

What is Tasking?

- Define units of serial work (tasks) of an application and dependencies among the tasks
- Problem-centric decomposition vs. data-centric (SPMD)

On-Node

- Tasks do not migrate across nodes
- Tasks coordinate with MPI to communicate across nodes
- Further decompose the domain of an MPI rank and assign tasks to the second-level decomposition

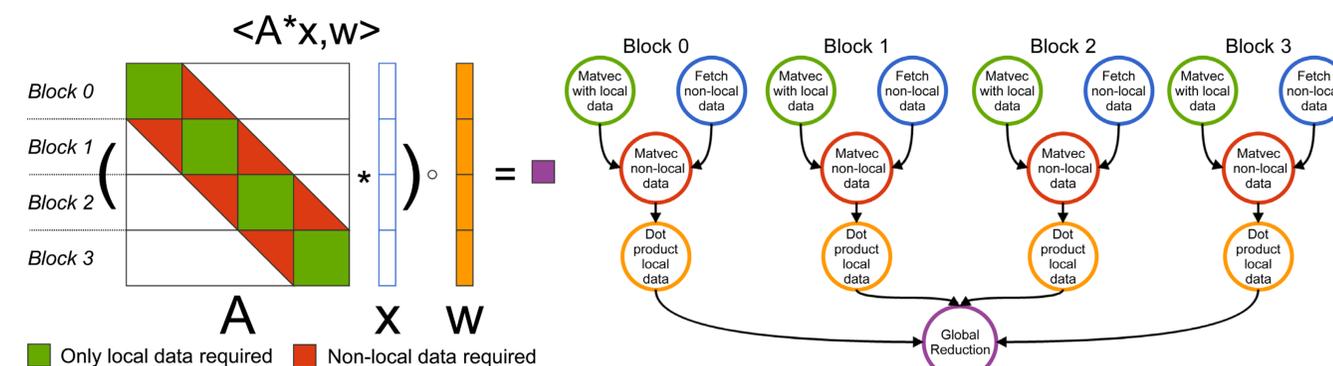
Why use tasks?

- Separates exposing parallelism (programmer) and mapping parallelism onto the architecture (runtime)
- Transparently adjust to the level of concurrency on the node, including heterogeneous accelerators
- Load balancing is natural by migrating tasks

Primary Challenges

- How to express and submit tasks?
 - fork/join
 - Dynamic Task Graph
- How to encode dependencies?
 - Dataflow: inferred from task access mode to data
 - Control flow: barriers, sync, futures
- How to schedule tasks onto the hardware efficiently?

Modern Runtimes Example



OpenMP^[1]

```
#pragma omp parallel{
#pragma omp master{
for(int i = 0; i < num_blocks; ++i){
start = i * block_size;

#pragma omp task depend(out:x_externals[i])
fetch_externals(A_external[i], x_externals[i]);

#pragma omp task depend(out: y[start:block_size])
local_matvec(A_local[i], x_local[i], &y[start]);

#pragma omp task depend(in: y[start:size],x_externals[i])
matvec_with_externals(A_external[i], x_externals[i]);

#pragma omp task depend(in:y[start:size]
\ depend(out: local_sums[i])
local_dot_product(&y[start], &w[start], &local_sums[i]);
}
#pragma omp task depend(in: local_sums[0:num_blocks])
global_reduction(local_sums[]);
}
}
```

Cilk Plus^[2]

```
block_function(int i){
cilk_spawn fetch_externals(A_external[i], x_external[i]);
cilk_spawn local_matvec(A_local[i], x_local[i], &y[start]);
cilk_sync;
cilk_spawn matvec_with_externals(A_external[i],
x_externals[i]);
cilk_sync;
local_dot_product(&y[start], &w[start], &local_sums[i]);
//implicit sync
}

cilk_for (int i = 0; i < num_blocks; ++i){
block_function(i);
}
cilk_sync;
global_reduction(local_sums[]);
```

miniFutures: A new task-based mini-application

- Proxy for unstructured finite element method
 - Discretization, element analysis, assembly
 - Pre-conditioned conjugate gradient solver
 - Gauss-Seidel pre-conditioner
- Runtime agnostic
 - Goal is to evaluate modern runtimes
 - Study how to design future task-based applications