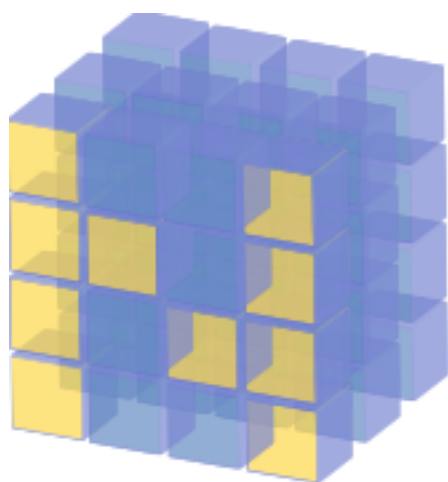
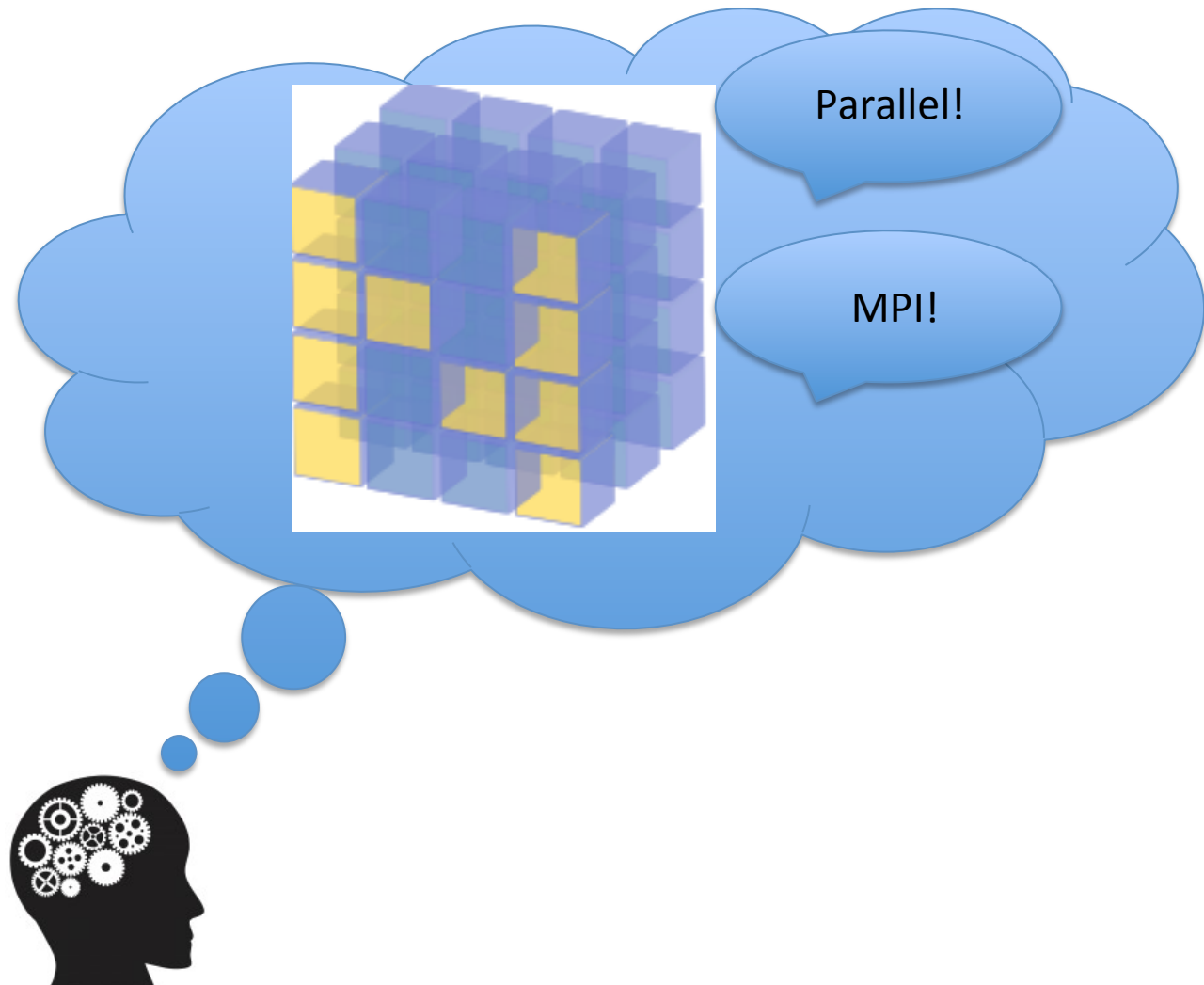


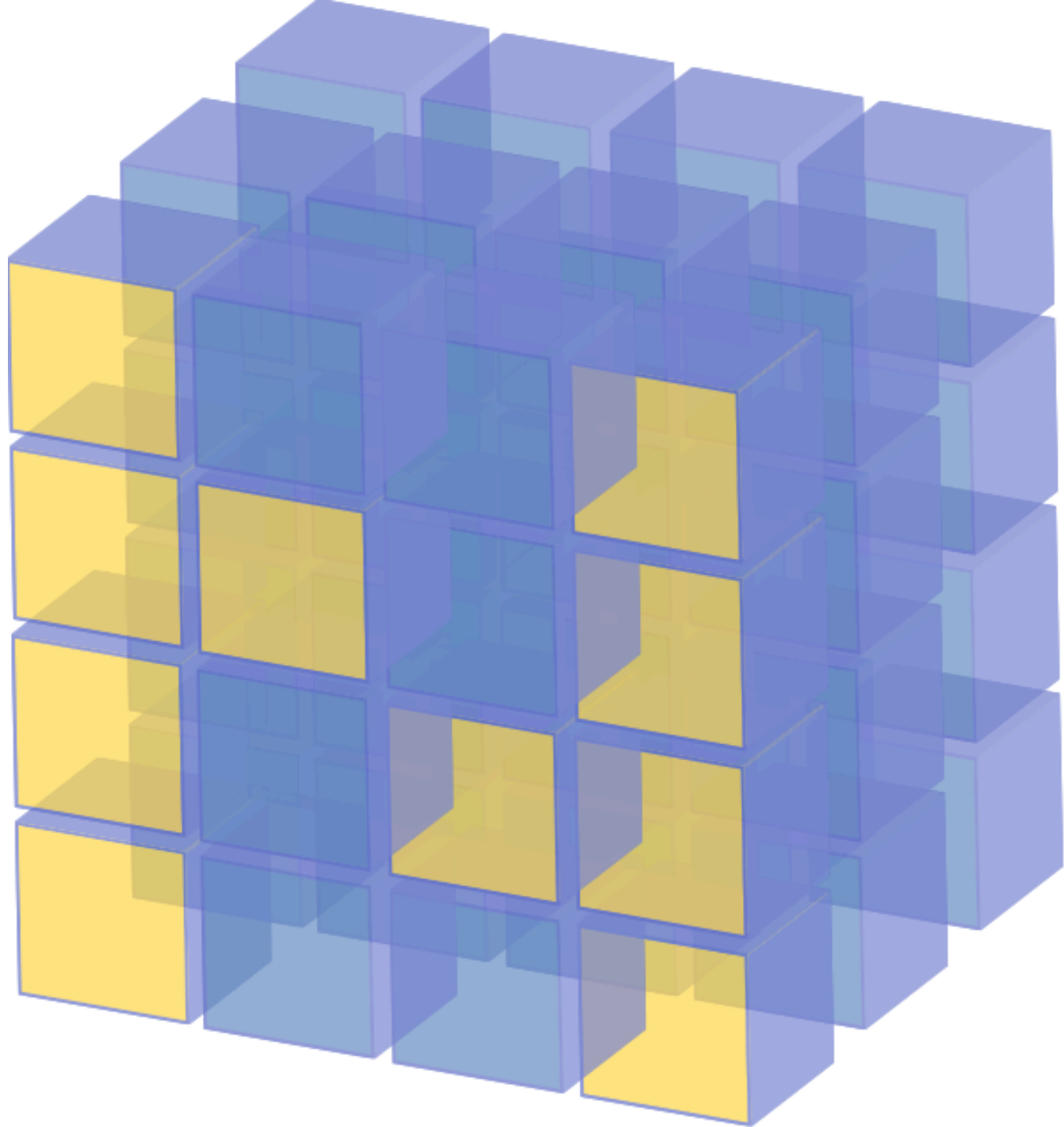
DistArray: NumPy + distributed computing

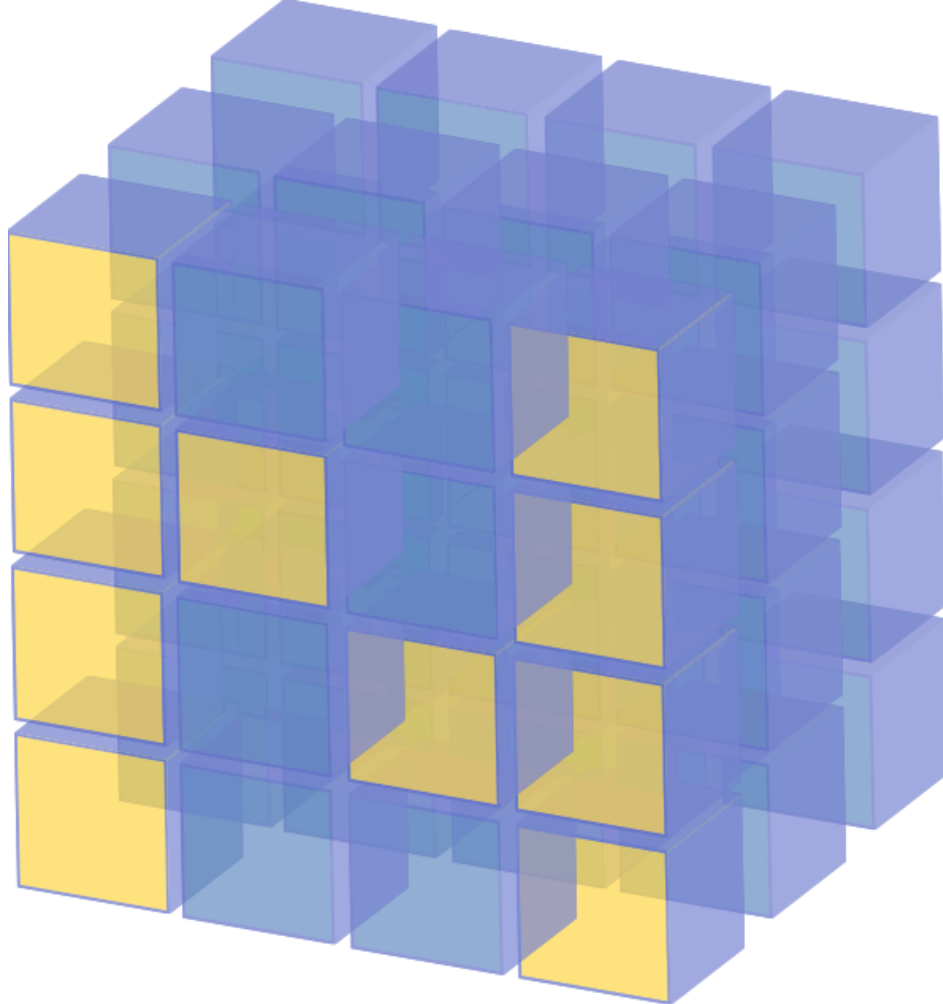
Kurt Smith

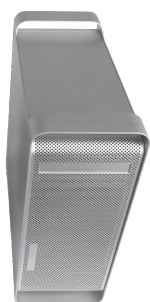
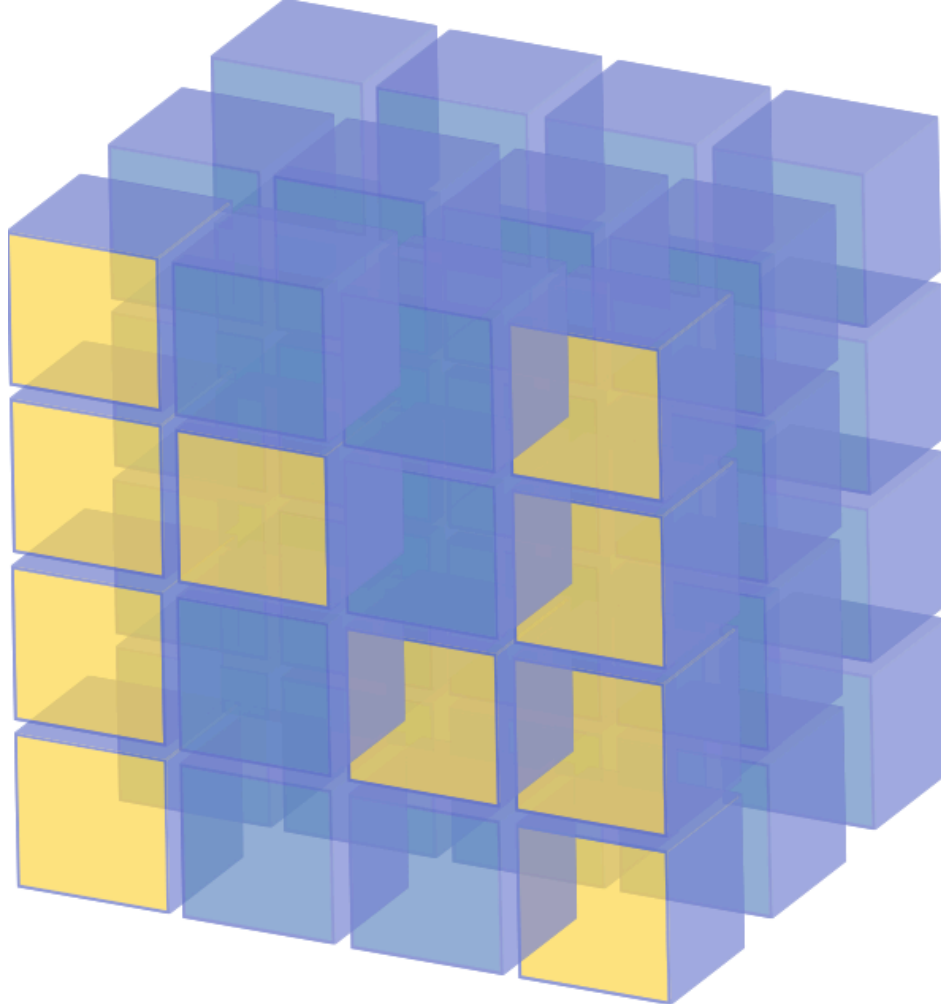
Enthought



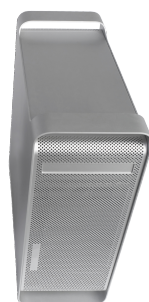


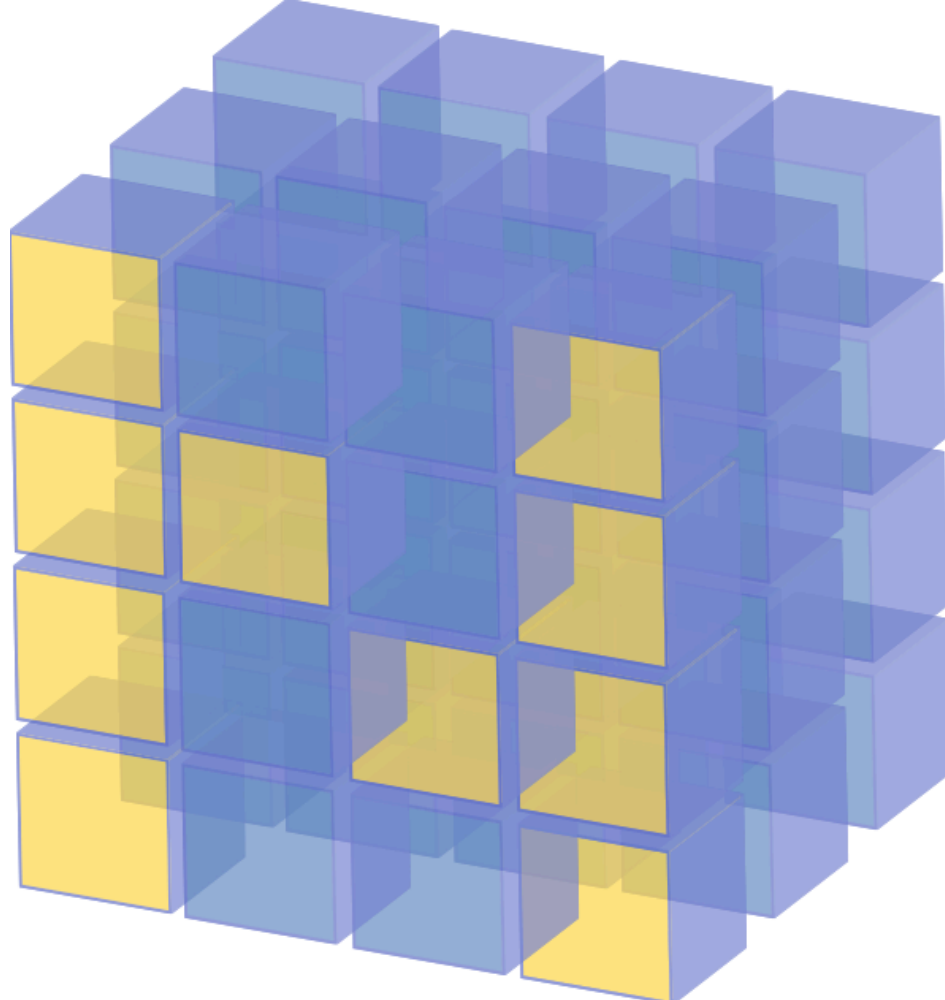


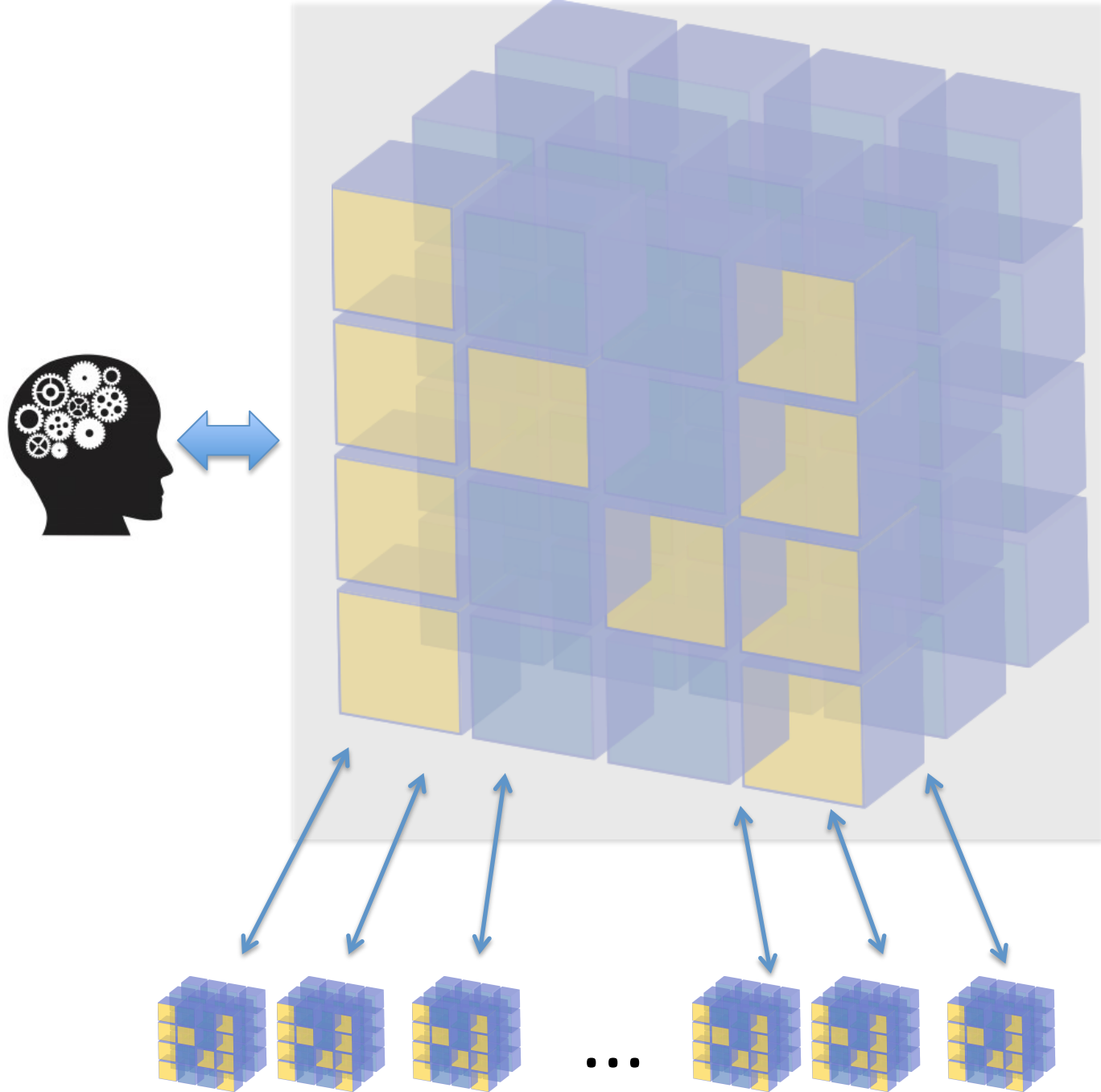




...









Parallel C++

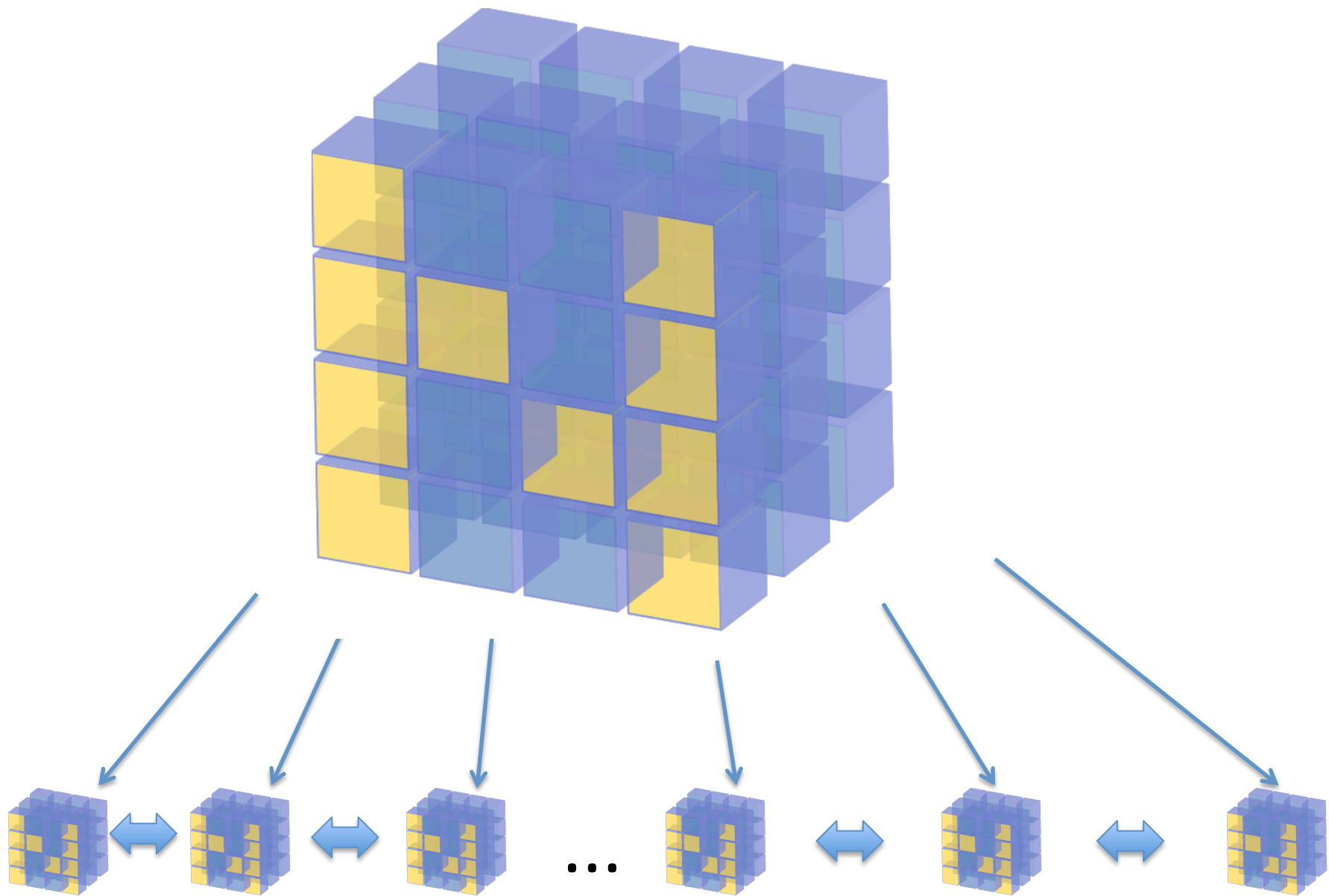
```
for(int i=0; i<n; i++)  
    sum += a[i];  
MPI_AllReduce(&sum, &total_sum,  
              1, MPI_REAL,  
              MPI_SUM, comm);
```

Parallel Python

```
def _basic_reducer(reduce_comm, op, func, args, kwargs, out):  
    """ Handles simple reductions: min, max, sum. Internal. """  
    if out is None:  
        out_ndarray = None  
    else:  
        out_ndarray = out.ndarray  
        if out.ndarray.dtype == np.bool:  
            out.ndarray.dtype = np.uint8  
    local_reduce = np.asarray(func(*args, **kwargs))  
    reduce_comm.Reduce(local_reduce, out_ndarray, op=op, root=0)  
    return out
```

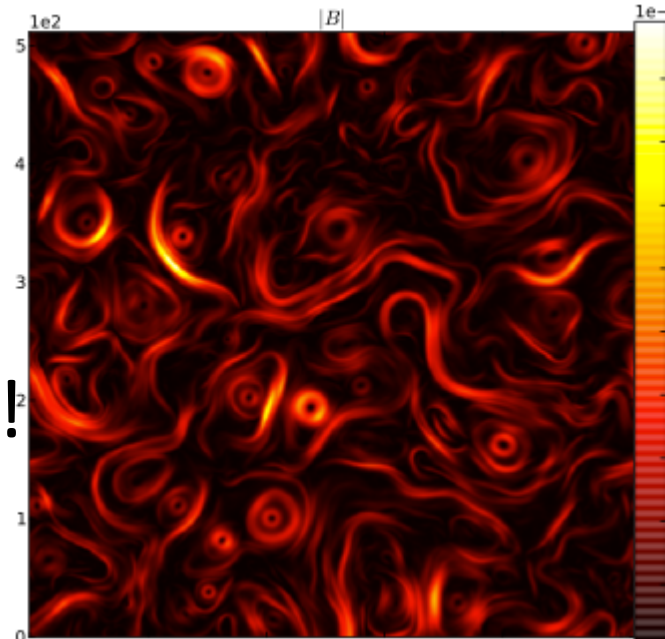
Parallel Fortran

```
do other_pid=1, np-1  
    if(pid .eq. other_pid) then  
        call mpi_send(carr, nelts, MPI_COMPLEX,&  
                      root, tag, MPI_COMM_WORLD, ierr)  
    endif  
    if(pid .eq. root) then  
        call mpi_recv(buffer, nelts, MPI_COMPLEX,&  
                      other_pid, tag, MPI_COMM_WORLD, stat, ierr)  
        if(formatted_flag) then  
            write(OUT_FH_BASE, *) buffer  
        else  
            write(OUT_FH_BASE) buffer  
        endif  
    endif  
enddo
```



Parallel Plasma Turbulence simulation

- Core: 30%
- Auxiliary stuff: 70%
 - Initialization, parallel IO, parallel test suite, etc.
- Convert nearly all auxiliary stuff to idiomatic NumPy with DistArray.
 - Much easier testing!
- Core stays the same.
- Same performance and scaling!



Current features:

- IPython.parallel integration
- Block, cyclic, block-cyclic, unstructured distributions
- Parallel IO – flat files, HDF5
- Slicing, ufuncs, reductions
- user-defined local functions

Coming features:

- Copy-free access to Trilinos packages
- MPI-only communication
- Lazy eval, latency hiding
- Padded arrays

Not just for modeling / simulation

- As flexible and generally useful as NumPy
- Use other upcoming NumPy implementations

Links, contact info

github.com/enthought/distarray

distarray.readthedocs.org

distarray@googlegroups.com



Kurt Smith



Robert Grant



Blake Griffith



Mark Kness

Acknowledgment: This material is based upon work supported by the Department of Energy under Award Number DE-SC0007699.

Disclaimer: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.