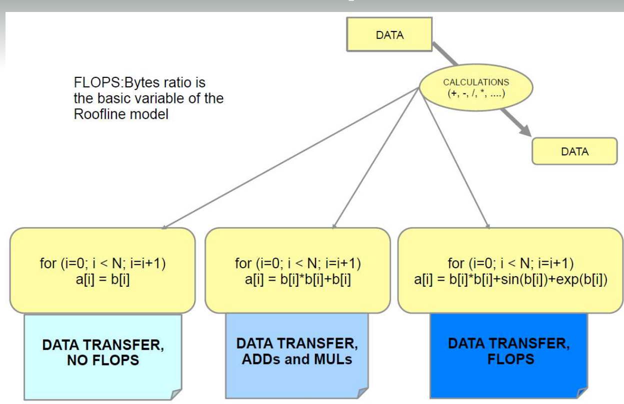
Roofline, arithmetic intensity

slides from Grinberg, IBM

ICSC 2014

FLOPS:Bytes-moved (AI)

FLOPS:Bytes ratio is the basic variable of the Roofline model



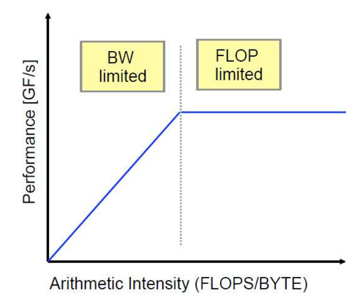
Roofline model

The Roofline Model - is a tool to understand the kernel/hardware limitation and it is also a tool for kernel optimization

Performance is upper **bounded** by:

1) the **peak flop** rate

2) the streaming **bandwidth**

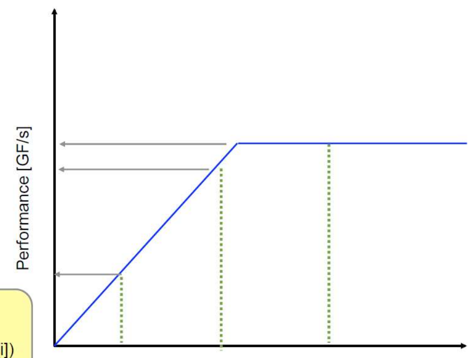


Where is your algorithm?

for (i=0; i < N; i=i+1)
a[i] = 2.3*b[i]

for (i=0; i < N; i=i+1)
a[i] = b[i]*b[i]+b[i]

for (i=0; i < N; i=i+1)
a[i] = b[i]*b[i]+sin(b[i])+exp(b[i])



Examples

FLOPS / Bytes ratio – one of the basic characteristics of a kernel

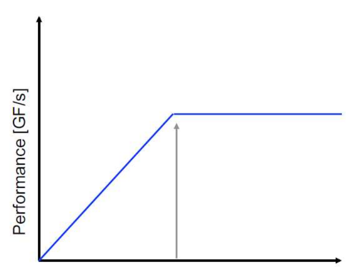
for (i = 0; i < N; ++i) z[i] = x[i]+y[i]	1 ADD 2 (8 byte) loads 1 (8 byte) write AI = 1 / (2*8 + 8) = 1/24*
for (i = 0; i < N; ++i) z[i] = x[i]+y[i]*x[i]	1 ADD 1 MUL 2 (8 byte) loads 1 (8 byte) write AI = 2 / (2*8 + 8) = 1/12*

for (i = 0; i < N; ++i){
I1 = A_offset[i]; I2 = A_offset[i+1];
sum = 0.0
for (j = 0; j < (I2-I1); ++j)
sum += A[I1+j] * x[col_index [I2+j]];
y[i] = sum;
}
	1 ADD
	1 MUL
	2 (8 byte) + 1 (4 bytes) loads
	1 (8 byte) write
	AI = 2 / (2*8 + 4 + 8) = 1/14

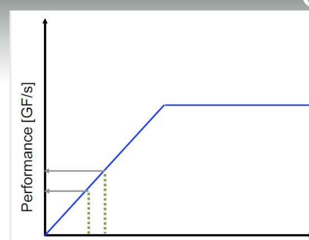
Hard

It is an art to find a perfect match between kernel and hardware characteristics

In another words it requires a lot of work to create a kernel that will exhaust both, the memory BW and FLOPs capacity *at the same time.* (many times it is even impossible ….)



Order can matter



N – is large, i.e., buffer does not fit cache

```
for (i=0; i < N; ++i)
   a[i] = buffer[i] + b[i];

for (i=0; i < N; ++i)
   c[i] = buffer[i] + d[i];
```

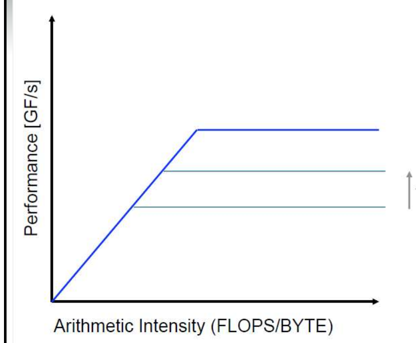
AI_total = 2 / (2 * 3 * 8) = 1/24;

```
for (i=0; i < N; ++i){
   a[i] = buffer[i] + b[i];
   c[i] = buffer[i] + d[i];
}
```

AI = 2/(5*8) = 1 / 20;

----

more




```
sum = 0;
for (i=0; i < N; ++i)
   sum = sum + a[i];

sum0 = sum1 = sum2 = sum3 = 0;
for (i=0; i < N; i+=4){
   sum0 = sum0 + a[i    ];
   sum1 = sum1 + a[i+1];
   sum2 = sum2 + a[i+2];
   sum3 = sum3 + a[i+3];
}
sum0 = sum0+sum1;
sum2 = sum2+sum3;
sum = sum0+sum2;
```


More processors == better?

If we run a problem on p processors, will it run p times faster?

- yes
- no
- maybe

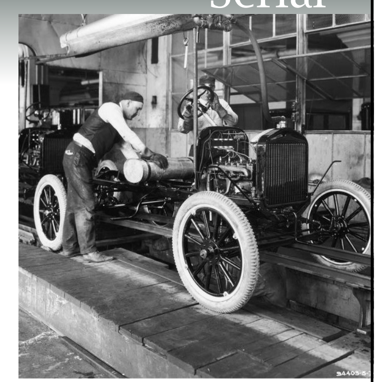
Flynn's taxonomy

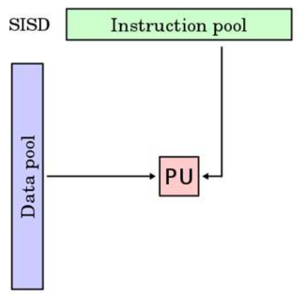
Some combination of instruction and data:

- SISD (single instr, single data)
- SIMD (single instr, multiple data)
- MISD (multiple instr, single data)
- MIMD (multiple instr, multiple data)

Serial

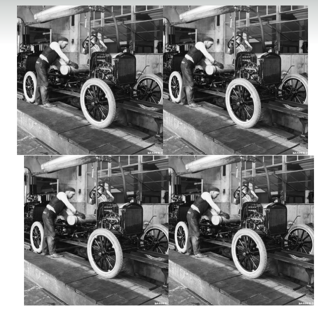
- One worker builds one car, one at a time, do all steps necessary to assemble the car
- taxonomy?

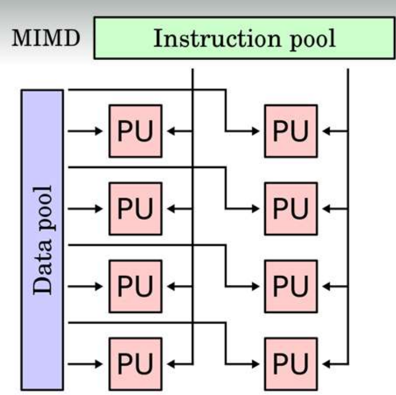




Data Parallelism

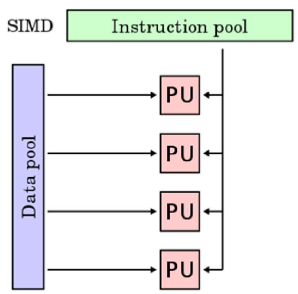
- N workers build N cars simultaneously, each worker completely all steps necessary to assemble one car.
- Taxonomy type?



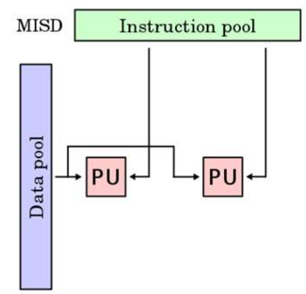


Functional

- N workers build one car, each worker completes some unique subset of the steps necessary to assemble one car
- taxonomy?

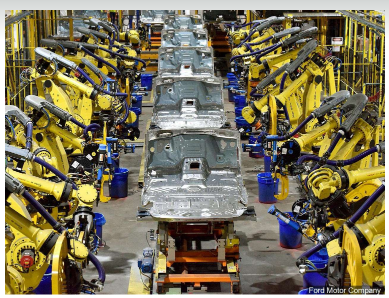


The weird one



Data/Function Parallelism

- N workers build N cars simultaneously, each worker completely some unique subset of the steps necessary to assemble one car
- Notice the analogy with communication and bandwidth represented by the moving assembly line!



Speedup, efficiency, and Amdahl

Speedup: $S_p = T_1/T_p.$

Efficiency: $E_p = S_p/p.$

T_1 : *the time the computation takes on a single processor*
T_p : *the time the computation takes with p processors*
T_∞ : *the time the computation takes if unlimited processors are available*
P_∞ : *the value of p for which $T_p = T_\infty$*

Amdahl's Law:

$$T_P = T_1(F_s + F_p/P).$$

asymptotically, $S_P \leq 1/F_s$

with communication:

$$T_p = T_1(F_s + F_p/P) + T_c,$$

Exercise 2.10

Exercise 2.10. Let's do a specific example. Assume that a code has a setup that takes 1 second and a parallelizable section that takes 1000 seconds on one processor. What are the speedup and efficiency if the code is executed with 100 processors? What are they for 500 processors? Express your answer to at most two significant digits.

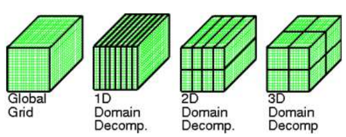
- Sequential time: $T_1 = 1001$
- With 100 processors: $T_{100} = 11$, $S_{100} = 1001/11 \sim 91$, $E_{100} \sim 0.91$
- With 500 processors: $T_{500} = 3$, $S_{500} = 333$, $E_{500} \sim 0.67$

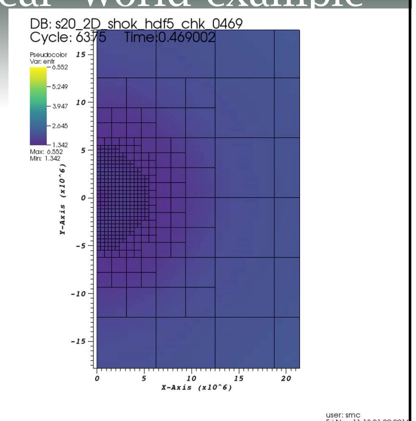
Scalability

- Strong scaling: Fixed problem size, increased number of processes
- Weak scaling: Fixed problem size *per process*

"Real" world example

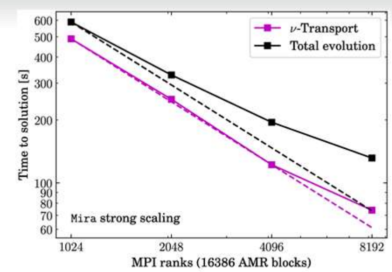
- Domain decomposition in a fluid calculation





Scalability

- Strong scaling: challenging!
- Hard to fit big problem on few processes
- Communication cost AND serial fraction are major limiters



Scalability

- Weak scaling: better idea of how application will perform "at scale"
- Strong and weak scaling together are needed to get a sense of real efficiency