Summing vector elements in C using OpenMP – openmp.org

```
#pragma omp parallel for reduction(+: s)
for (int i = 0; i < n; i++) {
  s += x[i];
}
```

Per element multiply in C++ using Intel® Array Building Blocks – intel.com/go/arbb

```
void products( const arbb::dense<arbb::f32>& a,
               const arbb::dense<arbb::f32>& b,
               arbb::dense<arbb::f32>& c) {
  c = a * b;
}
```

Dot product in Fortran using OpenMP – openmp.org

```
!$omp parallel do reduction ( + : adotb )
  do j = 1, n
    adotb = adotb + a(j) * b(j)
  end do
!$omp end parallel do
```

Sum in Fortran, using co-array feature – intel.com/software/products

```
REAL SUM[*]
CALL SYNC_ALL( WAIT=1 )
DO IMG= 2,NUM_IMAGES()
  IF (IMG==THIS_IMAGE()) THEN
    SUM = SUM + SUM[IMG-1]
  ENDIF
  CALL SYNC_ALL( WAIT=IMG )
ENDDO
```

Parallel function invocation in C using Intel® Cilk™ Plus – cilk.org

```
cilk_for (int i=0; i<n; ++i) {
  Foo(a[i]);
}
```

Parallel function invocation in C++ using Intel® Threading Building Blocks – threadingbuildingblocks.org

```
parallel_for (0, n,
  [=](int i) { Foo(a[i]); }
);
```