# OpenMP
# OpenACC
# Offloading
# BoF


# OpenMP

- OpenMP 5.0
  - Public draft in July 2018
  - To be finalized early October 2018
  - To be released in November 2018
- GCC support – WIP on gomp-5_0-branch
  - Won't be ready for GCC9, but plan to merge a subset of features that are complete and stable


# What's new in OpenMP 5.0

- Support for C11, C++{11,14,17}, Fortran 2008
- Host `teams` construct (mainly for NUMA)
- Loops with `!=` condition, selected non-rectangular loop nests, non-perfectly nested nests
- Task reductions
- Conditional lastprivate
- Task `depend` iterators, dependency objects, `mutexinoutset` dependency type
- Allocators: `omp_alloc`, `allocate` directive and `allocator` clause


# What's new in OpenMP 5.0 (cont)

- Metadirective and context dependent callee selection
- New clauses for simd: `if` and `nontemporal`
- Exclusive and inclusive scan worksharing loop and simd support
- New `loop` directive (similar to OpenACC loops)
- Thread affinity query APIs and env vars
- New clause for `taskwait`: `depend`
- New clauses for `atomic: acq_rel`, `acquire`, `release`, `hint`
- OMPT and OMPD


redhat.

# What's new in OpenMP 5 offloading

- `requires` directive – allows requiring e.g. shared memory only offloading, or device to host offloading, etc.
- `ancestor` clause on `target` for device to host offload
- Non-contiguous arrays including strided arrays in `target update`
- `declare mapper`, lvalue array section bases, mapping of references, mapping of `this[:1]`, member mapping changes
- `defaultmap` clause extensions
- `omp_get_device_num` API
- `omp_pause_resource{,_all}` APIs


redhat.

# Enhanced map clause expressions

- // sp base pointer:
  `#pragma omp target map((sp+ofst)->x[ofst2].y[j:n]) // valid`
- `#pragma omp target map(x[1].y[10], x[1].y[11])  // invalid`
- `#pragma omp target map(x[1].a, x[1].b) // valid`
- `#pragma omp target map(x[1].yp[2:4], x[2]) // valid`
- `p = &x[0];`
  `#pragma omp target map(p[0], x[2:4]) // valid`
- `struct A{int &a;}; struct B{A &b;} struct C{B &c;};`
  `extern C d; extern C e[10];`
  `#pragma omp target map(d, e[2:4]) // deep copy`
- `struct A; struct B {A &b;}; struct C {B &c;}; struct A{B &a;};`
  `extern C d;`
  `#pragma omp target map(d) // invalid`


redhat.

# Declare mapper

- ```
  #pragma omp declare mapper (S v) \
  map(tofrom: v) map(alloc: v.a[0:100]) \
  map(to: v.b.c->d[5]->e[0:10])
  #pragma omp declare mapper (foo: S t) \
  map(to: t) map(alloc: t.z[2])
  #pragma omp target map(mapper(foo), to:a) \
  map(tofrom: b)
  ```



  ```