```
KOKKOS_INLINE_FUNCTION void
divergence_sphere (const KernelVariables &kv,
                   const ExecViewUnmanaged<const Scalar [2][NP][NP][NUM_LEV]> v,
                   const ExecViewUnmanaged<      Scalar    [NP][NP][NUM_LEV]> div_v) const {
  const auto& Dinv = Homme::subview(m_dinv,kv.ie);
  const auto& metdet = Homme::subview(m_metdet,kv.ie);
  const auto& buf = Homme::subview(vector_buf_ml,kv.team_idx,0);
  Kokkos::parallel_for(Kokkos::TeamThreadRange(kv.team, NP*NP), [&](const int loop_idx) {
    const int igp = loop_idx / NP, jgp = loop_idx % NP;
    Kokkos::parallel_for(Kokkos::ThreadVectorRange(kv.team, NUM_LEV), [&] (const int& ilev) {
      const auto& v0 = v(0, igp, jgp, ilev);
      const auto& v1 = v(1, igp, jgp, ilev);
      buf(0,igp,jgp,ilev) = (Dinv(0,0,igp,jgp) * v0 + Dinv(1,0,igp,jgp) * v1) * metdet(igp,jgp);
      buf(1,igp,jgp,ilev) = (Dinv(0,1,igp,jgp) * v0 + Dinv(1,1,igp,jgp) * v1) * metdet(igp,jgp);
    });
  });
  kv.team_barrier();
  Kokkos::parallel_for(Kokkos::TeamThreadRange(kv.team, NP*NP), [&](const int loop_idx) {
    const int igp = loop_idx / NP, jgp = loop_idx % NP;
    Kokkos::parallel_for(Kokkos::ThreadVectorRange(kv.team, NUM_LEV), [&] (const int& ilev) {
      Scalar dudx(0.0), dvdy(0.0);
      for (int kgp = 0; kgp < NP; ++kgp) {
        dudx += dvv(jgp,kgp) * buf(0,igp,kgp,ilev);
        dvdy += dvv(igp,kgp) * buf(1,kgp,jgp,ilev);
      }
      div_v(igp,jgp,ilev) = (dudx + dvdy) * (1.0/metdet(igp,jgp) * PhysicalConstants::rrearth);
    });
  });
  kv.team_barrier();
}
```