

# Operační systémy

## Úvod a definice pojmů.

Jan Trdlička



České vysoké učení technické v Praze,  
Fakulta informačních technologií  
Katedra počítačových systémů

<https://courses.fit.cvut.cz/BI-OSY>

## 1 Model výpočetního systému

## 2 Uživatelé

## 3 Hardware

- Procesor, paměť, sběrnice, periferní zařízení

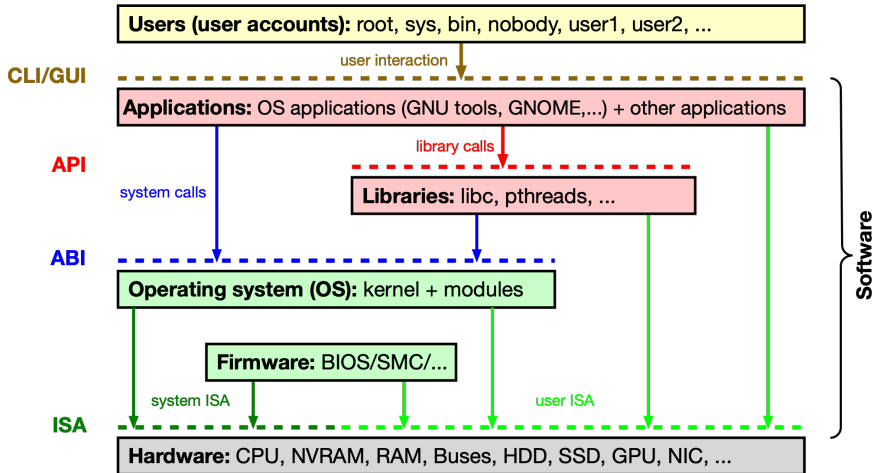
## 4 Operační systém

- Definice
- Jádro, systémová volání
- Vlastnosti moderních OS


## 5 Klasifikace výpočetních systémů

- Jednojádrový systém
- Vícejádrový systém se sdílenou pamětí (UMA, NUMA)
- Vícejádrový systém s distribuovanou pamětí (cluster)

# Model výpočetního systému



 Kernel mode

 User mode

**ISA** = Instruction Set Architecture  
(system ISA + user ISA)

**ABI** = Application Binary Interface

**API** = Application Program Interface

**CLI** = Command Line Interface

**GUI** = Graphic User Interface

- Uživatele můžeme rozdělit do několika kategorií, podle způsobu používání výpočetního systému.
  - ▶ **Uživatel**
    - ★ **běžný**: minimální znalosti OS, většinou přistupuje k systému přes GUI.
    - ★ **pokročilý**: hlubší znalosti OS, pro práci se systémem využívá jak GUI, tak i CLI, umí vytvářet např. skripty (dávky) a využívá např. API pro práci s některými aplikacemi.
  - ▶ **Správce**
    - ★ **systémový**: instaluje, konfiguruje, spravuje HW a OS
    - ★ **aplikační**: instaluje, konfiguruje a spravuje jednotlivé aplikace
  - ▶ **Vývojář**
    - ★ **systémový**: vyvíjí samotné jádro nebo jednotlivé drivery OS
    - ★ **aplikační**: vyvíjí jednotlivé aplikace

# Hardware: Processor

- Central processing unit (CPU)
- Implementuje konkrétní **Architekturu souboru instrukcí** (Instruction set architecture = ISA), která definuje
  - ▶ množinu instrukcí, adresní režimy, ...
  - ▶ množinu registrů, organizaci paměti, organizaci V/V, ...

**Příklad:** Systémy s různými CPU a ISA.

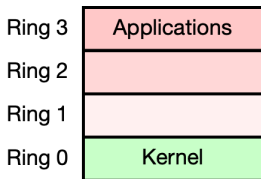
Systém	CPU	Výrobce	ISA	OS
A	AMD Ryzen 3 2200G	AMD	x86-64	Linux
B	Intel Core i5-8400	Intel	x86-64	Windows 10
C	Intel Xeon E3-1225 v6	Intel	x86-64	Linux
D	Ultra SPARC T2	Oracle	SPARC v9	Solaris

- ▶ CPU od různých výrobců (např. Intel a AMD) mohou implementovat stejnou ISA.
  - ★ Binární program ze systému A bude fungovat i na systému C (stejná ISA i OS).
  - ★ Binární program ze systému B nebude fungovat na ostatních systémech (různá ISA nebo OS) ⇒ možné zkompilovat, pokud kód nezávisí na ISA/OS.

# Hardware: Processor

- ISA také definuje různé **privilegované úrovně (módy)** běhu CPU
  - SW běžící v daném módu CPU může
    - ★ používat pouze instrukce povolené v tomto módu,
    - ★ modifikovat určité registry,
    - ★ modifikovat určité oblasti paměti, ...
  - Pro běh OS jsou důležité dva základní módy
    - ★ **Kernel mód**: vše je povoleno, typicky v něm běží jádro OS.
    - ★ **User mód**: omezený mód (nelze přímo manipulovat s periferními zařízeními, ...), typicky v něm běží uživatelské procesy.

**Příklad:** Privilegované módy v procesorech od Intelu



- ▶ **Ring 0**: vše je povoleno.
- ▶ **Ring 3**: nelze
  - ★ měnit aktuální ring,
  - ★ měnit tabulku stránek,
  - ★ registrovat interrupt handlers,
  - ★ provádět V/V instrukce,...

## ● Rozdělení procesorů podle typu instrukcí

### ▶ CISC (Complex Instruction Set Computer)

- ★ Relativně velký počet komplexnější instrukcí.
- ★ Obsahuje "memory-to-memory instrukce".
- ★ Instrukce jsou různě dlouhé a doba jejich zpracování se liší  
⇒ horší proudové zpracování instrukcí.
- ★ Větší nároky na hardware (architekturu CPU).
- ★ Program je obvykle reprezentován menším počtem instrukcí.
- ★ Příklad ISA: x86-64.

### ▶ RISC (Reduced Instruction Set Computer)

- ★ Relativně malý počet jednoduchých instrukcí.
- ★ Obsahuje hlavně "register-to-register" instrukce.
- ★ Instrukce jsou stejně dlouhé a s podobnou dobou zpracování  
⇒ lepší proudové zpracování instrukcí.
- ★ Větší nároky na SW (překladač).
- ★ Program je obvykle reprezentován větším počtem instrukcí.
- ★ Příklad ISA: SPARC v9, ARM

**Příklad:** Ukázka kódu pro násobení dvou čísel v paměti.

- Hodnoty dvou čísel jsou umístěné v paměti na adresách A1 a A2.
- Procesor obsahuje registry R1 a R2.

- CISC architektura

```
multm R1, A1, A2
store A1, R1
```

- ▶ Instrukce `multm` načte hodnoty z paměti do registrů a vynásobí obsahy registrů a výsledek uloží do registru.
- ▶ Instrukce `store` uloží obsah registru do paměti.

- RISC architektura

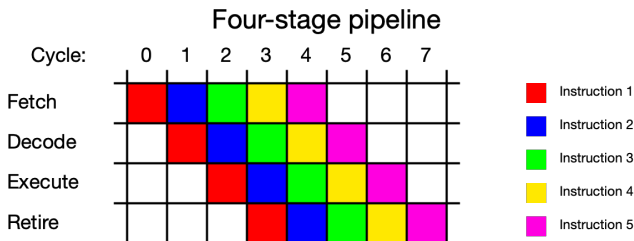
```
load R1, A1
load R2, A2
multr R1, R2
store A1, R1
```

- ▶ Instrukce `load` načte hodnotu z paměti do registru.
- ▶ Instrukce `multr` vynásobí obsah registrů, výsledek uloží do registru.



## ● Zpracování instrukcí

- ▶ Probíhá obvykle v několika fázích v konkrétních částech CPU
  - 1 **Fetch**: načtení instrukce z paměti do CPU.
  - 2 **Decode**: dekódování instrukce a načtení operandů do registrů.
  - 3 **Execute**: zpracování instrukce.
  - 4 **Retire**: uložení výsledku.
- ▶ Aby byly všechny části CPU maximálně vytížené používá se **proudové zpracování** instrukcí.



- ▶ Různé typy instrukcí trvají různě dlouho ⇒ **CPU může používat několik proudových jednotek (pipelines) pro různě složité instrukce.**

## ● Více instrukčních proudových jednotek (instruction pipelines)

### ▶ Load/Store pipe

- ★ Provádí instrukce pro načítání/ukládání dat z/do paměti.
- ★ Doba zpracování instrukce závisí na aktuálním umístění dat (skryté paměti, hlavní paměť, systém souborů).

### ▶ Integer pipe

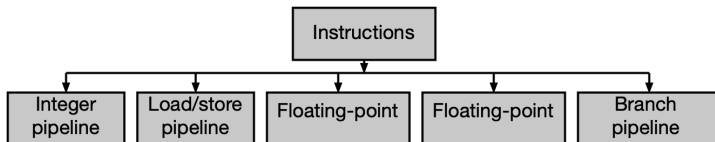
- ★ Zpracovává instrukce pro celočíselnou aritmetiku.

### ▶ Floating-point pipe

- ★ Složitější operace než s celými čísly  $\Rightarrow$  vyžadují více času.

### ▶ Branch pipe

- ★ Instrukce skoku způsobí načtení další instrukce z jiné oblasti paměti.
- ★ Dva způsoby: "branching" (větvení výpočtu, např. If/else) nebo "calling" (volání a návrat  $\Rightarrow$  nutné si zapamatovat info pro návrat).



## ● Rozdělení procesorů podle pořadí zpracování instrukcí

### ▶ in-order-execution

- ★ Instrukce jsou zpracovávány v pořadí definovaném překladačem.
- ★ Pokud instrukce používá výsledek z předchozí instrukce, který není ještě k dispozici, pak musí počkat na dokončení předchozí instrukce.
- ★ "Správné" pořadí instrukcí je důležité  $\Rightarrow$  kvalitní překladač.
- ★ Horší využití CPU.

### ▶ out-of-order-execution

- ★ Snaha minimalizovat čekání.
- ★ Místo čekání na dokončení předchozí instrukce se CPU pokusí začít zpracovávat některou z dalších instrukcí, která není závislá na dokončení předchozích instrukcí  $\Rightarrow$  lepší využití, ale složitější architektura CPU.
- ★ Příklad ISA: používáno v moderních CPU (x86-64, SPARC v9,...).

- **Rozdělení procesorů podle počtu jader**

- ▶ **Jedno jádrový (single-core)**

- ★ CPU obsahuje pouze jedno jádro.
    - ★ Toto jádro zpracovává pouze jeden instrukční proud (jedno vláko).

- ▶ **Více jádrový (multi-core)**

- ★ CPU obsahuje více jader.
    - ★ Každé jádro umí zpracovávat minimálně jeden instrukční proud.

- **Některé CPU umožňují zpracovávat "současně" více instrukčních proudů jedním jádrem.**

- ▶ **Hyper-threading**

- ★ Technologie používaná v některých procesorech od Intelu.
    - ★ Kritické části jádra jsou zduplikované, tak aby jádro umožňovalo zpracovávat dva instrukční proudy (dvě vlákna).

- ▶ **Multi-threading**

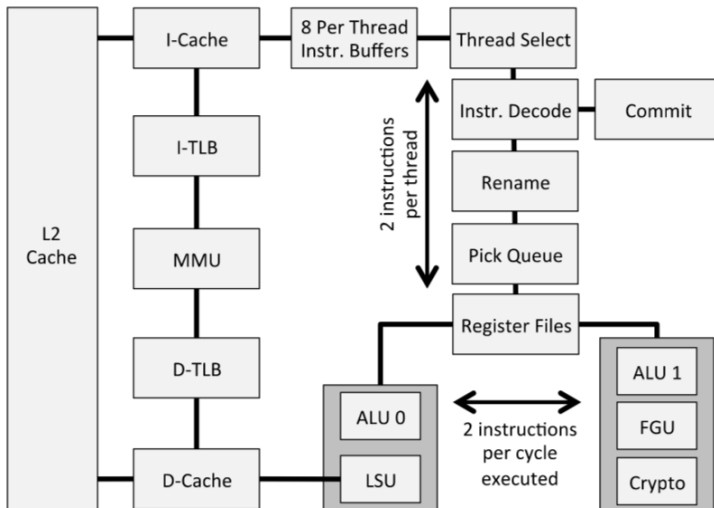
- ★ Technologie používaná u procesorů řady Ultra SPARC T.
    - ★ Části každého jádra jsou zdvojené.
    - ★ Využívá proudového zpracování 2x4 instrukčních proudů (8 vláken).

- ▶ Levnější varianta, jak zpracovávat více instrukčních proudů, aniž bychom museli přidat další jádra.

- ▶ Efektivita je závislá na zpracovávaných instrukčních proudech.

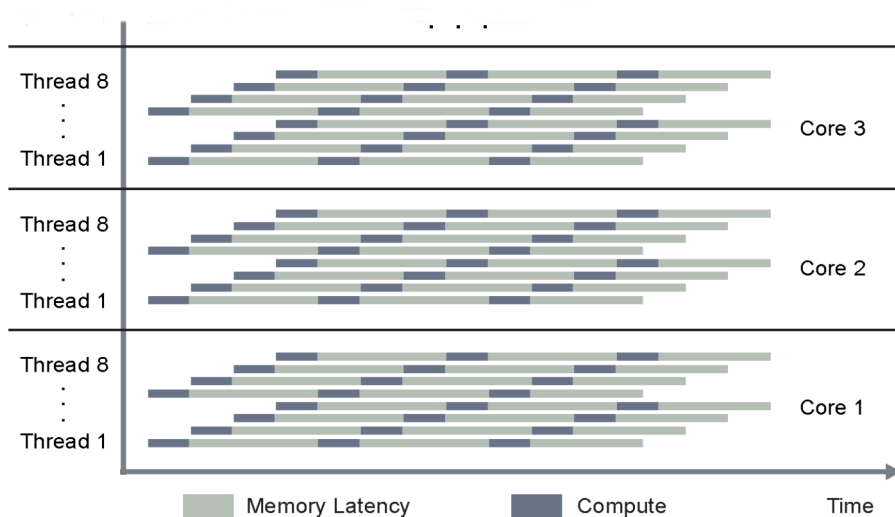
# Hardware: processor

## Příklad: Blokový diagram SPARC S3 core



# Hardware: processor

## Příklad: Procesor Ultra SPARC T4 – multi-threading



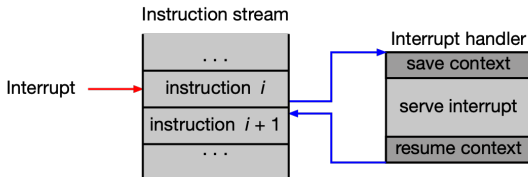
# Hardware: processor

## ● Přerušení (interrupts)

- ▶ Asynchronní reakce na nějakou událost.
- ▶ Normální zpracování instrukčního proudu jádra je přerušeno a začne se provádět obslužná rutina přerušení.
- ▶ Mechanismus přerušení je definován ISA.

## ● Kdy dochází k přerušení

- ▶ při zpracování instrukce
  - ★ přepnutí do jiného modu CPU, ...
  - ★ při chybě (např. dělení nulou, přetečení,...).
- ▶ když V/V zařízení žádá o pozornost
  - ★ generováno řadičem zařízení při dokončení operace,
  - ★ při chybě.



## Paměť

- **NVRAM (non-volatile random-access memory)/flash paměť**
  - ▶ Paměť s přímým přístupem, drží informaci i při odpojeném napájení.
  - ▶ Obsahuje
    - ★ **proměnné**: např. specifikace zařízení, ze kterého se má spustit OS,...
    - ★ **firmware**: detekce a konfigurace HW, spuštění OS (BIOS, Open Firmware,...).
- **RAM (random-access memory)**
  - ▶ Paměť s přímým přístupem, drží informaci pouze při napájení.
  - ▶ Reprezentuje hlavní paměť, ve které se nachází jádro OS a spuštěné aplikace.

## Sběrnice (bus)

- Zajišťuje přenos dat a řídicích povelů mezi jednotlivými částmi výpočetního systému (PCI-Express, Fibre Channel, SCSI, ...).



## Periferní zařízení

- Datová úložiště
  - ▶ Sekundární paměť pro zápis a čtení adresovatelných dat, drží informaci i při odpojeném napájení.
  - ▶ Reprezentována různými typy zařízení (HDD, SSD, RAID, ...).
- Síťové karty
- Grafické karty
- Monitory, klávesnice, myš, ...
- Zdroje, chlazení, ...

# Operační systém (OS)

## ● Definice

- ▶ Základní SW, který funguje jako prostředník mezi HW a aplikacemi/uživateli.

## ● Úkoly

- ▶ Správa a sdílení výpočetních prostředků
  - ★ fyzických (procesor, paměť, disky, ...)
  - ★ logických (uživatelská konta, procesy, soubory, přístup. práva,...)
- ▶ Poskytuje rozhraní (abstrakce složitosti HW)
  - ★ aplikacím (Win32 API, Win64 API, systémová volání Unixu,...)
  - ★ uživatelům (CLI a GUI)

## ● V tomto předmětu se zaměříme na principy univerzálních OS

- ▶ MS Windows, OS unixového typu (Linux, Solaris, MacOS, BSD,...), VMS, ...

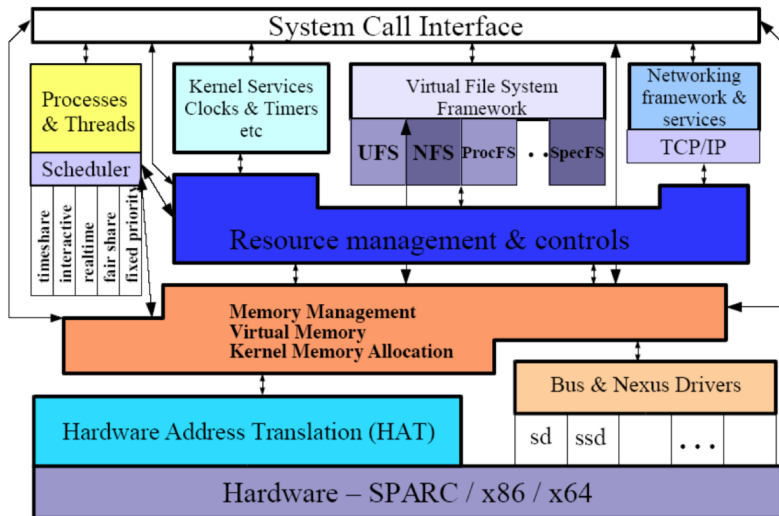
# Operační systém (OS)

## ● OS jako celek je reprezentován

- ▶ samotným **jádrem OS** (např. Linux kernel), které implementuje
  - 1 správu procesů a vláken (vytváření, plánování, ...),
  - 2 správu IPC prostředků (synchronizace, mezi procesová komunikace,...),
  - 3 správu paměti (alokace/dealokace, informace pro překlad adres,...),
  - 4 správu datových úložišť (RAID, HDD, SSD, USB flash, ...),
  - 5 správu systému souborů,
  - 6 správu V/V (přerušení, ovladače, ...)
  - 7 správu síťové infrastruktury (síťová rozhraní, IP stack, ...),...
- ▶ **aplikacemi**, které jsou součástí OS a které implementují
  - ★ CLI ( např. GNU nástroje, PowerShell,...),
  - ★ GUI (např. X11, GNOME, KDE, ...),
  - ★ správu SW (instalce aplikace,...),
  - ★ správu uživatelů, ...

## ● V tomto předmětu se zaměříme problematiku související s body 1-5.

- **Příklad:** architektura jádra Solarisu



# Systémová volání (syscalls)

## ● Aplikace

- ▶ typicky běží v user modu CPU
  - ⇒ **nemohou přímo používat prostředky systému,**
- ▶ pokud chce použít prostředek systému
  - ⇒ **musí požádat jádro OS pomocí systémového volání.**
    - ★ Konkrétní mechanismus systémového volání je závislý na ISA procesoru a použitém OS.

## ● **Příklad:** možný mechanismus systémového volání

- ▶ Systémové volání `write(int fd, void *buf, int N)`
  - ★ zapíše `N` bytů z paměti `buf` na souboru určeným deskriptorem `fd`.
- ▶ **Aplikace**
  - ★ uloží parametry systémového volání `fd`, `buf` a `N` (např. na zásobník, registrů,...),
  - ★ uloží informaci o požadovaném systémovém volání,
  - ★ vyvolá softwarové přerušení pomocí příslušné instrukce, která způsobí přepnutí CPU do kernel modu a spustí obslužnou rutinu v jádře OS.
- ▶ **Jádro**
  - ★ na základě požadovaného systémového volání provede příslušné operace (ověří přístupová práva, pokusí se zapsat `N` bytů,...).

# Systémová volání (syscalls)

- **Příklad:** Informace o systémových volání v různých OS
  - ▶ Linux: Manualové stránky: `man 2 intro` nebo
  - ▶ Solaris: [Oracle Docs: System Calls](#)
  - ▶ MS Windows: [Microsoft Docs: Windows API](#)
- **Příklad:** Sledování systémových volání volaných z aplikace
  - ▶ Linux

```
linux:~> strace date
execve("/usr/bin/date", ["date"], 0x7ffe540997c0 /* 88 vars */) = 0
...
write(1, "Fri Feb 15 09:14:46 CET 2019\n", 29) = 29
...
```

- ▶ Solaris

```
fray1:~> truss date
execve("/usr/bin/date", 0xFE01BB2C, 0xFE01BB34)  argc = 1
...
write(1, " F r i   F e b   1 5   0".., 29)          = 29
_exit(0)
```

# Vlastnosti moderních OS

- **Licencování OS**

- ▶ Různé podmínky pro používání, šíření, modifikaci a různá dostupnost zdrojového kódu OS (Open software, Libre software,...).

- **Víceúlohový (multitasking, time-sharing)**

- ▶ Běh více úloh (procesů) se sdílením času.
- ▶ Ochrana paměti, plánování procesů/vláken.

- **Vícevláknový (multithreading)**

- ▶ Proces se může skládat z několika současně běžících úloh (vláken)
- ▶ Přejít od plánování procesů na plánování vláken (thread).

- **Víceuživatelský (multi-user)**

- ▶ Možnost současné práce více uživatelů.
- ▶ Identifikace a vzájemná ochrana uživatelů.

- **Podpora multiprocesorových systémů (SMP)**

- ▶ Použití vláken v jádře a jejich plánování na různých jádrech CPU.

- **Unifikované prostředí**

- ▶ Přenositelnost mezi platformami (90% jádra v jazyce C).

# OS a různé typy výpočetních systémů

- **Vestavěné zařízení (embedded device)**
  - ▶ OS: Linux, proprietární OS,...
- **Chytré zařízení**
  - ▶ OS: Adroid, iOS, watchOS, tvOS,...
- **Laptop, stolní počítač**
  - ▶ OS: MS Windows, MacOS, Linux, ChromeOS, ...
- **Server**
  - ▶ OS: unixového OS (Linux, Solaris, HP-UX, AIX), MS Windows,...
  - ▶ Příklad konfigurace: [Oracle](#)
- **Super počítače**
  - ▶ OS: unixového OS (Linux, ...), proprietární OS, ...
  - ▶ Příklady konfigurací: [www.top500.org](http://www.top500.org)

- **Příklad:** Zastoupení OS na různých zařízeních

Mobile 2019-01 (netmarketshare.com)

Android	70,64 %
iOS	28,15 %
Others	1,03 %
Windows phone	0,06 %

Desktop 2019-01 (netmarketshare.com)

Windows	86,23 %
MacOS	10,59 %
Linux	2,4 %
Chrome OS	0,37 %
Others	0,41 %

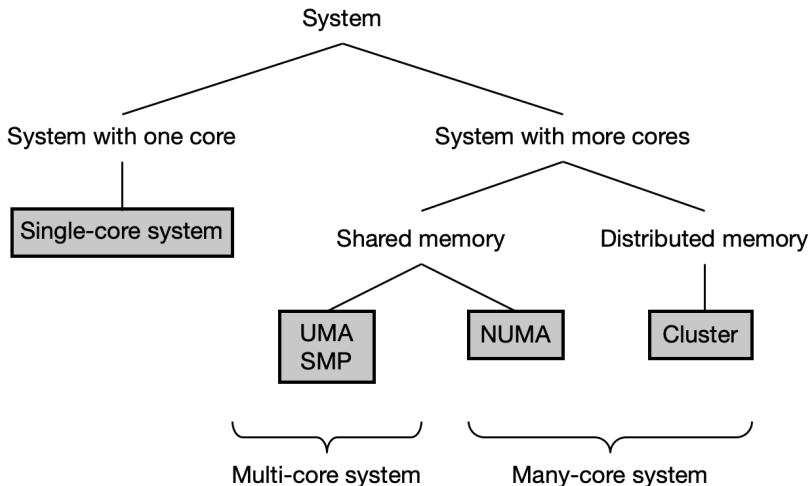
Websites 2019-01 (w3techs.com)

Linux	37,00 %
Windows	30,70 %
Others	32,3 %

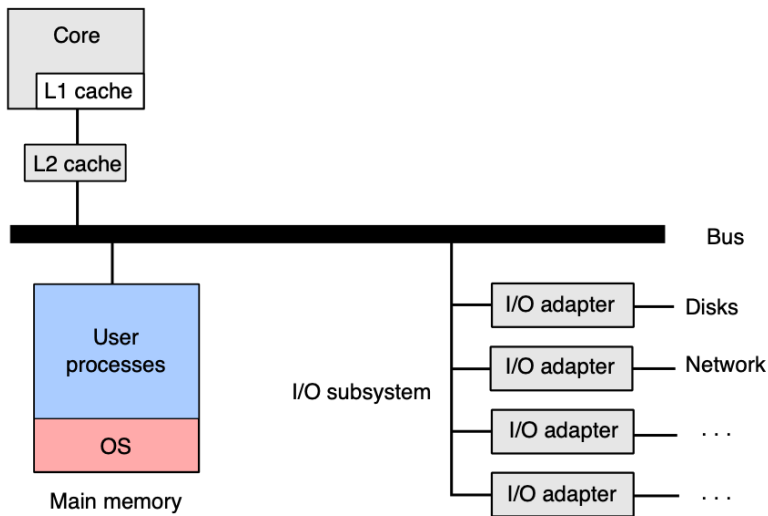


# Klasifikace výpočetních systémů

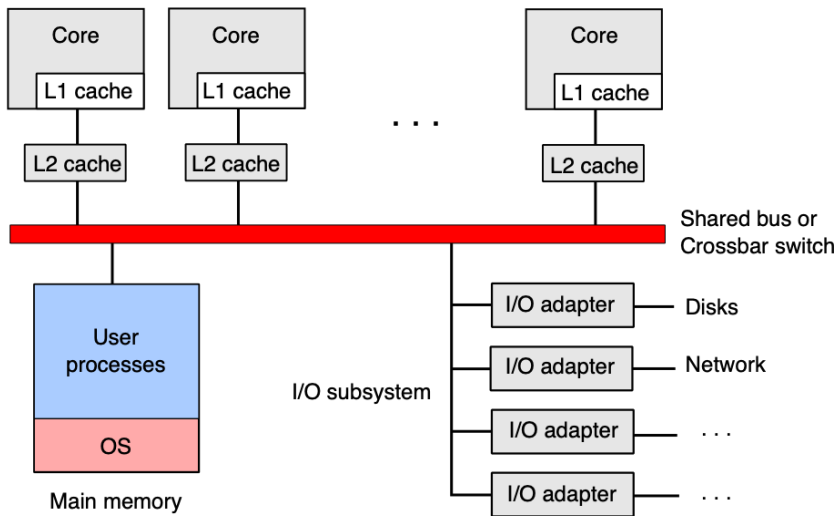
- Podle počtu jader a způsobu přístupu k paměti



# Jedno jádrový systém



# UMA - uniform memory access

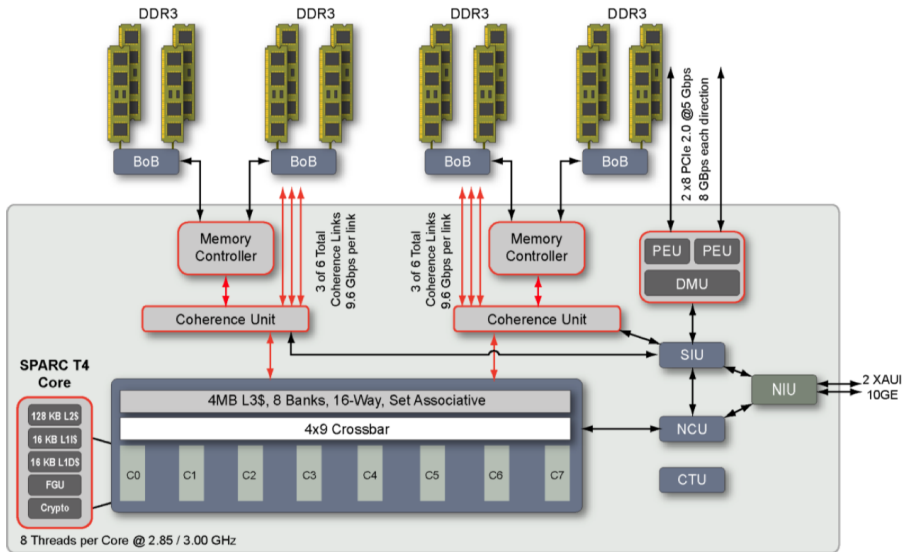


# UMA - uniform memory access

- Systém se skládá z několika identických jader.
- Jádra jsou propojena sdílenou sběrnicí nebo propojovací sítí.
- Jádra sdílí stejnou paměť (jeden nebo několik modulů).
- **Přístupový čas (latence) do paměti je pro všechny jádra téměř stejný.**
- Všechny jádra sdílí I/O.
- Tato HW architektura je také označována jako **SMP (symmetric multiprocessors)**.
- Sdílená sběrnice je slabé místo, proto může být nahrazena propojovací sítí (crossbar switch).

# UMA - uniform memory access

## Příklad: Procesor Ultra SPARC T4



# UMA - uniform memory access

## Příklad:

- Informace o serveru `fray1.fit.cvut`

- ▶ ISA

```
fray1:~> uname -a  
SunOS fray1 5.11 11.3 sun4v sparcsunw,SUNW,SPARC-Enterprise-T5120
```

- ▶ CPU

```
fray1:~> psrinfo -pv  
The physical processor has 4 cores and 32 virtual processors (0-23,32-39)  
  The core has 8 virtual processors (0-7)  
  The core has 8 virtual processors (8-15)  
  The core has 8 virtual processors (16-23)  
  The core has 8 virtual processors (32-39)  
    UltraSPARC-T2 (chipid 0, clock 1165 MHz)
```

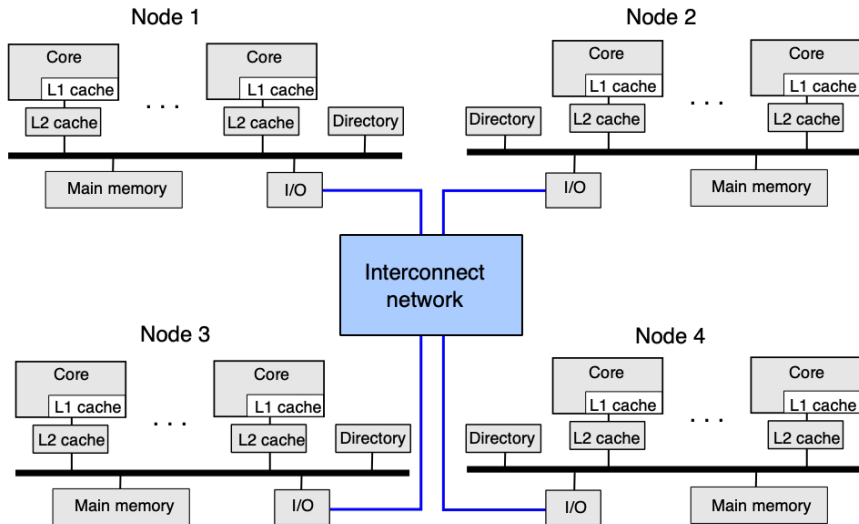
- ▶ OS

```
fray1:~> cat /etc/release  
  
Oracle Solaris 11.3 SPARC  
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.  
Assembled 06 October 2015
```

- Informace o serveru s OS Linux

- ▶ Pomocí příkazů: `uname -a`, `lscpu`, `cat /etc/os-release`.

# NUMA - nonuniform memory access



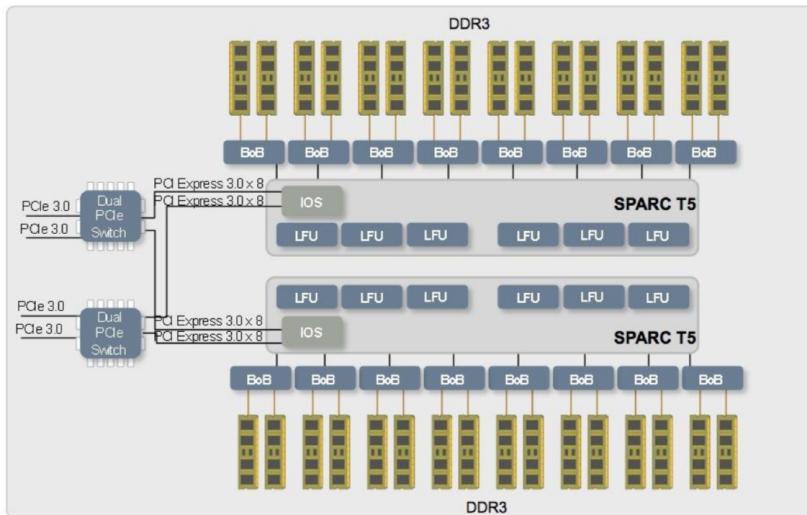
# NUMA - nonuniform memory access

- Lokální paměť uzlu je viditelná pro jádra z ostatních uzlů.
- Přístup do vzdálené paměti je pomalejší než do lokální paměti.
- Koherence skrytých pamětí je zajištěna hardwarem, proto je tato architektura někdy označována jako cache-coherent NUMA.
- OS má znalost o paměťových latencích mezi jednotlivými jádry a pamětmi, a může ji využívat při plánování vláken.



# NUMA - nonuniform memory access

- Příklad: Dual-socket SPARC T5 konfigurace



# NUMA - nonuniform memory access

## Příklad:

- Informace o serveru `fray2.fit.cvut`

- ▶ CPU

```
fray2:~> psrinfo -vp
The physical processor has 8 cores and 64 virtual processors (0-63)
  The core has 8 virtual processors (0-7)
  The core has 8 virtual processors (8-15)
  The core has 8 virtual processors (16-23)
  The core has 8 virtual processors (24-31)
  The core has 8 virtual processors (32-39)
  The core has 8 virtual processors (40-47)
  The core has 8 virtual processors (48-55)
  The core has 8 virtual processors (56-63)
    UltraSPARC-T2+ (chipid 0, clock 1414 MHz)
The physical processor has 8 cores and 64 virtual processors (64-127)
  The core has 8 virtual processors (64-71)
  The core has 8 virtual processors (72-79)
  The core has 8 virtual processors (80-87)
  The core has 8 virtual processors (88-95)
  The core has 8 virtual processors (96-103)
  The core has 8 virtual processors (104-111)
  The core has 8 virtual processors (112-119)
  The core has 8 virtual processors (120-127)
    UltraSPARC-T2+ (chipid 1, clock 1414 MHz)
```

# NUMA - nonuniform memory access

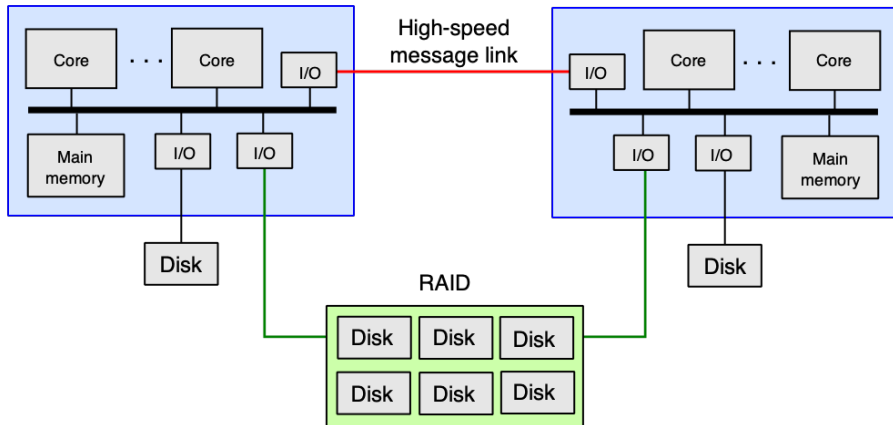
- Locality group (lgroup): info o paměťových latencích mezi uzly

```
fray2:~> lgrpinfo
lgroup 0 (root):
    Children: 1 2
    CPUs: 0-127
    Memory: installed 32G, allocated 11G, free 20G
    Lgroup resources: 1 2 (CPU); 1 2 (memory)
    Latency: 18
lgroup 1 (leaf):
    Children: none, Parent: 0
    CPUs: 0-63
    Memory: installed 16G, allocated 4.3G, free 11G
    Lgroup resources: 1 (CPU); 1 (memory)
    Load: 0.0188
    Latency: 12
lgroup 2 (leaf):
    Children: none, Parent: 0
    CPUs: 64-127
    Memory: installed 16G, allocated 7.2G, free 8.8G
    Lgroup resources: 2 (CPU); 2 (memory)
    Load: 0.0118
    Latency: 12
```

- Informace o procesech jejich přiřazení do lgroup

```
fray2:~> ps -H
  PID LGRP TTY          TIME CMD
25351   2 pts/27      0:00 ps
25008   2 pts/27      0:00 bash
```

# Cluster



## ● Základní uzel

- ▶ CPU, lokální paměť, síťové rozhraní, disk,...
- ▶ Ostatní periferie mohou chybět.
- ▶ OS + SW podporující některé funkce clusteru (vyvažování výkonu,...).

## ● Uzly jsou propojeny vysokorychlostní propojovací sítí.

## ● Použití

- ▶ Výpočetní systémy.
- ▶ Fault-tolerant systémy.
- ▶ Systémy s vyvažováním zátěže.

## ● Typy podle konfigurace

- ▶ Homogenní
  - ★ Uzlu mají stejnou/podobnou hardwarovou konfiguraci.
  - ★ Na všech uzlech je stejný OS.
- ▶ Heterogenní
  - ★ Uzly mají různé hardwarové konfigurace.
  - ★ Na jednotlivých uzlech se používají různé OS.

- ❶ A. S. Tanenbaum, H. Bos: *Modern Operating Systems (4th edition)*, Pearson, 2014.
- ❷ W. Stallings: *Operating Systems: Internals and Design Principles (9th edition)*, Pearson, 2017.
- ❸ A. Silberschatz, P. B. Galvin, G. Gagne: *Operating System Concepts (9th edition)*, Wiley, 2012.
- ❹ R. Buyya: *Mastering Cloud Computing: Foundations and Applications Programming (1st Edition)*, Morgan Kaufmann, 2013.
- ❺ D. Gove: *Solaris Application Programming (1st Edition)*, Prentice Hall, 2008.
- ❻ Oracle's SPARC T4-1, SPARC T4-2, SPARC T4-4, and SPARC T4-1B Server Architecture, Oracle White Paper, 2012.
- ❼ READ\_ME\_FIRST: *What Do I Do with All of Those SPARC Threads?*, Oracle Technical White Paper, 2013.