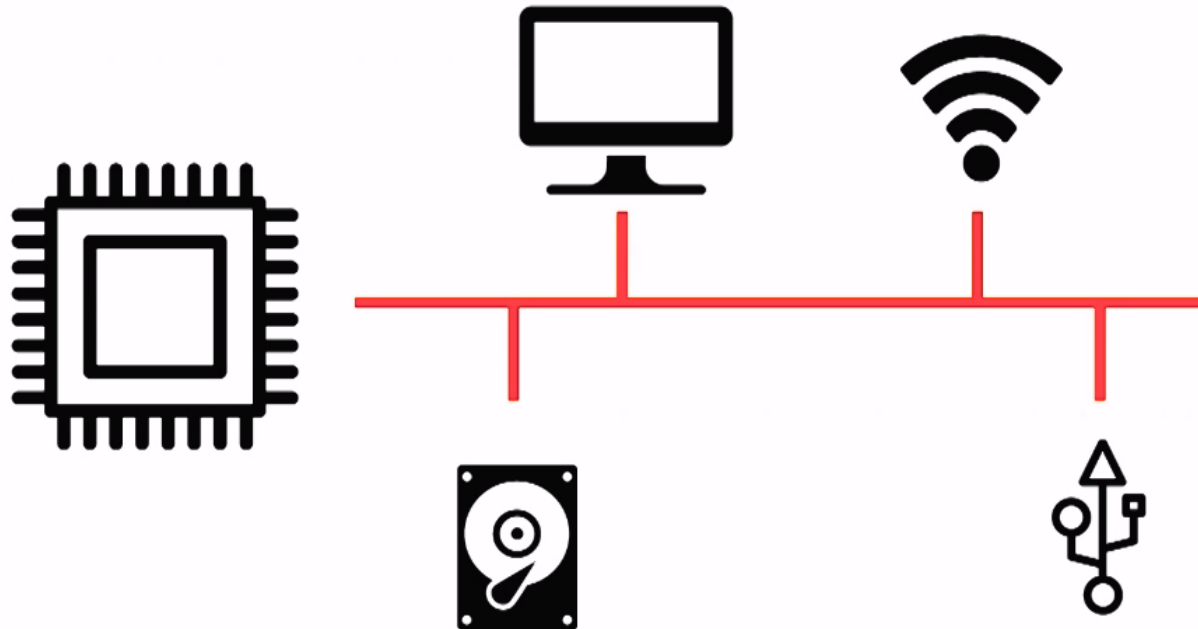
1. Výpočetní jednotky nemají **sdílenou paměť**
2. Výpočetní jednotky **nesdílejí** globální **hodiny**
3. Výpočetní jednotky **selhávají nezávisle**

Naopak:

Paralelní systémy sdílejí

1. **Paměť**: předávání stavu používá sdílenou paměť a synchronizační mechanismy
2. **Hodiny**: přístup ke společným hodinám
3. **Osud**: selže buď nic nebo vše

Je počítač distribuovaný systém (z pohledu programátora aplikací)?



Není* → Proč?

(*záleží na úrovni, na které se počítačem pracujeme)

Cíle při vývoji DS

Výkon/Škálovatelnost

Schopnost řešit **více úloh** nebo **větší instance** úloh, než je možné s jedním počítačem

Podobné jako u paralelních výpočtů, ale řeší i případy, když už jeden počítač nestačí.

Spolehlivost/Dostupnost

Schopnost zajistit **(téměř) trvalou dostupnost** požadovaných služeb

U paralelních výpočtů tento cíl typicky nemáme

zaměření distribuované části PDV

Další cíle: otevřenost, bezpečnost, sdílení zdrojů, geografická dostupnost...

Nezávislá (a částečná) selhání

Nezávislá selhání komplikují život vývojářům

Algoritmy je třeba navrhnout tak, aby s možností selhání počítaly a byly **robustní vůči selhání**.

n závislých procesů,
pravděpodobnost selhání $p \rightarrow$
dostupnost $(1 - p)^n$



Nezávislá selhání usnadňují život uživatelům

Replikace pomocí více nezávislých počítačů **zvyšuje odolnost** vůči selhání

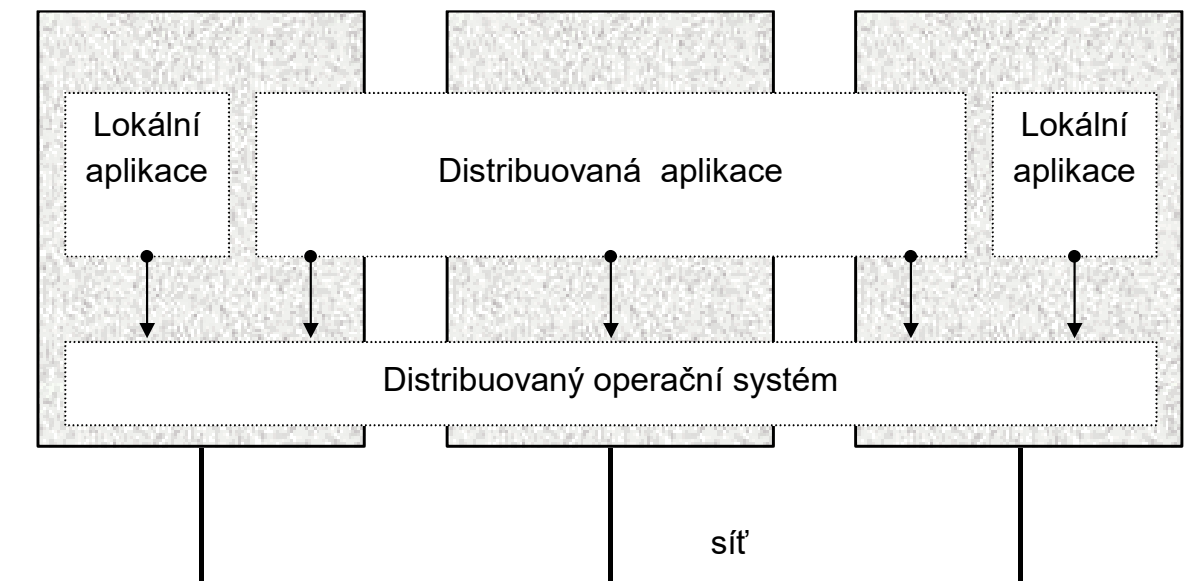
n nezávislých procesů,
pravděpodobnost selhání $p \rightarrow$
dostupnost $1 - p^n$

199x: Distribuované operační systémy

Operační systém vyvinutý speciálně pro potřeby DS

- “zlatý věk”: Amoeba*, Sprite, V, Chorus, Mosix ... T4
- **součástí jádra** OS podpora distribuovanosti, komunikace, synchronizace
- aplikace: **transparentní** využití distribuovaných služeb

Neexistuje rozdíl mezi lokální a distribuovanou aplikací



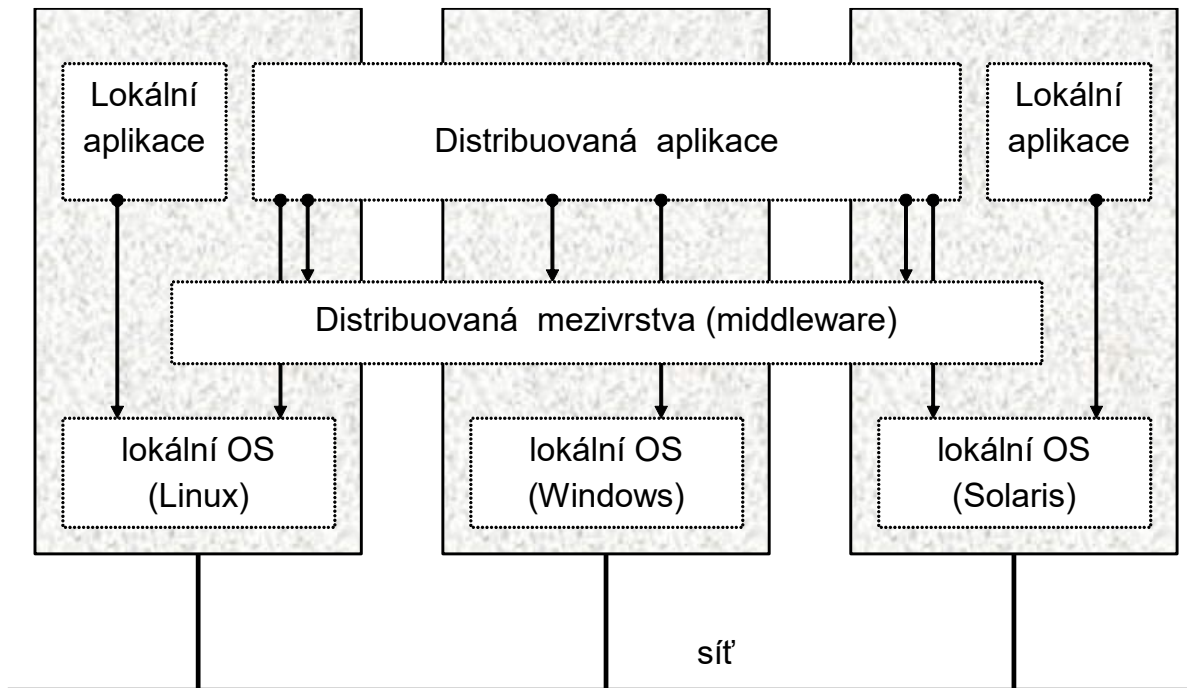
Nesplněná očekávání:
žádný systém
nedotažen do masově
použitelného stavu

*Python byl původně vyvinutý pro tento projekt!

200x - Distribuované frameworky a aplikace

Lokální OS (Win/Ux) + rozšiřující vrstva pro distribuované aplikace

Prostředí pro distribuované aplikace → **middleware** (OSF DCE, CORBA, DCOM, Globe, ..)



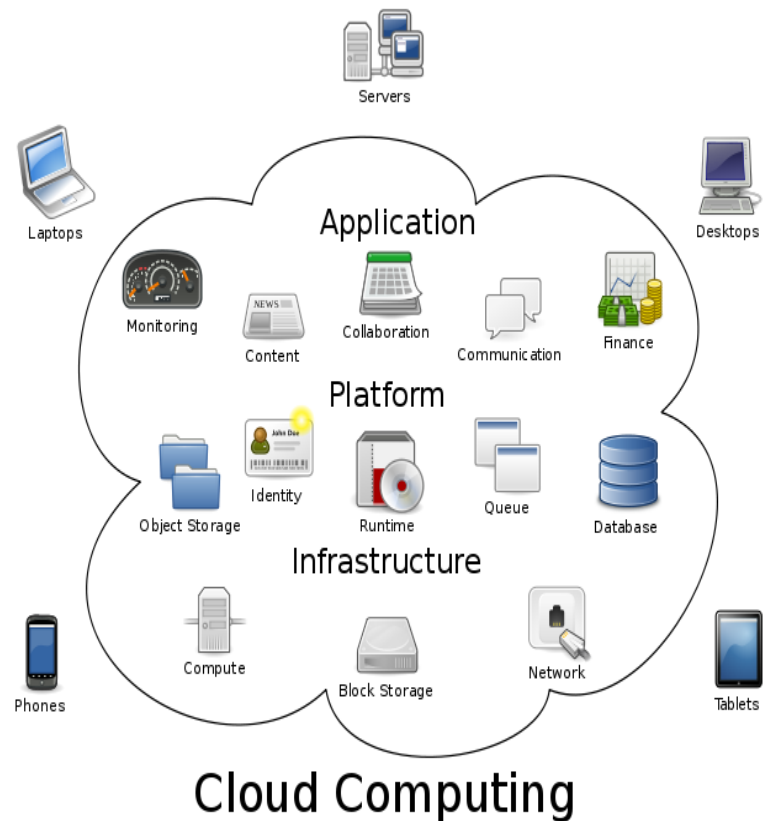
(Relativně) úspěšné
využití pro cluster /
grid computing

201x - Cloud computing, XaaS

Service-oriented computing: SaaS, PaaS, IaaS

- nutná podpora **virtualizace**
- kompletní infrastruktura připravená k použití:
Windows Azure, Google App Engine, Amazon EC2, OpenStack, ...

Mnoho různých
úspěšných aplikací



Aktuální trendy

Disribuovaná
AI

Edge
Computing

Serverless
Service mesh

Hlavní typy DS

DS pro vysoko
výkonnostní výpočty
(high-performance
computing)

Distribuované
informační systémy

DS pro pervazivní
výpočty/loT

DS pro spolehlivé a rozsáhlé (high-performance) výpočty

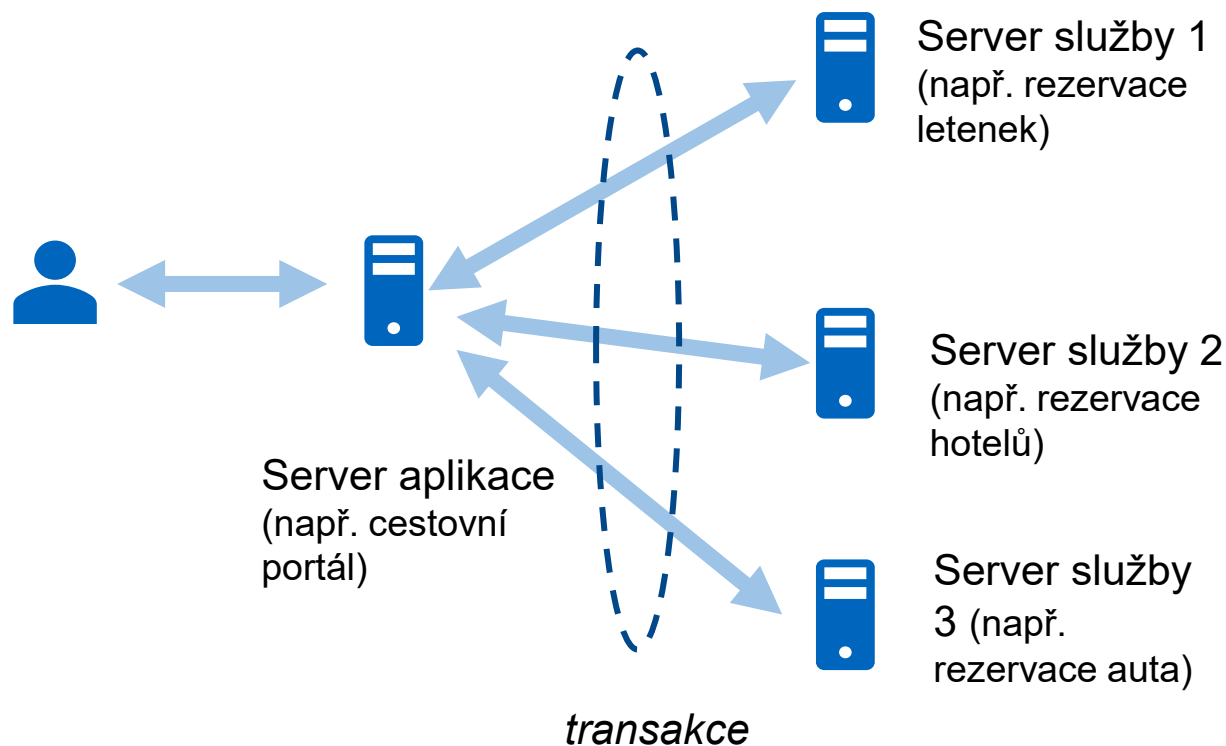
DS jako nástroj pro zvýšení výkonů, spolehlivosti a ekonomické efektivity.

Model sdílené paměti se na multi-počítačové architektury **nepodařilo rozšířit** → Další škálování možné jen v paradigmatu *distribuovaných* výpočtů založených na *posílání zpráv*.



Distribuované informační systémy

Distribuce vynucena příslušností jednotlivých výpočetních uzlů do **různých organizací**.

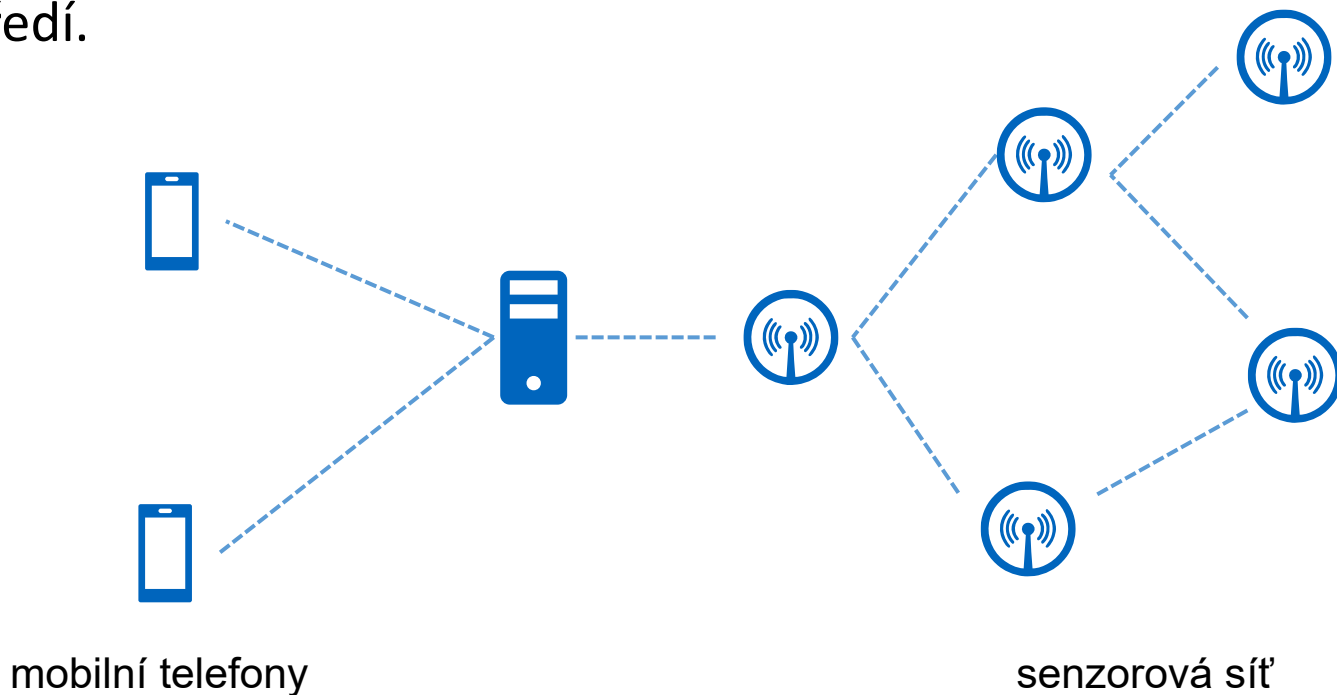


Hlavní výzva: **spolehlivost a konzistence.**

Pervazivní DS/IoT

Malé, mobilní výpočetní uzly, často **vtělené do fyzického prostředí**.
Omezení na dosah a rychlost komunikace, a na spotřebu energie.

Distribuce vynucena nutností **fyzické blízkosti** uzlu k uživateli nebo prostředí.



Hlavní výzva: **efektivita, dostupnost,**

Proč existují DS?

Distribuce jako nástroj

Distribuce může být cílenou **volbou v softwarovém návrhu** směřující k splnění specifických **požadavků**: odolnost vůči selhání, zvýšený výkon nebo poměr cena / výkon, nebo minimální QoS.

Např. replikované servery, cloud, ...



Vynucená distribuce

Aplikace vyžadující sdílení zdrojů nebo šíření informace mezi geograficky nebo organizačně **vzdálenými** entitami jsou „**přirozené**“ distribuované systémy.

Např. banka se několika pobočkami, senzorová síť, ...



Modelování DS

Klíčové elementy DS

Výpočet

Interakc
e

Čas

Selhání

Výpočet

Proces: jednotka výpočtu v distribuovaném systému – *počítá a komunikuje* (někdy nazýván uzel, hostitel, element, ...)

Množina (skupina) procesů: označována Π – je složena ze souboru N jednoznačně identifikovaných procesů p_1, p_2, \dots, p_N .

Klasické **předpoklady DS:**

- množina procesů je **konstantní** (N je dobře definováno)
- procesy se navzájem **znají**
- BÚNO: všechny procesy provádějí kopii stejného **algoritmu** (souhrn těchto kopií vytváří distribuovaný algoritmus)

V **reálných rozsáhlých** (někdy tzv. **extrémních**) distribuovaných systémech tyto předpoklady nemusí být splněny.

Interakce

Procesy komunikují **zasíláním zpráv**

- $send(m, p)$ pošle zprávu m procesu p
- $receive(m)$ přijme zprávu m

Zprávy mohou být v některých případech jednoznačně **identifikovány**

- odesílatelem zprávy
- sekvenčním číslem, které je lokální odesílateli

Klasickým předpoklad: každý pár procesů je propojen **obousměrným** komunikačním **kanálem** (point-point messaging)

- plně propojená topologie může být realizována pomocí směrování (routingu)

Čas

Přesné globální hodiny by umožnily globální uspořádání výpočetních kroků v DS. Bohužel **neexistují**.

Každý proces má své **lokální hodiny**.

Lokální hodiny **nemusí** ukazovat **přesný čas**.

Synchronizace lokálních hodin je možná **jen** s určitou **přesností**.

DS a Čas

V distribuovaných systémech je obtížné uvažovat o čase nejen kvůli absenci globálních hodin, ale také proto, že obecně nelze dát časové **limity** (horní meze) na **komunikaci** a **délku výpočtů**.

Různé možné modely:

- **Asynchronní DS**
- **Synchronní DS**
- **Částečně synchronní DS**

Synchronní vs. Asynchronní

Asynchronní systém

- Žádné časové limity na **relativní rychlost** vykonávání procesů.
- Žádné časové limity na **trvání přenosu** zpráv.
- Žádné časové limity na **časový drift** lokálních hodin.



Synchronní systém

- **Synchronní výpočty:** známe horní limit na relativní rychlost vykonávání procesů.
- **Synchronní komunikace:** známé horní limit na dobu přenosu zpráv.
- **Synchronní hodiny:** procesy mají lokální hodiny a je znám horní limit na rychlosti driftu lokálních hodin vzhledem k globálním hodinám.

Částečně synchronní DS

Částečná synchronicita: pro většinu systému je relativně snadné definovat časové limity, které platí **většinu času**. Občas se ale mohou vyskytnout období, během kterých tyto časové limity neplatí.

- zpoždění procesů: např. swappování, garbage collection
- zpoždění komunikace: přetížení sítě, ztráta zpráv (vyžadující jejich opakovaný přenos)

Prakticky užitečné systémy **jsou částečně synchronní**.

→ Umožňuje **v praxi vyřešit** problémy, které jsou za předpokladu plně asynchronních DS **neřešitelné**.

- některé z časových úseků synchronního běhu DS jsou dostatečně dlouhá na to, aby distribuovaný výpočet skončil

Selhání

Jak procesy, tak komunikační kanály mohou v DS selhat.

Selhání procesu

- **havárie (crash/fail-stop):** proces přestane vykonávat algoritmus (a reagovat na zprávy)
- **libovolné (byzantské) selhání:** proces může pracovat dále (a reagovat na zprávy), ale vykonává chybný algoritmus (z důvodu softwarové chyby nebo úmyslu)

Selhání kanálu

- **ztráta zprávy (message drop):** zpráva není doručena cílovému procesu (např. kvůli přetížení sítě nebo přetečení zásobníku v OS u přijímacího procesu)
- **rozdělení (partitioning):** procesy jsou rozdělené do disjunktních množin (oddílů - partitions) tak, že v rámci oddílu je komunikace možná, ale mezi oddíly nikoliv

V případě synchronních DS definujeme ještě **selhání časování**, pokud doba odezvy procesu nebo přenosu zprávy po síti vybočila z dohodnutého **časového rozmezí**.

Předpoklad na komunikační kanál

Dokonalý
(perfect)
kanál

Spolehlivé doručování: Pokud proces p pošle zprávu procesu q ani p a ani q nehavaruje, pak q nakonec zprávu obdrží.

Žádná **duplikace:** Žádná zpráva není doručena vícekrát než jednou.

Žádné **vytváření:** Je-li zpráva m doručena procesu p , tak zpráva m byla dříve poslána nějakým procesem q procesu p .

Garantované pořadí doručování: Odešle-li proces p procesu q zprávy m_1 a m_2 , tak pokud byla m_1 odeslána dříve než m_2 , tak m_2 nemůže být doručena aniž by předtím byla doručena m_1 .

Spolehlivý
kanál
FIFO kanál

Chybné předpoklady při vývoji DS

Fallacies of Distributed Computing

Řada DS je **zbytečně komplexních**. Komplexita je způsobena chybami, které je třeba později záplatovat. Chyby vycházejí z **mylných předpokladů**.

Typické mylné předpoklady

- Síť je spolehlivá
- Síť má nulovou/nízkou latenci
- Síť má konzistentní latenci
- Procesy odpovídají v časovém limitu
- Síť má neomezenou kapacitu
- ...

Shrnutí

Distribuované systémy jsou všude kolem nás a jejich **význam a složitost** dále **roste**.

Základním rozdílem mezi paralelními a distribuovanými výpočty jsou: absence **sdílené paměti**, absence **globálních hodin** a nezávislá **selhání**.