ICS导引

游震邦

自我介绍

- 游震邦
- 北大信科2018级本科生
- 2020/2021助教

大纲

- Warming up
- 性能优化（计算密集型）
- 性能优化（IO密集型）
- 代码正确性与编程语言

问题0.1: 你真的没接触过计算机系统吗?

- 第一门系统课 ≠ 第一门涉及系统的课
- 当你:
  - 考虑到浮点数运算并非精确
    - a == b -> **abs(a - b) < 1e-7** => Wrong Answer -> Accepted
      - 浮点数编码问题
  - 以渐近复杂度之外的手段优化代码
    - **复杂度最优**，OpenJudge：**1001ms，Time Limit Exceeded**
    - 性能是系统里最根本的问题之一
  - 思考为什么选课网总是崩溃
    - 网页服务器吞吐量与延迟

你其实都在接触计算机系统!

计算机系统在软件中无处不在!

问题0.2: 为什么要学计算机系统?

- 经典回答：让我们成为更好的程序员
- 我的回答：好奇
  - 想知道计算机底层怎么工作
  - 想压榨手里的机器的性能

越往后，好奇心占比越大

问题0.3: CSAPP的题眼是什么

- Computer Systems: A **Programmer's** Perspective
- ICS教会我们用系统，而非造系统

问题1: 性能优化 之 矩阵乘法 (计算密集型)

- 基本概念
  - 计算密集型：程序的多数时间用于计算（CPU、内存等）
  - IO密集型：I/O即Input/Output，意味着程序的多数时间用于输入输出（网络，磁盘等）
- 问题背景
  - 矩阵乘法是深度学习中最常见的计算
  - 给定输入矩阵A, B，计算C=A*B，其中C(i;j) = Σ_k A(i;k)*B(k;j)

```
for (int i = 0; i < DIM; i++) {
  for (int j = 0; j < DIM; j++) {
    for (int k = 0; k < DIM; k++) {
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

后续优化不改变渐近复杂度！

问题1: 性能优化 之 矩阵乘法 (计算密集型)

- 原始方案:

```
for (int i = 0; i < DIM; i++) {
  for (int j = 0; j < DIM; j++) {
    for (int k = 0; k < DIM; k++) {
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

- 编译并执行 (DIM = 512, float): g++ -o mm mm.cpp && ./mm
- 时间: 460ms

问题1: 性能优化 之 矩阵乘法 (计算密集型)

- 优化1: 提升编译优化等级（O3优化）
  - g++ -o mm mm.cpp **-O3** && ./mm
  - **460ms -> 146ms**
  - 编译器做了什么?
    - 第3章：汇编语言
    - 第4章：CPU体系结构
    - 第5章：性能优化
  - ~~怎么写一个这样的编译器~~
    - ~~编译原理~~

Free Lunch!

问题1: 性能优化 之 矩阵乘法 (计算密集型)

- 优化2: 交换循环顺序 (ijk -> ikj)

```
for (int i = 0; i < DIM; i++) {
  for (int j = 0; j < DIM; j++) {
    for (int k = 0; k < DIM; k++) {
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

```
for (int i = 0; i < DIM; i++) {
  for (int k = 0; k < DIM; k++) {
    for (int j = 0; j < DIM; j++) {
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

- **460ms -> 146ms -> 12ms**
- 第6章：内存局部性

问题1: 性能优化 之 矩阵乘法 (计算密集型)

- 优化3: 多线程

```
for (int i = 0; i < DIM; i++) {
  for (int j = 0; j < DIM; j++) {
    for (int k = 0; k < DIM; k++) {
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

```
#pragma omp parallel for
for (int i = 0; i < DIM; i++) {
  for (int k = 0; k < DIM; k++) {
    for (int j = 0; j < DIM; j++) {
      c[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

- **460ms -> 146ms -> 12ms -> 3ms**
- 第12章：多线程

问题1: 性能优化 之 矩阵乘法 (计算密集型)

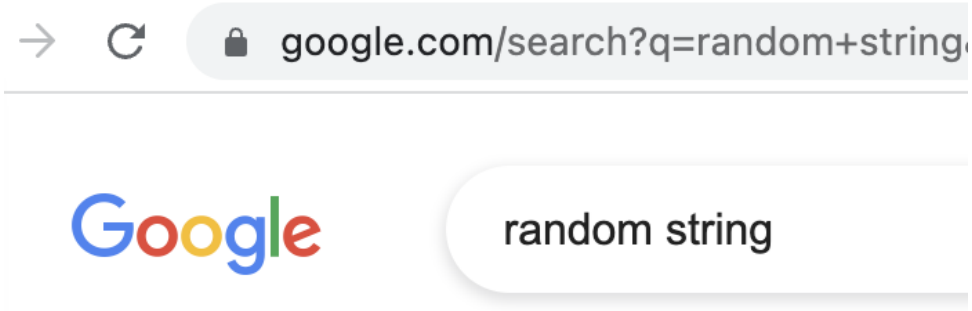
- 后续问题
  - CPU上还有哪些优化空间
    - 分块
  - GPU？TPU？
  - 如果矩阵非常大，但是你有大量机器，怎么办？
    - LLM（大语言模型）常见场景
    - 分布式计算
    - **LLM的计算平台和数据平台都远未成熟！**
  - 多线程对矩阵乘法非常有效，但是对于向量加法并非如此，为什么？

问题2: 性能优化 之 获取大量网页 (IO密集型)

- 背景：压力测试
  - ~~测试选课网吞吐量~~
  - 测试我部署的服务的吞吐量

问题2: 性能优化 之 获取大量网页 (IO密集型)

- 浏览器搜索的实质



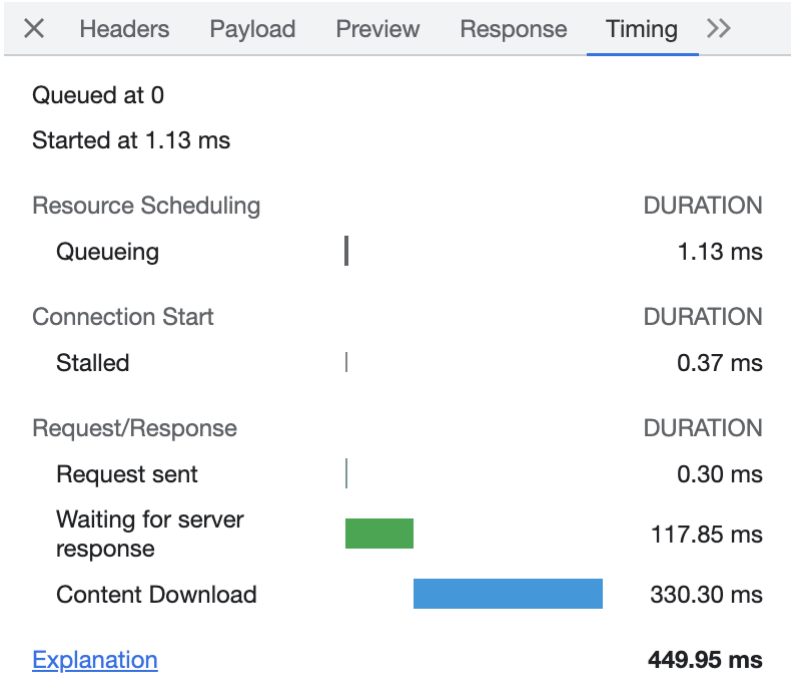
HTTP
GET
Request

```python
import requests

response = requests.get("https://www.google.com/search?q=random+string")

with open("random.html", "wb") as file:
    file.write(response.content)
```

这个HTML文件可以被浏览器打开!

问题2: 性能优化 之 获取大量网页 (IO密集型)

- 性能分析
  - 绝大多数时间消耗在IO
  - CPU利用率很低



问题2: 性能优化 之 获取大量网页 (IO密集型)

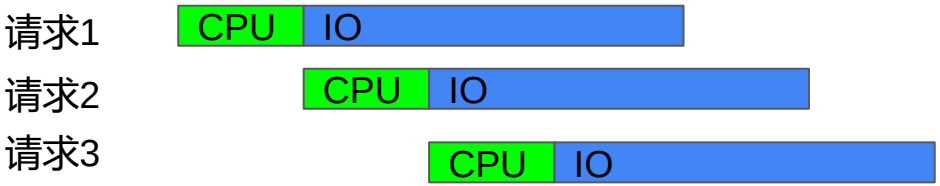
- 如果获取大量网页，是否可以让IO时间重合?
  - 无需等待上一个请求结束，就可以开始下一个请求
- 性能比较（100个网页，Python实现）
  - 原始方案：71.4s
  - 多线程：5.6s
  - 异步编程：1.1s

CPU在不同请求之间有切换开销!

原始方案

请求1　　请求2　　请求3

改进方案



理想情况!
切换开销决定了性能（第12章）
近年迅速发展的领域

问题2: 性能优化 之 获取大量网页 (IO密集型)

- 性能比较（100个网页，Python实现）
  - 原始方案：71.4s
  - 多线程：5.6s
  - 异步编程：1.1s
- CSAPP
  - 第8、10章：IO相关
  - 第11章：网络编程
  - 第12章：多线程与异步编程

问题3: 编程语言

- CSAPP
  - Computer Systems: A Programmer’s Perspective
  - Computer Systems: A **C** Programmer’s Perspective
- CSAPP里的很多编程陷阱，仅限于C语言
  - 现代语言中很少/不会遇到
  - 不要在C语言特定的bug上耗费太多时间!
  - 举例
    - 内存安全（第9章）
      - 数组越界，空指针，段错误
    - 类型安全
      - 指针类型可随意转换
    - 链接的奇怪bug（第7章）
      - 全局变量解析
    - 难用的网络编程（第11章）/多线程（第12章）接口

问题3: 编程语言

- 这是一本经典**老**教材!
- 编程语言不仅决定会遇到什么bug，还从根本上影响**软件架构**
- 用现代语言重读CSAPP!
  - 语言推荐（仅代表个人建议）
    - 入门：Kotlin, C#
    - 进阶：Scala, C++20
    - 发烧：Rust, Haskell

欢迎邮件交流!

zhenbangyou@pku.edu.cn