Teljesítmény-optimalizált szoftverek

Dr. Bakay Árpád – Ericcson

(Dr. Szendrei Rudolf – ELTE)

Miért indult el ez a tantárgy?

- A szoftverek „használati értéke” közel sincs arányban az impresszív módon növekvő hardverekkel
  - CPU, Memória, adat tárhely, hálózati forgalom, megbízhatóság/hibatűrőség
- Különösen nagyvállalati fejlesztésben elszaporodtak az eszetlen szoftver „konglomerátumok”
  - Ki mihez ért, abban/azzal fejleszt
    - Ha épp működik, akkor kész…
  - Többszáz készen letölthető library/keretrendszer használata, sokszor kevés ismerettel, haszonnal
  - Indokolatlan biztonsági, védelmi megoldások
    - És mégis hibák, sérülékenységek, összeomlások
- A nagy méretű szoftverek saját maguknak termelik a problémákat
    - Sok szekrény, nagy géptermek, erős tápellátás és hűtési igény
    - Pl. mire el kell adni, már csak nagy/gazdag ügyfelek jönnek szóba

A szoftvertechnológiai ‚mesterlövész'

- Türelmesen végére jár egy-egy problémának
- A legintenzívebben dolgozó modulokra/algoritmusokra fókuszál
- Meg tudja beccsülni, hogy hol, mi a reális erőforrás-igény
  - És mi a teendő, ha nem ezzel szembesül
- Értékeli az egyszerűséget, tisztaságot, akár a maximális funkcionalitással / biztonsággal szemben is.

Miért jobb az optimalizált szoftver?

- Jobb felhasználói élmény
- Kisebb, egyszerűbb, kevesebb gépen elfut
  - Kisebb beruházás
  - Kisebb áram-fogyasztás (és klimatizálás)
  - Kisebb géptermi hely-igény
- Kevesebb komponens, kevesebb meghibásodás
  - Átláthatóbb
  - „Fully fault tolerant” rendszerek megvalósítása sosem triviális
- Sok esetben átgondoltabb szoftver-struktúra
- stb.

A „szoftver jóság” egyéb tényezői

- Funkcionalitás
- Könnyű kezelhetőség
- Biztonság
- Karbantarthatóság
- Gyorsan kifejleszthető
- Kevésbé képzett, azaz olcsóbb fejlesztői munkaerő

... ezekkel konkurrál a performancia

- de nem mindig egymás ellen: gyakran **kooperál** is

Plusz: „kétes értékű” szempontok az optimalizálás ELLEN

- **Big hardver** = big money = big influence

- **CV-ware** = bizonyos keretrendszerek ismerete előny az álláskeresésnél.
  - *„I have developed a Kafka based distributed system in the last 4 years”*
  - *v.ö.*
    - *„I have developed a low footprint, optimized message passing middleware in C++”*

Most hadd halljak Önökről....

- Hány évet járt eddig, ill. mikor tervezi megszerezni a diplomáját?

- Milyen nyelveken mit fejlesztett eddig?
  - Egyetemi feladatot
  - Külső projekteket
- Mi ragadta meg Önt leginkább az eddigi tantárgyak/témák közül?
  - Mely tárgy(ak)on foglalkoztak optimalizálással?

- C, C++, Java, Python, Javascript: volt-e ezekre alapozó tantárgya, feladata?
  - File műveletek, hálózati kommunikáció, serialization?
- Linux/bash/scriptek? (+ grep, sed, cut, tr, sort, find, vi stb.)
  - Linux kernel, modulok, driverek?
- Adatbázis tantárgy volt-e? (pl. SQL CREATE, INSERT, SELECT)
- Egyéb speciális nyelvek

- Are you happy with English slides (from lesson #2)?

Magamról

- 40 éve a pályán
- Témák
  - Ipari képfeldolgozás 1987-1995
  - Beágyazott hálózati eszközök fejlesztése 1994-1997
  - Model alapú szoftver-definíciós szoftverek fejlesztése 1999-2002
  - Távközlési mérőeszközök és mérőrendszerek 2006-2019
  - Mobilszolgáltatás analítikai szoftver (Ericsson) 2019-
- Kb. 8-10 programnyelvű fejlesztésen dolgoztam, fejlesztőként/lektorként/managerként.
- „Specialitásom” a hálózati technológiák és a Linux kernel közeli ismerete
- Közel 20 éve oktatok
- Közel áll hozzám a gyakorlatias, igényes, optimalizált szoftverfejlesztés

Témák dióhéjban – „Bottom-up” stratégia

- CPU teljesítmény
  - CPU Core: egységek, utasítás-feldolgozás, alapműveletek időigénye
  - Cache-ek teljesítménye és optimalizált használata, koherencia problémák
  - Memória: technológiák, virtuális memória
  - Multi-core, multi CPU hatások, optimalizálások
  - Processzek, thread-ek, ütemezés, Inter-Process Communication
- Szoftver nyelvek teljesítményének összehasonlítása
  - Konkrét problémáknál
  - Pl. C, Java, Python, Javascript
- Hálózati kommunikáció és diszk IO, sebessége költségei, korlátai
  - Transzport protokollok és titkosítás hatása a performanciára
  - Diszk típusok, teljesítményük
  - Throughput és latency
- Alkalmazások, hálózati szerverek, adatbáziskezelés, message bus middleware-k teljesítménye
  - Tudunk-e jobbat, egyszerű feladatokra?
  - OS-Level accelerations: kernel modules, DPDK
- Egyéni feladatok választása, megtervezése, megvalósítása, bemutatása

Nyelvek

- Példák nagy része C
- Lesznek Java, Python és Javascript példák is.
  - Meg amit Önök hoznak...
- Kicsit belenézünk az Assembly kódba is.

Miről nem lesz szó?

- GPU programozás – reméljük a következő félévben, már igen
- C# - nem nagyon értek hozzá
- RISC és korábbi, 32bites CPU-k – csak X64

Számonkérés

- Osztályzat az egyéni feladat alapján
  - Feladatválasztás (az 5.-7. héten esedékes)
  - Megvalósítás mélysége és minősége
  - A beadott feladat és a dokumentáció / mérési jegyzőkönyv
  - Prezentáció tartalma és stílusa
  - Osztályzás több fázisban: 1. „vizsgálati fázis”, „code complete” állapot ill. 2. végső beadott anyag ill. 3. bemutató
- Az elkészítéshez számoljanak 20+ produktív órával!

- A rendszeres, figyelmes és aktív részvétel „tudat alatt” befolyásolja az értékelést
  - Lesz katalógus; hiányzások kezelése az általános irányelvek szerint
  - Indokolt esetben rugalmasak leszünk

- Diplomamunka-lehetőség, akár MSc szinten is
- Álláslehetőségek az ELTÉ-n és másutt

Mire számítsunk a kurzus végén?

- ‚A handful of young engineers', aki pályája kezdtetétől „érzi és értékeli" a szoftver a performanciát

- Nincsenek általános, minden szoftver számára érvényes igazságok
  - De közelről megvizsgáltunk bizonyos dolgokat
  - Jobban ráéreznek a performancia-problémákra
  - Módszertanilag helyes „benchmarking"

- Az előadó messze nem tud mindent
  - Önök dolga, hogy felfedezzék/megértsék a dolgokat.
  - Olyan utakon járunk, ahol nagy emberek jártak előttünk

- Fejlesszük együtt ezt a tárgyat!
  - Konkrét hibák jelzésével, követhetőség javításával, téma-ötletekkel, OHV-vel

...Csapjunk bele!

- C programok
- Gitlab Snippetek itt: **https://gitlab.inf.elte.hu/-/snippets/**

Példa 1. Iteráció Mátrixban

- https://gitlab.inf.elte.hu/-/snippets/27
- A memory elméletileg „Random Access”
- Mégis: nem mindegy, hogyan iterálunk!
- Mért nem?

Példa 1. Iteráció Mátrixban

- https://gitlab.inf.elte.hu/-/snippets/27
- A memory elméletileg „Random Access”
- Mégis: nem mindegy, hogyan iterálunk!
- Mért nem?

- A CPU Cache három rétege
  - L1: 2x32 kbyte/Core, access time: 4 CPU ciklus -> kb 1G olvasás/sec
  - L2: 256 kbyte/Core, access time: 12 CPU ciklus
  - L3: 4-8 Mbyte/CPU: Access time: 30 CPU ciklus
- V.ö: DRAM: Access time: 80 ciklus
- Ráadásul:
  - 1 cache-line - 64 byte
  - DRAM Read-ahead

Példa 2: Egy feladat – 3 nyelv

- https://gitlab.inf.elte.hu/-/snippets/26
- Python – lassú, de van rá megoldás
- Java – OK
- C – Nem mindig veri a Java-t de optimaizálva már igen
    - Kulcs a vektoros feldolgozás - Singe Instruction Multiple Data (SIMD)

===

- Egyéb „apróságok”
    - Miért nem használtunk IDE-t?
    - A Unix alatti C programozás szépségei: man, gcc
    - Tool-ok „barkácsolása”

Példa 3: Multi-threaded program

- https://gitlab.inf.elte.hu/-/snippets/28
- Inspired by Scott Meyers: https://www.youtube.com/watch?v=WDIkqP4JbkE
- Tanulság: nagyon oda kell figyelni a multi-threaded programoknál
  - Közös erőforrások konkurrens elérése, „hazard” helyzetek
  - Teljesítmény problémák

Mára ennyi!