Covariant programming : capturing the variation between SIMD and SIMT in a single code

WAS

```
LatticeFermionView a,b,c;
accelerator_for(ss,volume, {
   a[ss] = b[ss] + c[ss] ;
});
```

NOW

```
LatticeFermionView a,b,c;
accelerator_for(ss,volume,Spinor::Nsimd(), {
   coalescedWrite(a[ss], b(ss) + c(ss) );
});
```

On GPU accelerator for sets up a volume $\times$ Nsimd thread grid.

- Each thread is responsible for one SIMD lane

On CPU accelerator for sets up a volume OpenMP loop.

- Each thread is responsible for Nsimd() SIMD lanes

Per-thread datatypes inside these loops cannot be hardwired.

C++ auto and decltype use the return type of operator () to work out computation variables in architecture dependent way.

Single source high performance kernels are optimal
on BOTH CPU and CUDA