Session 4

Performance measurements

Massimiliano Fasi

Durham University

Overview

Roofline

- comparison between hardware configurations
- high-level overview of code performance
- general guidance for optimisation

Overview

Roofline

- comparison between hardware configurations
- high-level overview of code performance
- general guidance for optimisation

Measurements

- to get more information about bottleneck
- to confirm the hypothesis formed through roofline analysis

Performance measurements

Special purpose **registers**:

- common in modern hardware
- record low-level performance events
  - number of Flops of different type (scalar, sse, avx)
  - cache miss/hit counts at various levels
  - branch prediction success rate
  - ...

Performance measurements

Special purpose **registers**:

- common in modern hardware
- record low-level performance events
  - number of Flops of different type (scalar, sse, avx)
  - cache miss/hit counts at various levels
  - branch prediction success rate
  - ...
- can be overwhelming
- best used to confirm hypothesis from some model

Caveats

- Information about
  - the algorithm you implemented
  - the way you implemented it
  - the data moved in the measured run
- Does not consider
  - potentially better algorithms
  - potentially superior ways of implementing those
  - data you could have moved in a different run

Caveats

- Information about
  - the algorithm you implemented
  - the way you implemented it
  - the data moved in the measured run
- Does not consider
  - potentially better algorithms
  - potentially superior ways of implementing those
  - data you could have moved in a different run

Only meaningful as complements to models.

Granularity

- Direct read of low-level hardware counters
  - most detailed
  - hardware dependent
  - not portable
- Abstract metrics
  - groups of low-level counters
  - easier to compare across hardware

“instructions” $\rightarrow$ “instructions per cycle”

How do we measure them?

- Use `likwid-perfctr` (installed on Hamilton via the `likwid` module).
- Offers a reasonably friendly command-line interface.
- Provides access both to counters directly, and many useful predefined “groups”.
- Will use `likwid-perfctr` to measure memory references in different implementations of the same loop.

Example: STREAM

Scalar

```
for i from 0 to n:
  load a[i:i+1] reg1
  load b[i:i+1] reg2
  load c[i:i+1] reg4
  mul reg1 reg2 reg3
  add reg4 reg3 reg4
  store reg4 c[i:i+1]
```

SSE

```
for i from 0 to n by 2:
  vload a[i:i+2] vreg1
  vload b[i:i+2] vreg2
  vload c[i:i+2] vreg4
  vmul vreg1 vreg2 vreg3
  vadd reg4 reg3 reg4
  vstore reg4 c[i:i+2]
```

AVX

```
for i from 0 to n by 4:
  vload a[i:i+4] vreg1
  vload b[i:i+4] vreg2
  vload c[i:i+4] vreg4
  vmul vreg1 vreg2 vreg3
  vadd reg4 reg3 reg4
  vstore reg4 c[i:i+4]
```

AVX2

```
for i from 0 to n by 4:
  vload a[i:i+4] vreg1
  vload b[i:i+4] vreg2
  vload c[i:i+4] vreg3
  vfma vreg1 vreg2 vreg3
  vstore reg3 c[i:i+4]
```

Measurement

Model

For $N = 10^6$, how many loads and stores in each case?

Measurement

Model

For N = 10^6, how many loads and stores in each case?

Answer

Each loop iteration has 3 loads and 1 store.
With vector width W and N iterations we need:

- $\frac{3N}{W}$ loads
- $\frac{N}{W}$ stores

Exercise 5: Models and measurements

1. Split into small groups
2. Make sure one person per group has access to Hamilton
3. Download the STREAM TRIAD benchmark
4. Compile with `likwid` annotations
5. Measure loads and stores
6. Ask questions!