

Operační systémy

PŘEDNÁŠKA 12B. DISTRIBUOVANÉ SYSTÉMY

Distribuované systémy

„Distribuovaný operační systém běží na množině procesorů, které nesdílejí společnou paměť a přesto se navenek jeví jako jeden stroj.“

- **Distribuovaný systém:**

- Na uživatelské úrovni:

- ✦ Podpora distribuce úloh nad nedistribuovaným OS

- Na úrovni OS – distribuovaný operační systém:

- ✦ Distribuce mezi různé stroje implementována přímo v jádře OS

- Pro srovnání - síťový systém:

- Každý počítač má svůj OS
- Aplikace běží ve vrstvě nad lokálními OS
- Uživatelé vědí o existenci ostatních počítačů a síťových zdrojů

Výhody distribuce

- **Výpočetní výkon**
- **Vrozené sklony k distribuci**

Využijeme „distribuovanou povahu úlohy“: řízení výrobní linky, řízení provozu, banky apod.
- **Spolehlivost**

Výpadek jednoho uzlu systém zpomalí, ale nezastaví.
- **Rozšiřitelnost**

Lepší přidávat malé stroje, než kupovat a instalovat nový superstroj.
Přesněji naladíme výkon na okamžitou potřebu.
- **Škálovatelnost**

Lepší rozdělení výkonu, je možné určit vhodný procesor (dílčí stroj) pro konkrétní část úloh.

Rizika distribuce

- **Software:**

- V oboru obecně méně zkušeností s vývojem
- Méně vývojových nástrojů
- Omezenější volba OS a dalších prostředků
- S postupem doby se lepší a lepší

- **Hardware**

- Jádro má větší část rozhodování (rozděluje výkon mezi stroje, řídí více paměti apod.) – vyšší nároky na HW paměť a výkon procesoru než u jednotlivého stroje.

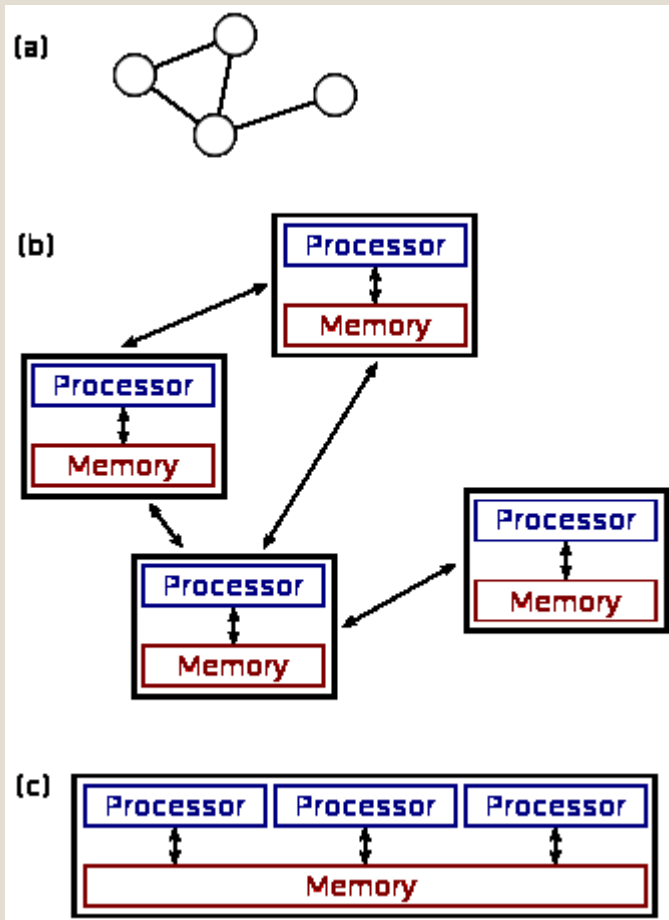
- **Propojení jednotek:**

- Nejčastěji „nějaká“ síť – musí být spolehlivé a s dostatečnou kapacitou.
- Riziko zvláště při geografické distribuci (pobočky banky).

- **Bezpečnost:**

- Dochází ke sdílení dat mezi jednotkami, možný útok či únik dat (zas ta banka...).

Distribuované vs. paralelní systémy



- Distribuované systémy mají oddělenou paměť (a,b).
- Paralelní systémy sdílejí paměť (c).
- Zdroj: wikipedia.org

Příklady použití

Distribuované systémy

Hodně nezávislých uzlů

- Řízení výrobní linky, letového provozu
- Směrování a řízení sítě
- Telefonní sítě
- Systémy virtuální reality, hry s mnoha hráči
- Distribuované databáze

Paralelní systémy

Spojení strojů = výkon

- Vědecké výpočty
- Cloud computing, grid computing
- Počítačová grafika (rendering)

Požadavky kladené na DS

- **„Dojem jednoho stroje“ (single system image):**
 - Dobře přidělit síťové prostředky.
 - Skrýt jejich fyzické umístění před uživateli.
 - Všechny prostředky dostupné všem procesům (místním i vzdáleným).
 - Prostředky se dají přemísťovat mezi stroji, aniž by to v systému bylo „vidět“.
- **Dobré propojení uzlů**
 - Identifikační systém pro jednotky a uživatele.
 - Nutná spolehlivá komunikace mezi uzly (zasílání zpráv aj.).

Požadavky kladené na DS II.

- **Samostatnost uzlů:**

Uzel je schopen pracovat a rozhodovat i při výpadku komunikace se zbytkem systému.

- **Schopnost vyvážit zátěž:**

Podle aktuální kapacity uzlů přesouváme úlohy sem a tam, navenek to ale nikdo nepozná.

- **Transakce** pro důležité operace (prevence problémů se sdílením zdrojů a synchronizací).

Rozšiřitelnost, spolehlivost

- V budoucnu by DS mohly mít i tisíce jednotek, zatím spíše stovky, vícejaderné procesory desítky.
- Základní pravidlo „velkých DS“:
 - Vyhnout se všemu centrálnímu.
 - Žádný ze strojů v systému nesmí být klíčový pro jeho chod, jinak ztrácíme výhody DS.
 - Příklad: iluze jednotných hodin – je vhodné nespoléhat na „přesně ve 12:00 GMT“

IPC – inter process communication

- V distribuovaných systémech nejčastěji zasílání zpráv.
- **Realizace:**
 - Klient/server (velká režie, ale robustní).
 - Request/reply: jednodušší než K/S, posílá se jen návratový kód.
 - RPC – remote procedure call – analogie volání lokální procedury - umožňuje použít „standardní“ programovací metody.
 - Skupinová komunikace (všichni se všemi, jeden se všemi) – příklad broadcast.

Synchronizace

- Semaforey a monitory vyžadují sdílenou paměť – nevhodné v DS.
- Problém s měřením času (nemáme společné hodiny).
- **Logické hodiny:**
 - Potřebujeme, aby se procesy shodly na pořadí událostí, nikoliv na absolutním čase.
 - Konzistentní měření času (událostí) v rámci DS.
 - Definujeme, které události si předcházejí a dbáme na jejich návaznost (matematický model).

Synchronizace fyzických hodin

- Tam, kde nestačí logické hodiny
 1. Vnější zdroj referenčního času
 - Může být n-násobný v silně distribuovaném systému (pozor na synchronizaci těchto zdrojů).
 2. Bez vnějšího zdroje, jen řízení
 - Každý stroj měří čas sám.
 - Centrální stroj se ptá ostatních, kolik je hodin a hlídá, aby některý stroj "neujel", dává pokyny k seřízení hodin.

Synchronizace zdrojů I.

- **Lamportův algoritmus**

1. Proces musí prostředek uvolnit před přidělením dalšímu procesu
 2. Požadavky jsou vyřizovány v pořadí, jak přicházejí
 3. Jestliže procesy odpoví v konečném čase, je algoritmus konečný
- Proces vysílá ostatním zprávu *request* s časovým razítkem.
 - Proces uvolňující zdroj píše *release* s časovým razítkem.
 - Ostatní procesy vždy odpovídají *reply* pro potvrzení, že zprávu mají.
 - Nevýhoda: velký počet zpráv nutných k chodu.

Synchronizace zdrojů II.

- **Ricart-Agarwala**

- Modifikace Lamperta, méně zpráv
- Procesy se rozhodují, komu dají přednost
 1. Proces vysílá *request*.
 2. Ostatní odpovídají buď bezprostředně *reply*, nebo *reply* posílají až po skončení svého požadavku.

- **Cyklické předávání pověření**

- Na množině procesů definujeme kruh, v němž si pověření předávají.
- Zjednodušení: pověření se předává i bez žádosti, prostě "chodí pešek okolo".

Deadlock v DS

- Prolomení podmínek: správou prostředku pověříme jeden proces, který prostředek ostatním přiděluje.
- **Lometovy algoritmy:**
 - Proces rozdělíme na kritické a nekritické části.
 - Před vstupem do kritické části hlásí proces své nároky.
 - Průběžně provádíme analýzu závislostí pro tyto předběžné nároky; nepřidělíme prostředky tam, kde uváznutí hrozí.
 - Distribuce:
 - ✦ Algoritmus distribuujeme do všech procesů.
 - ✦ Vyhodnocení tedy probíhá ve **všech** procesech současně.
 - ✦ **Druhý Lometův algoritmus:** nevyhodnocujeme celý graf, ale jen lokální grafy - nehrozí-li lokálně deadlock, nehrozí ani globálně.

Distribuovaná paměť

- Distribuované stránkování:
 - Stránkování lokálně/globálně; distribuované procesy mohou sdílet jednu stránku paměti
 - Možná bolestivá místa:
 - ✦ Replikace (lokální kopie pro zajištění přístupu pro všechny)
 - ✦ Konzistence a správa kopií – synchronizace replikovaných stránek
 - ✦ Nalezení správné stránky
 - Každá stránka má svého vlastníka, který řídí její změny

Distribuovaná data

- Replikované databáze.
- Distribuce objektů:
 - Komunikace a synchronizace s pomocí metod objektů
 - Podpora v jazycích CORBA, Java

Identifikace objektů v DS

- Sestává ze tří pilířů
 - **Identifikátor, id** – určení objektu.
 - **Adresa** – popis fyzického umístění.
 - **Cesta** – popis přístupu k objektu (adresa je výchozí bod k určení cesty).
- **Jmenný prostor**: množina možných jmen
 - Může být strukturováno, dynamické, samostatná věda.

Migrace mezi stroji

- Celé musí proběhnout bez vědomí ostatních (například nutné zachovat kontinuitu zasílaných zpráv apod.).
- Kromě plánování úloh máme navíc migrační strategie.
- **Postup:**
 1. Zmražení procesu (proces a jeho vlákna jsou ve stavu „migrován“ a jsou vyjmuty z front)
 2. Inicializace cílového počítače
 3. Kopírování procesu (PCB a dalších součástí)
 4. Obnovení komunikačních kanálů (nová adresa)
 5. Vymazání z původního místa
 6. Spuštění v novém místě

Správa prostředků

- Fyzické: paměti, procesory, tiskárny.
- Logické: procesy, soubory, data.
- Centralizovaná správa: jedna výchozí tabulka.
- Distribuovaná správa (kopie databází a jejich vzájemná replikace, synchronizace).