Pratham Mehta

NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 2

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistants

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 2 [released Tuesday September 20th 2022] [due Friday September 30th by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

---

1. After graduating, you are asked to become the lead computer designer at Hyper Computers, Inc. Your study of usage of high-level language constructs suggests that procedure calls are one of the most expensive operations. You have invented a scheme that reduces the loads and stores normally associated with procedure calls and returns. The first thing you do is run some experiments with and without this optimization. Your experiments use the same state-of-the-art optimizing compiler that will be used with either version of the computer. These experiments reveal the following information:

- The clock rate of the unoptimized version is 5% higher.
- 30% of the instructions in the unoptimized version are loads or stores.
- The optimized version executes 2/3 as many loads and stores as the unoptimized version. For all other instructions the dynamic counts are unchanged.
- All instructions (including load and store) take one clock cycle.

Which is faster? Justify your decision quantitatively.

Solution:

$\Rightarrow$

Problem

After graduating, you are asked to become the lead computer designer at Hyper Computers, Inc. Your study of usage of high-level language constructs suggests that procedure calls are one of the most expensive operations. You have invented a scheme that reduces the loads and stores normally associated with procedure calls and returns. The first thing you do is run some experiments with and without this optimization. Your experiments use the same state-of-the-art optimizing compiler that will be used with either version of the computer. These experiments reveal the following information:

- The clock rate of the unoptimized version is 5% higher.
- 30% of the instructions in the unoptimized version are loads or stores.
- The optimized version executes 2/3 as many loads and stores as the unoptimized version. For all other instructions the dynamic counts are unchanged.
- All instructions (including load and store) take one clock cycle.

Which is faster? Justify your decision quantitatively.

Solution

CPU performance equation: $CPUTime = IC * CPI * ClockTime$

We know:
$ClockTime_{unop} = 0.95 * ClockTime_{op}$
$IC_{ld/st,unop} = 0.3 * IC_{unop}$
$IC_{ld/st,op} = 0.67 * IC_{ld/st,unop}$
$IC_{others,unop} = IC_{others,op}$
$CPI = 1$
Thus:

$$\begin{aligned} CPUTime_{unop} &= IC_{unop} * ClockTime_{unop} &= 0.95 IC_{unop} * ClockTime_{op} \qquad (1) \\ CPUTime_{op} &= IC_{op} * ClockTime_{op} \end{aligned}$$

but,

$$IC_{op} = 0.7 * IC_{unop} + 0.3 * 0.67 * IC_{unop} = 0.9 * IC_{unop}$$

then,

$$CPUTime_{op} = 0.9 * IC_{unop} * ClockTime_{op} \qquad (2)$$

Comparing 1 and 2 we see that the optimized version is faster.

$$Speedup = \frac{0.95 * IC_{unop} * ClockTime_{op}}{0.9 * IC_{unop} * ClockTime_{op}} = 1.06$$

2. General-purpose processes are optimized for general-purpose computing. That is, they are optimized for behavior that is generally found across a large number of applications. However, once the domain is restricted somewhat, the behavior that is found across a large number of the target applications may be different from general-purpose applications. One such application is deep learning or neural networks. Deep learning can be applied to many different applications, but the fundamental building block of inference—using the learned information to make decisions—is the same across them all. Inference operations are largely parallel, so they are currently performed on graphics processing units, which are specialized more toward this type of computation, and not to inference in particular. In a quest for more performance per watt, Google has created a custom chip using tensor processing units to accelerate inference operations in deep learning.1 This approach can be used for speech recognition and image recognition, for example. This problem explores the trade-offs between this process, a general-purpose processor (Haswell E5-2699 v3) and a GPU (NVIDIA K80), in terms of performance and cooling. If heat is not removed from the computer efficiently, the fans will blow hot air back onto the computer, not cold air. Note: The differences are more than processor—on-chip memory and DRAM also come into play. Therefore statistics are at a system level, not a chip level.

a. If Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what is the speedup of the TPU system over the GPU system?

⇒ **FIRST METHOD OF SOLVING THIS**

Speed up system A over B = Performance of system A/Performance of System B

= Exec time of sys B/ Exec time of sys A Let the total time to finish both the workloads be T

Given,

$ET_{GPU\ on\ A} = 0.7T$ and $ET_{GPU\ on\ B} = 0.3T$

From the give tables;

Speed up of TPU over GPU for task A = 225000/13461 = 16.71

=> $ET_{TPU\ on\ A} * 16.71 = ET_{GPU\ on\ A} = 0.7T$

Speed up of TPU over GPU for task B = 280000/36465 = 7.67

=> $ET_{TPU\ on\ B} * 7.7 = ET_{GPU\ on\ B} = 0.3T$

Speed up of TPU over GPU for the overall task = $ET_{GPU\ on\ A\ and\ B} / ET_{TPU\ on\ A\ and\ B}$

= T/((0.7T/16.71) + (0.3T/7.7))

= 12.37

b. Google’s data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what percentage of Max IPS does it achieve for each of the three systems?

⇒

b. [10] <1.9> If Google’ on workload A and 30% of
its time on workload B ge of Max IPS does it
achieve for each of the

Step by step solution:

General Purpose unit => 0.42*0.7 + 0.3*1 = 0.594 => **59.4%**

Grapics processing unit => 0.37*0.7+0.3*1 = 0.559 => **55.9%**

TPU => 0.8*0.7+0.3*1 = 0.86 => **86%**

—---

c. Building on (b), assuming that the power scales linearly from idle to busy power as IPS grows from 0% to 100%, what is the performance per watt of the TPU system over the GPU system?

⇒

c. [15] <1.5, 1.9> Building on (b), assuming that the power scales linearly from idle to busy power as IPS grows from 0% to 100%, what is theperformance per watt of the TPU system over the GPU system?

Step by step solution:

Performance per watt = throughput/watt

Performance per watt of TPU over GPU = Performance per watt of TPU/ performance per watt of GPU

= (0.86/(384-290))/(0.559/(991-357))

= 0.86*634/0.559*94 = 545.24/52.54

= **10.367**

—--

d. If another data center spends 40% of its time on workload A, 10% of its time on workload B, and 50% of its time on workload C, what are the speedups of the GPU and TPU systems over the general-purpose system?

e. A cooling door for a rack cost $4000 and dissipates 14 kW (into the room; additional cost is required to get it out of the room). How many Haswell-, NVIDIA-, or Tensor-based servers can you cool with one cooling door, assuming TDP in Figures 1.27 and 1.28?

f. Typical server farms can dissipate a maximum of 200 W per square foot. Given that a server rack requires 11 square feet (including front and back clearance), how many servers from part (e) can be placed on a single rack, and how many cooling doors are required?

System	Chip	TDP	Idle power	Busy power
General-purpose	Haswell E5-2699 v3	504 W	159 W	455 W
Graphics processor	NVIDIA K80	1838 W	357 W	991 W
Custom ASIC	TPU	861 W	290 W	384 W

Figure 1.27 Hardware characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system, including measured power

System	Chip	Throughput			% Max IPS		
		A	B	C	A	B	C
General-purpose	Haswell E5-2699 v3	5482	13,194	12,000	42%	100%	90%
Graphics processor	NVIDIA K80	13,461	36,465	15,000	37%	100%	40%
Custom ASIC	TPU	225,000	280,000	2000	80%	100%	1%

Figure 1.28 Performance characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system on two neural-net workloads

ANOTHER METHOD OF SOLVING

—

a. If Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what is the speedup of the TPU system over the GPU system?

A_operations = 0.7*throughput_A = 9422.7;

B_operations = 0.3*throughput_B = 10939.5;

TPU_time = A_operations/throughput_A + B_oprations/throughput_B = 0.0419 + 0.0391 = 0.081

Speedup = GPU_time/TPU_time = 12.35

b. Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what percentage of Max IPS does it achieve for each of the three systems?

General-purpose:70%*42% + 30%*100% = 59.4%

GPU:70%*37% + 30%*100% = 55.9%

TPU:70%*80% + 30%*100% = 86%

c. Building on (b), assuming that the power scales linearly from idle to busy power as IPS grows from 0% to 100%, what is the performance per watt of the TPU system over the GPU system?

GPU_Performance_per_watt = 55.9%/(991W-357W)

TPU_Performance_per_watt = 86%/(384W-290W)

TPU over CPU = TPU_Performance_per_watt/ GPU_Performance_per_watt = 10.3764

d. If another data center spends 40% of its time on workload A, 10% of its time on workload B, and 50% of its time on workload C, what are the speedups of the GPU and TPU systems over the general-purpose system?

General-purpose_oprations: (assume time = 1)

A: 40%*5482 = 2192.8; B: 10%*13194 = 1319.4; C: 50%*12000 = 6000

GPU_time = 2192.8/13461 + 1319.4/36465 + 6000/15000 = 0.1629+0.0362+0.4 =

0.5991

GPU_speedup = 1/0.5991 = 1.6692

TPU_time = 2192.8/225000 + 1319.4/280000 + 6000/2000 = 0.0097+0.0047+3 = 3.0144

TPU_speedup = 1/3.0144 = 0.3317

e. A cooling door for a rack costs $4000 and dissipates 14 kW (into the room; additional cost is required to get it out of the room). How many Haswell-, NVIDIA-, or Tensor-based servers can you cool with one cooling door, assuming TDP in Figures 1.27 and 1.28?

Haswell: 14kW/504W = 27.78

NVIDIA:14kW/1838W = 7.62

TPU:14kW/861W = 16.26

Take them to integer, 27,7 and 16.

f. Typical server farms can dissipate a maximum of 200 W per square foot. Given that a server rack requires 11 square feet (including front and back clearance), how many servers from part (e) can be placed on a single rack, and how many cooling doors are required?

200W * 11 = 2200W

Haswell: 2200W/504W = 4 < 27, 1 cooling door

NVIDIA:2200/1838 = 1 < 7, 1 cooling door

TPU:2200/861 = 2 <16, 1 cooling door

—

3. In this exercise, assume that we are considering enhancing a quad-core machine by adding encryption hardware to it. When computing encryption operations, it is 20 times faster than the normal mode of execution. We will define percentage of encryption as the percentage of time in the original execution that is spent performing encryption operations. The specialized hardware increases power consumption by 2%.

a. Draw a graph that plots the speedup as a percentage of the computation spent performing encryption. Label the y-axis "Net speedup" and label the x-axis "Percent encryption."

b. With what percentage of encryption will adding encryption hardware result in a speedup of 2?

c. What percentage of time in the new execution will be spent on encryption operations if a speedup of 2 is achieved?

4. Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 50% of the time, measured as a percentage of the execution time when the enhanced mode is in use. Recall that Amdahl's Law depends on the fraction of the original, unenhanced execution time that could make use of enhanced mode. Thus, we cannot directly use this 50% measurement to compute speedup with Amdahl's Law.

a. What is the speedup we have obtained from fast mode?

b. What percentage of the original execution time has been converted to fast mode?

5. When parallelizing an application, the ideal speedup is speeding up by the number of processors. This is limited by two things: percentage of the application that can be parallelized and the cost of communication. Amdahl's Law takes into account the former but not the latter.

a. What is the speedup with N processors if 80% of the application is parallelizable, ignoring the cost of communication?

b. What is the speedup with eight processors if, for every processor added, the communication overhead is 0.5% of the original execution time.

c. What is the speedup with eight processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

d. What is the speedup with N processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

e. Write the general equation that solves this question: What is the number of processors with the highest speedup in an application in which P% of the original execution time is parallelizable, and, for every time the number of processors is doubled, the communication is increased by 0.5% of the original execution time?

Answer: Speedup Overall Formula:

$$3.\quad \textit{Amdahl's Law: } \text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \dfrac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

a. 1 / (0.2 + 0.8/N)

b. 1 / (0.2 + 8 × 0.005 + 0.8/8) = 2.94

c. 1 / (0.2 + 3 × 0.005 + 0.8/8) = 3.17

d. 1 / (0.2 + logN × 0.005 + 0.8/N)

e. d / dN (1 / ((1 – P) + logN × 0.005 + P/N)) = 0

6. Your company has just bought a new 22-core processor, and you have been tasked with optimizing your software for this processor. You will run four applications on this system, but the resource requirements are not equal. Assume the system and application characteristics listed in Table 1.1 below (from textbook)

Table 1.1 Four applications

Application	A	B	C	D
% resources needed	41	27	18	14
% parallelizable	50	80	60	90

The percentage of resources of assuming they are all run in serial. Assume that when you parallelize a portion of the program by X, the speedup for that portion is X.

a. How much speedup would result from running application A on the entire 22-core processor, as compared to running it serially?

b. How much speedup would result from running application D on the entire 22-core processor, as compared to running it serially?

c. Given that application A requires 41% of the resources, if we statically assign it 41% of the cores, what is the overall speedup if A is run parallelized but everything else is run serially?

d. What is the overall speedup if all four applications are statically assigned some of the cores, relative to their percentage of resource needs, and all run parallelized?

e. Given acceleration through parallelization, what new percentage of the resources are the applications receiving, considering only active time on their statically-assigned cores?

—---

Solution:

Theoretical Task 3

You got a new 22-core processor from AMD, and you have been tasked with optimizing your software for this processor. You run four applications on this system, but the resource requirements are not equal for these applications. Assume the system and application characteristics.

The percentage of resources assumes they are all run in serial. Assume that when you parallelize a portion of the program by X, the speedup for that portion is X.

a. **How much speedup would result from running application A on the entire 22-core processor, as compared to running it serially?**

Solution:

Parallelizable portion of A is 50%. so, speedup is 50

b. **How much speedup would result from running application D on the entire 22-core processor, as compared to running it serially?**

Solution:

Parallelizable portion of D is 90%. so, speedup is 90

c. **Given that application A requires 41% of the resources, if we statically assign it 41% of the cores, what is the overall speedup if A is run parallelized but everything else is run serially?**

Solution:

Cores by A = 41% of 22 = 9 cores

$$\text{Speed up} = \frac{1}{0.41 \times \left(\frac{0.5}{9}+0.5\right)+0.27+0.18+0.14} \text{(50\% of A is parallelized)}$$

$$= \frac{1}{0.228+0.27+0.18+0.14} = \frac{1}{0.818} = 1.22$$

d. What is the overall speedup if all four applications are statically assigned some of the cores, relative to their percentage of resource needs, and all run parallelized?

Solution:

Cores assigned to each application relative to resource needs

Cores by A = 41% of 22 $\approx 9\ cores$

Cores by B = 27% of 22 $\approx 6\ cores$

Cores by C = 18% of 22 $\approx 4\ cores$

Cores by D = 14% of 22 $\approx 3\ cores$

Speed up = $\frac{1}{0.41 \times \left(\frac{0.5}{9}+0.5\right)+0.27+\left(\frac{0.8}{6}+0.2\right)+0.18 \times \left(\frac{0.6}{4}+0.4\right)+0.14 \times \left(\frac{0.9}{3}+0.1\right)}$

$=\frac{1}{0.228+0.0899+0.099+0.056} = \frac{1}{0.4729} = 2.115$

e. Given acceleration through parallelization, what new percentage of the resources are the applications receiving, considering only active time on their statically assigned cores?

Solution:

A = $0.41 \times \left(\frac{0.5}{9} + 0.5\right) = 0.228 = 22.8\%$

B = $0.27 \times \left(\frac{0.8}{6} + 0.2\right) = 0.0879 = 8.99\%$

A = $0.18 \times \left(\frac{0.6}{4} + 0.4\right) = 0.099 = 9.9\%$

A = $0.14 \times \left(\frac{0.9}{3} + 0.1\right) = 0.056 = 5.6\%$

7. When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floating-point unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's Law equation does not take into account this trade-off.

a. If the new fast floating-point unit speeds up floating-point operations by, on average, 2x, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

b. Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5x slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

c. After implementing the new floating-point operations, what percentage of execution time is spent on floating-point operations? What percentage is spent on data cache accesses?

Solution:

Answer and Explanation:

(a) Given fraction $=20\%$ = 0.20

Speed up is considered as 2

speed up$=\frac{1}{(1-fraction)+\frac{fraction}{speedup}} = \frac{1}{(1-0.20)+\frac{0.20}{2}} = 1.11$

(b) For the speed-up, the floating-point unit gets slower i.e. 1.5

speed-up = $\frac{2}{3}$ times

Consumption of data cache = 10%(of execution time)

Now, calculating overall speed-up:

Speed-up = $\frac{1}{0.7+\frac{0.20}{2}+0.10\times\frac{3}{2}} = 1.05$

(c) Fractional speed-up=0.20

Fraction=0.70

improvements difference=0.70-0.20=0.5

Therefore, improvement percentage= 5%

Execution time ratio = 0.7:0.1:0.15

Percentage of time spent on floating point = $\frac{1}{0.7+0.1+0.15} = 0.105 = 10.5\%$

Therefore, the cache percentage=5%

The floating-point=10.5%

8. When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floating-point unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's Law equation does not take into account this trade-off.

a. If the new fast floating-point unit speeds up floating-point operations by, on average, 2x, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

b. Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5x slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

c. After implementing the new floating-point operations, what percentage of execution time is spent on floating-point operations? What percentage is spent on data cache accesses?

Solution:

Answer and Explanation:

(a) Given fraction =20% = 0.20

Speed up is considered as 2

speed up=$\frac{1}{(1-fraction)+\frac{fraction}{speedup}} = \frac{1}{(1-0.20)+\frac{0.20}{2}} = 1.11$

(b) For the speed-up, the floating-point unit gets slower i.e. 1.5

speed-up = $\frac{2}{3}$ times

Consumption of data cache = 10%(of execution time)

Now, calculating overall speed-up:

Speed-up = $\frac{1}{0.7+\frac{0.20}{2}+0.10\times\frac{3}{2}} = 1.05$

(c) Fractional speed-up=0.20

Fraction=0.70

improvements difference=0.70-0.20=0.5

Therefore, improvement percentage= 5%

Execution time ratio = 0.7:0.1:0.15

Percentage of time spent on floating point = $\frac{1}{0.7+0.1+0.15} = 0.105 = 10.5\%$

Therefore, the cache percentage=5%

The floating-point=10.5%

$\Rightarrow$