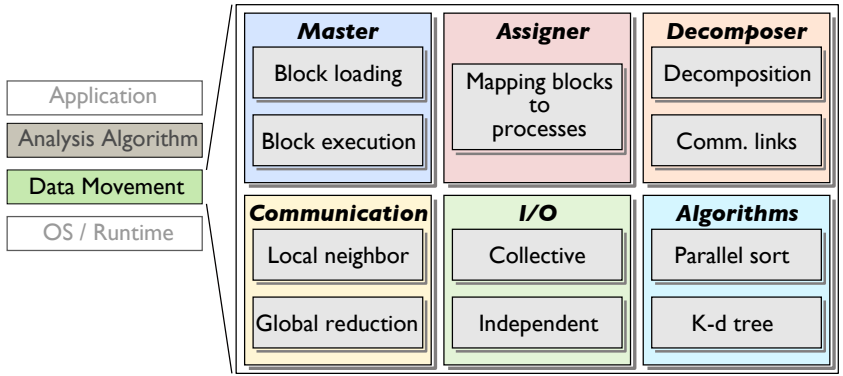
# DIY Block-Parallel Data Analysis




Components of DIY and its place in the software stack are designed to address the data movement challenge in extreme-scale data analysis.


[1] Morozov and Peterka, Block-Parallel Data Analysis with DIY2, LDAV 2016.
[2] Morozov and Peterka, Efficient Delaunay Tessellation through K-D Tree Decomposition, SC16.
[3] Nashed et al., Parallel Ptychographic Reconstruction, Optics Express 2014.



Work was performed at Argonne and Lawrence Berkeley National Labs.


## Scientific Achievement

**DIY is a programming model and runtime for block-parallel analytics on DOE leadership machines; all parallel operations and communications are expressed in terms of blocks, not processors, which enables the same program to run in- and out-of-core with single or multiple threads.**

## Significance and Impact

**DIY enabled Delaunay and Voronoi tessellation of cosmology dark matter particles to 128K processes and improved performance by 50X [2], and it enabled ptychographic phase retrieval of synchrotron X-ray images on 128 GPUs in real time [3]; DIY won an honorable mention paper at LDAV 2016 [1].**

## Research Details

- **Enabling VTK-m by DIY-ing various VTK distributed-memory filters: parallel resampling, multipart dataset redistribution, and stream tracing.**
- **Ongoing preparation for exascale: relaxing synchronization, using deeper memory hierarchy, compatibility with many-core thread models.**

# Parallel Event Generation and Analysis with DIY

## Scientific Achievement

**Fermilab researchers developed two HPC parallel codes using DIY.**

- **Pythia8 Monte Carlo event generator [1]**
- **Feldman-Cousins correction [2]**

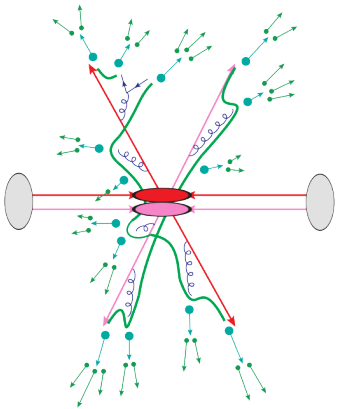
## Significance and Impact

**DIY efficiently utilizes HPC workflows, resources, and HEP community tools.**

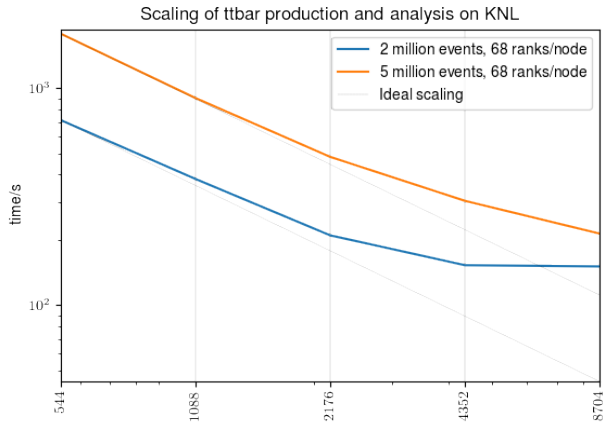
## Research Details

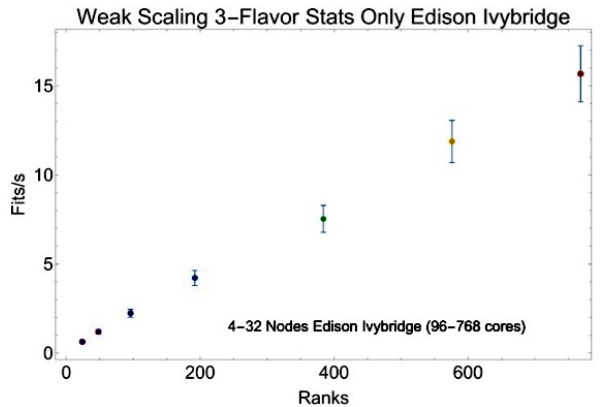
- Allows for extremely short turn-around of large parameter space explorations (e.g. generator tuning)
- Paves the way for new and advanced optimization algorithms, e.g. LHC search analyses.


[1] Buchanan et al., JINST 2020 (in preparation)
[2] Hoche et al., *arXiv 2019*.
[3] Sousa et al., *CHEP 2018*.

**Event generator model for proton-proton collision: Robust predictions of collider events are needed to search for new physics effects. Much of the dynamics is described by tunable parameters. The calculation of event generator predictions is expensive, and must be done for each choice of parameters. A full detector simulation of these calculations is even more expensive, requiring parallel HPC codes.**




**Scalability:** Top: strong scaling of Pythia8 DIY code. Bottom: weak scaling of Feldman-Cousins DIY code.

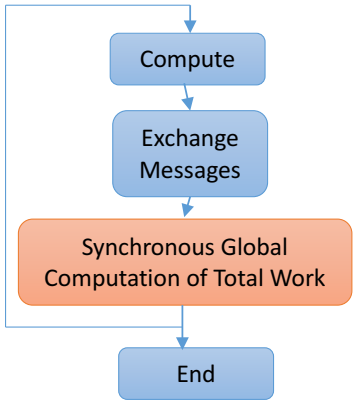

U.S. DEPARTMENT OF ENERGY | Office of Science

Work was performed at Argonne and Fermilab under SciDAC HEP on HPC Partnership

Fermilab | BERKELEY LAB | Argonne NATIONAL LABORATORY | University of CINCINNATI | Colorado State University

# IExchange: Programming

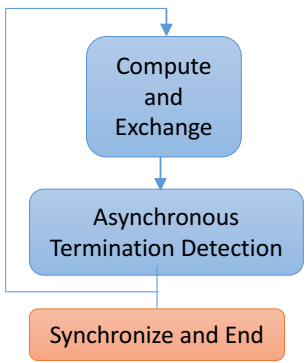
**Old Synchronous Exchange**




```
for (max_rounds) {
   master.foreach(foo);
   master.exchange();
   all_done = reduce(local_work);  // synch. collective
   if (all_done)
      break;
}
```

```
void foo() {
   deque_icoming();
   compute();
   enqueue_outgoing();
}
```

**New Asynchronous IExchange**




```
master.iexchange(bar);
```

```
bool bar() {
   do {
      dequeue_incoming();
      compute();
      enqueue_outgoing();
   } while (fill_incoming());
   return true;
}
```

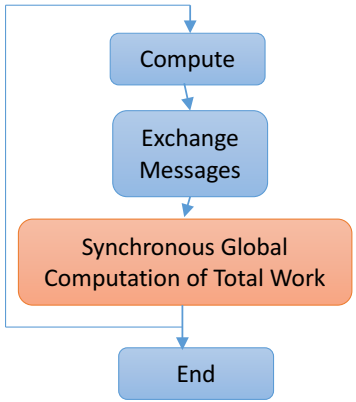


Morozov et al., IExchange: Generic Asynchronous Pattern for Interleaved Computation and Communication, in preparation, 2019.

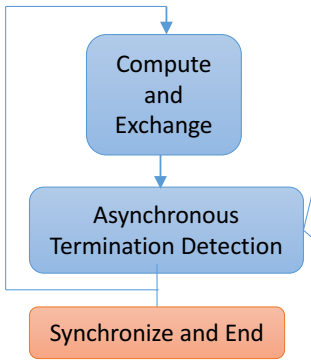
Work was performed at LBNL and ANL

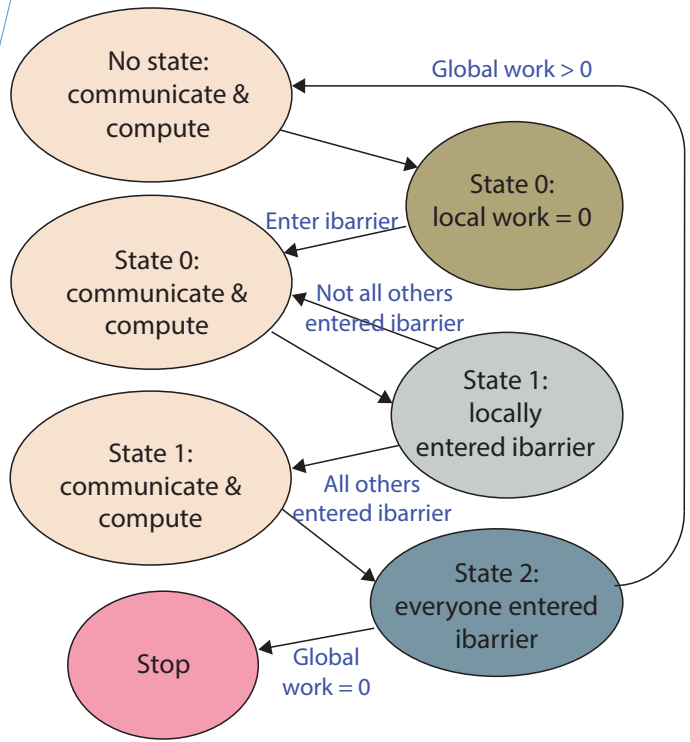
# IExchange: Termination Detection

**Old Synchronous Exchange**




**New Asynchronous IExchange**




**Algorithm 2:** all_done(*local_work*, reference to *state*)

```
1  if state = 0 and local_work = 0 then
2      ibarrier_request = ibarrier
3      dirty = 0
4      state = 1
5      return false
6  if state = 1 then
7      if test(ibarrier_request) = true then
8          iall_reduce_request =
             iall_reduce(all_dirty ← dirty,
             logical_or)
9          state = 2
10         return false
11 if state = 2 then
12     if test(iall_reduce_request) = true then
13         if all_dirty = 0 then                    // done
14             return true
15         else                                     // reset
16             state = 0
17             return false;
```


U.S. DEPARTMENT OF ENERGY | Office of Science

Morozov et al., IExchange: Generic Asynchronous Pattern for Interleaved Computation and Communication, in preparation, 2019.

Work was performed at LBNL and ANL

Argonne NATIONAL LABORATORY | BERKELEY LAB

# IExchange: Asynchronous Communication and Computation in DIY

## Scientific Achievement

**Interleaved asynchronous communication pattern for iterative computations in DIY**

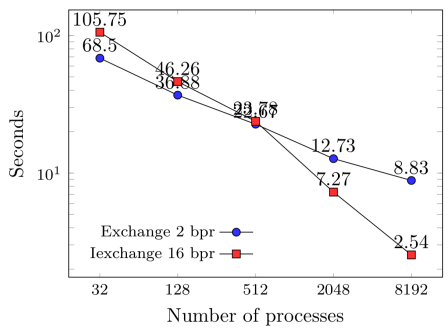
- Eliminates global synchronization on every iteration
- Easier to use: asynchronous communication and termination detection handled by DIY

## Significance and Impact

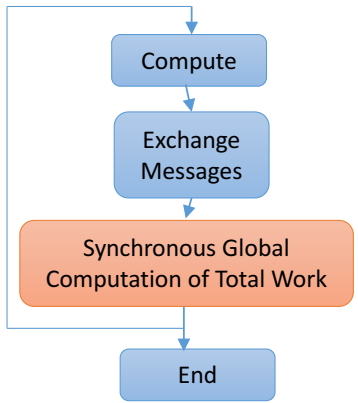
**Irregular imbalanced workloads can be accelerated using IExchange.**

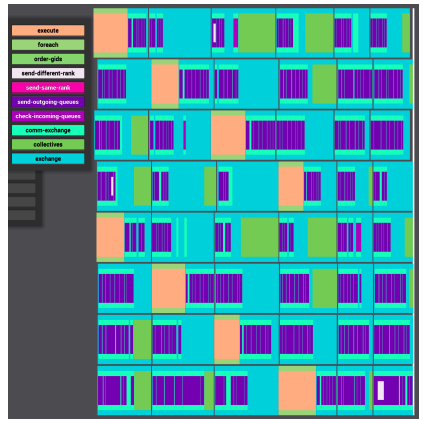
## Research Details

- Asynchronous communication and termination detection, interleaved with computation
- Handles non-monotonic progress and/or unknown amount of global work.

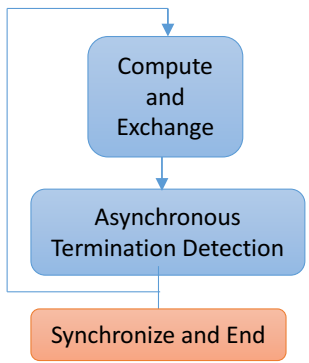


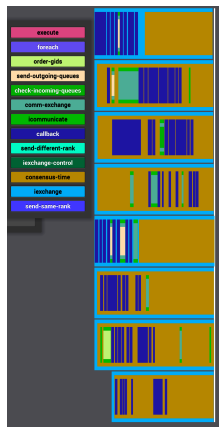

**Scalability:** strong scaling plot shows iexchange up 3.5X faster and 5.4X better efficiency than exchange for particle tracing in Nek5000 thermal hydraulics application.



**Old Synchronous Exchange**



**New Asynchronous IExchange**


U.S. DEPARTMENT OF ENERGY | Office of Science

Morozov et al., IExchange: Generic Asynchronous Pattern for Interleaved Computation and Communication, in preparation, 2019.

Work was performed at LBNL and ANL

Argonne NATIONAL LABORATORY

BERKELEY LAB