



Profiling a Memory-Access Kernel

Linux perf | Practical 1

Ömer Gölcük — Sabancı University

Course lead: Prof. Dr. Özcan Öztürk

The operation

$$B[j, i] = 2 \times A[i, j] + 1$$

Scale each value and write it to the transposed position.

Input A

0	1	2
3	4	5
6	7	8



Output B

1	7	13
3	9	15
5	11	17

Separate arrays; row-major storage; 8-byte doubles.

The kernel

```
void scaled_transpose(const double* a, double* b, std::size_t n)
{
    for (std::size_t i = 0; i < n; ++i)
        for (std::size_t j = 0; j < n; ++j)
            b[j * n + i] = 2.0 * a[i * n + j] + 1.0;
}
```

i selects an input row; j moves across that row.

Each output element is written exactly once per pass.

Read order and write order

Within the inner loop: i is fixed and j increases.

Read A

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

0 → 1 → 2 → 3

+1 element = +8 bytes

Write B

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

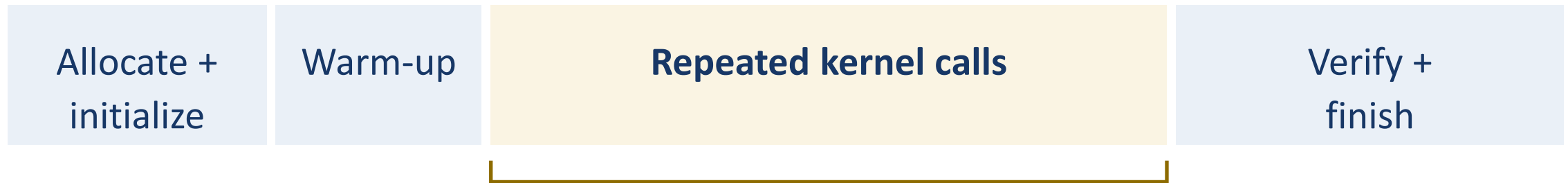
0 → 4 → 8 → 12

+n elements = +8n bytes

Illustration: element offsets for n = 4. This is an address sequence, not a measured cache trace.

Two measurement scopes

The program reports elapsed time per pass. The counters cover a wider interval.



C++ timer: measured loop only

perf with :u: user-space activity throughout the command

[1, 2] External perf collection includes setup, warm-up, repeated passes and validation.

Interpreting the live results

Elapsed time

Use repeated, unprofiled runs as the baseline.

The timer reports the cost of the repeated loop.

Instructions and cycles

$IPC = \text{instructions} \div \text{cycles}$, for matching event scopes.

Low IPC alone does not identify the limiting resource.

Sampled profile

Locate the hot function, then inspect source and assembly.

A sample is not a stopwatch for the instruction shown.

[1–3] Counting and sampling provide different evidence. Check event support, scope and collection warnings.

References

[\[1\] Linux perf tutorial](#)

Counting events and interpreting sampled profiles.

[\[2\] perf-stat\(1\) manual page](#)

Event counting, scope modifiers and collection warnings; see also perf-record(1).

[\[3\] perf-annotate\(1\) manual page](#)

Source and assembly views of a recorded profile; see also perf-report(1).

Thank you!



**Co-funded by
the European Union**



EuroHPC
Joint Undertaking



EuroCC 3 has received funding from the [European High-Performance Computing Joint Undertaking \(JU\)](#) under Grant Agreement No. 101306701. The JU receives support from the [European Union](#)'s Digital Europe Programme and Germany, Albania, Austria, Belgium, Bosnia and Herzegovina, Bulgaria, Croatia, Cyprus, Czechia, Denmark, Estonia, Finland, France, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Montenegro, the Netherlands, North Macedonia, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Türkiye, and Kosovo.

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or EuroHPC Joint Undertaking. Neither the European Union nor the EuroHPC Joint Undertaking can be held responsible for them.