OpenMP loops: A primer

- OpenMP provides a loop construct that specifies that the iterations of one or more associated loops will be executed in parallel by threads in the team in the context of their implicit tasks.[1]

```
#pragma omp for [clause[ [,] clause] ... ]
   for (int i=0; i<100; i++){}
```

- Loop needs to be in canonical form.
- The *clause* can be one or more of the following: `private(…), firstprivate(…), lastprivate(…), linear(…), reduction(…), schedule(…), collapse(...), ordered[…], nowait, allocate(…)`
- We focus on the clause ***`schedule(…)`*** in this talk.

1: OpenMP Technical Report 6. November 2017. http://www.openmp.org/press-release/openmp-tr6/

Credits: Vivek Kale

A Schedule for an OpenMP loop

```
#pragma omp parallel for schedule([modifier [modifier]:]kind[,chunk_size])
```

- A ***schedule*** in OpenMP is:
  - a specification of how iterations of associated loops are divided into contiguous non-empty subsets
    - We call each of the contiguous non-empty subsets a *chunk*
  - *and* how these chunks are distributed to threads of the team.[1]
- The *size of a chunk*, denoted as `chunk_size` must be a positive integer.

1: OpenMP Technical Report 6. November 2017. http://www.openmp.org/press-release/openmp-tr6/

Credits: Vivek Kale

The Kind of a Schedule

- A schedule *kind* is passed to an OpenMP loop schedule clause:
  - provides a hint for how iterations of the corresponding OpenMP loop should be assigned to threads in the team of the OpenMP region surrounding the loop.
- Five *kinds of schedules* for OpenMP loop[1]:
  - `static`
  - `dynamic`
  - `guided`
  - `auto`
  - `runtime`
- The OpenMP implementation and/or runtime defines how to assign chunks to threads of a team given the kind of schedule specified by as a hint.

1: OpenMP Technical Report 6. November 2017. http://www.openmp.org/press-release/openmp-tr6/

Credits: Vivek Kale

› `schedule(static[, chunk_size])`

- Iterations are divided into chunks of `chunk_size`, and chunks are assigned to threads before entering the loop
- If `chunk_size` unspecified, = `NITER/NTHREADS` (with some adjustement…)

› `schedule(dynamic[, chunk_size])`

- Iterations are divided into chunks of `chunk_size`
- At runtime, each thread requests for a new chunk after finishing one
- If `chunk_size` unspecified, then = 1

credits: Paolo Burgio

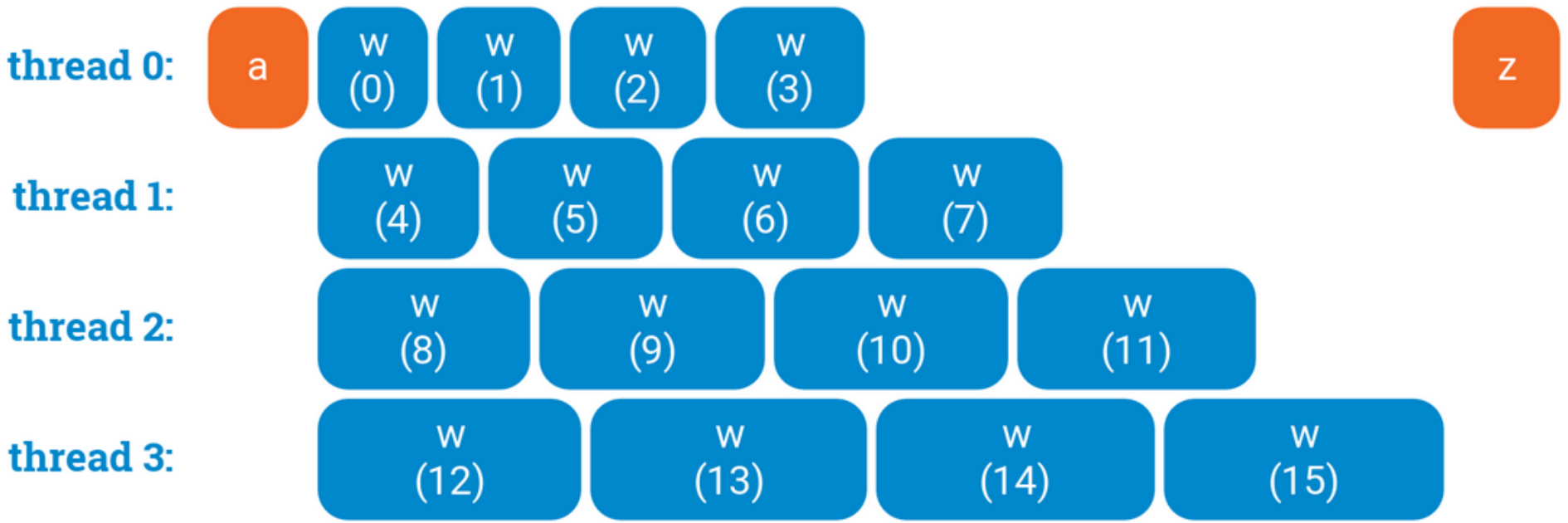
- `schedule(guided[, chunk_size])`
  - A mix of static and dynamic
  - `chunk_size` determined statically, assignment done dynamically

- `schedule(auto)`
  - Programmer let compiler and/or runtime decide
  - Chunk size, thread mapping..
  - "I wash my hands"

- `schedule(runtime)`
  - Only runtime decides according to run-sched-var ICV
  - If run-sched-var = `auto`, then implementation defined

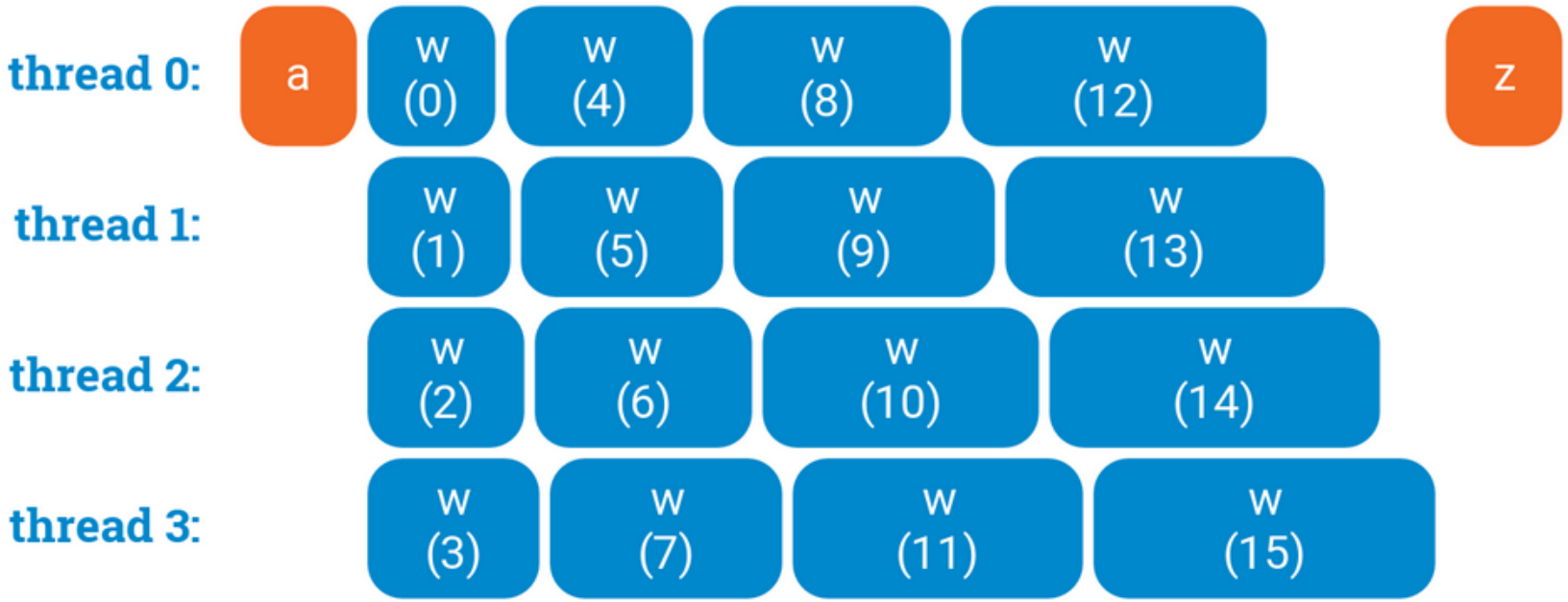
credits: Paolo Burgio

```
a();
#pragma omp parallel for
for (int i = 0; i < 16; ++i) {
    w(i);
}
z();
```



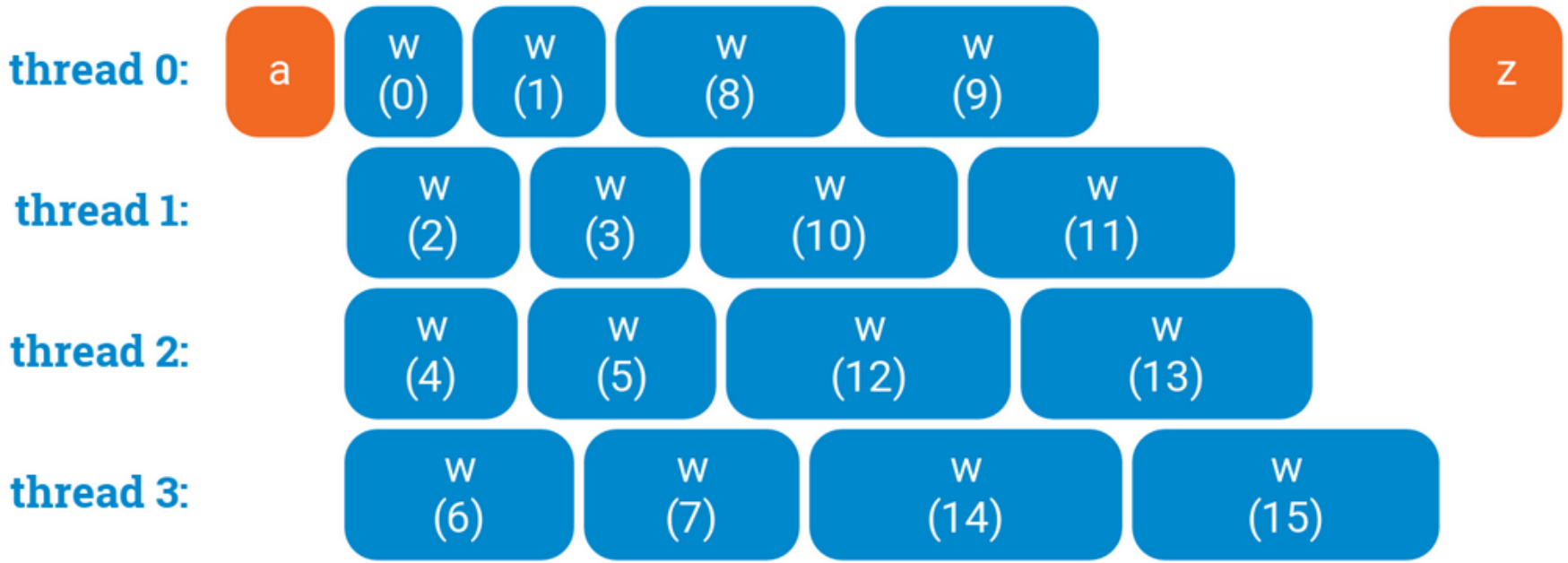
credits Aalto University

```
a();
#pragma omp parallel for schedule(static,1)
for (int i = 0; i < 16; ++i) {
    w(i);
}
z();
```



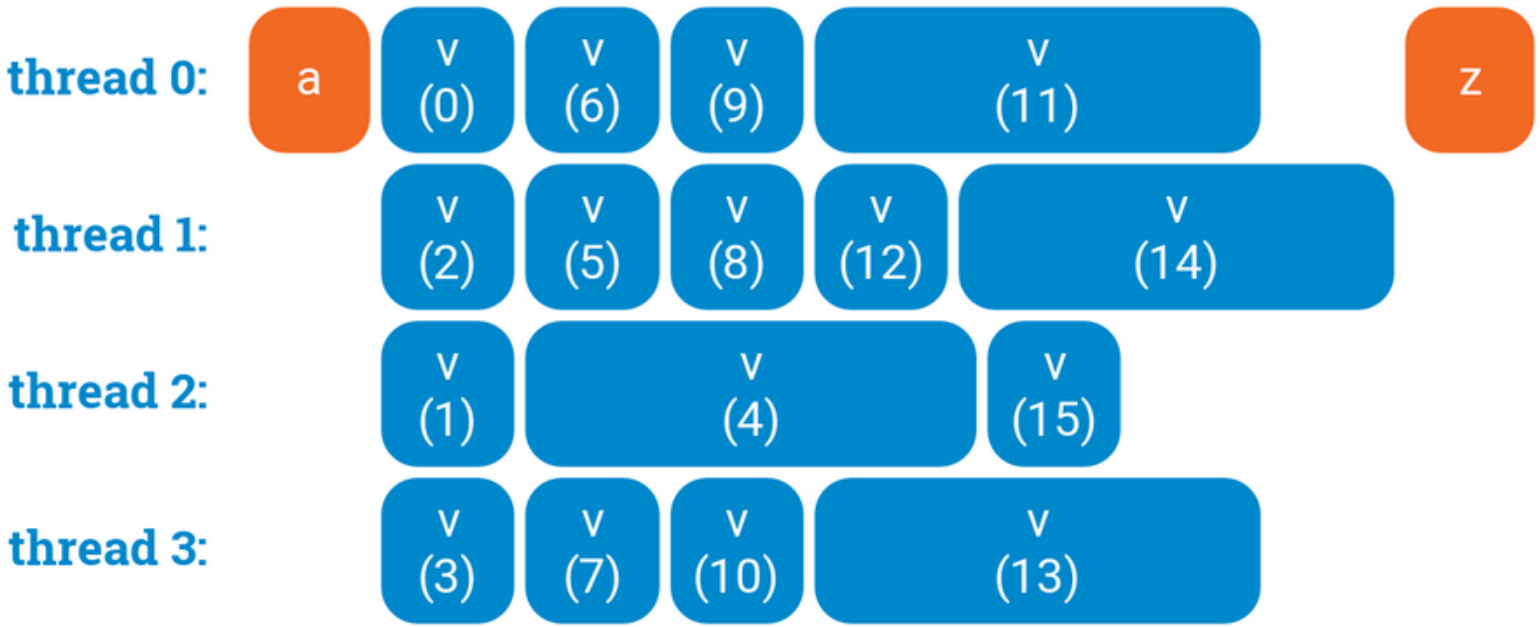
credits Aalto University

```
a();
#pragma omp parallel for schedule(static,2)
for (int i = 0; i < 16; ++i) {
    w(i);
}
z();
```



credits Aalto University

```
a();
#pragma omp parallel for schedule(dynamic,1)
for (int i = 0; i < 16; ++i) {
    v(i);
}
z();
```



credits Aalto University

Static vs. Dynamic

- Static has less initialization cost
- Dynamic has better load balancing
- Dynamic scheduling is expensive (setup, fetching work, loop exit)