Unal Center of Education Research and Development
www.ucerd.com

Parallel Program Design

Dr. Tassadaq Hussain

RIPHAH INTERNATIONAL UNIVERSITY

BSC Microsoft Research Centre

- Applications Types
- PCAM Methodology
- Decomposition Patterns
- Program Structure

Applications

Compute Intensive
Data Intensive
Complex and Irregular

C and C++ Applications

http://people.sc.fsu.edu/~jburkardt/cpp_src/cpp_src.html

http://people.sc.fsu.edu/~jburkardt/c_src/c_src.html

PCAM Methodology

Partitioning

Communication

Agglomeration

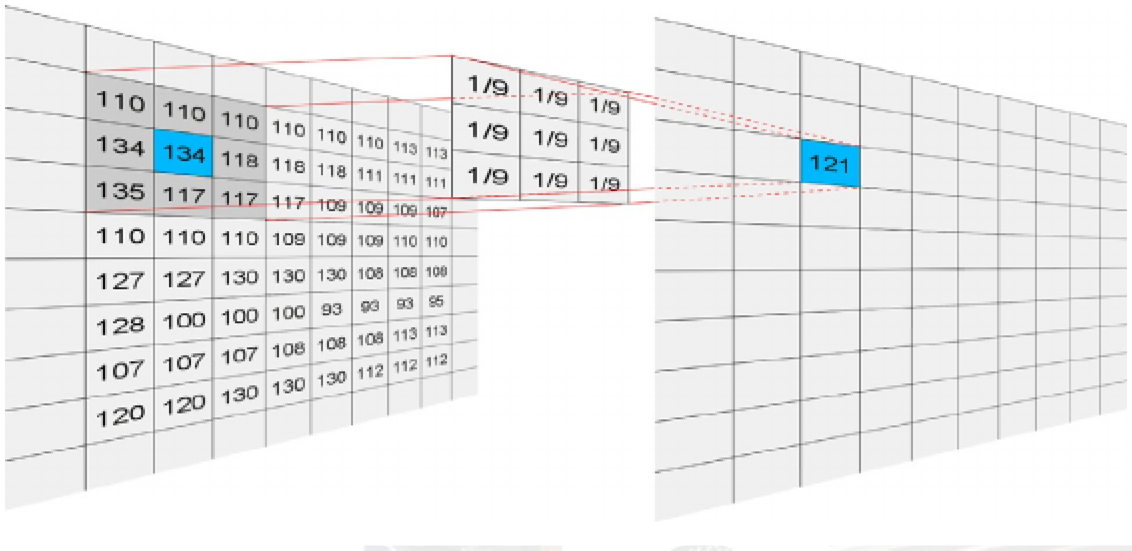
Mapping

Application Understanding

Metamathematical Representation

$$g(x,y) = \sum_{i=-n2}^{n2} \sum_{j=-n2}^{n2} k(n2+i, n2+j) f(x-i, y-j)$$

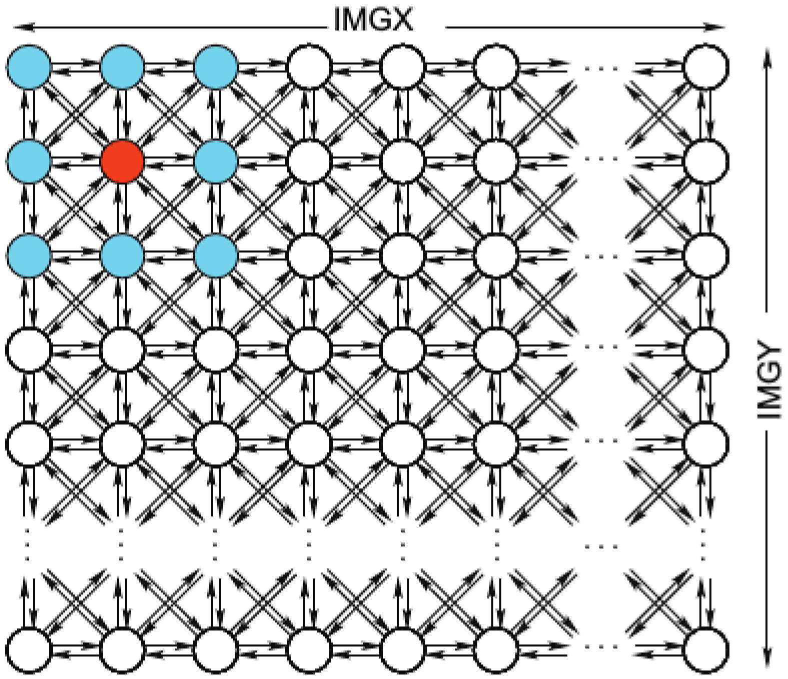
Working Operation



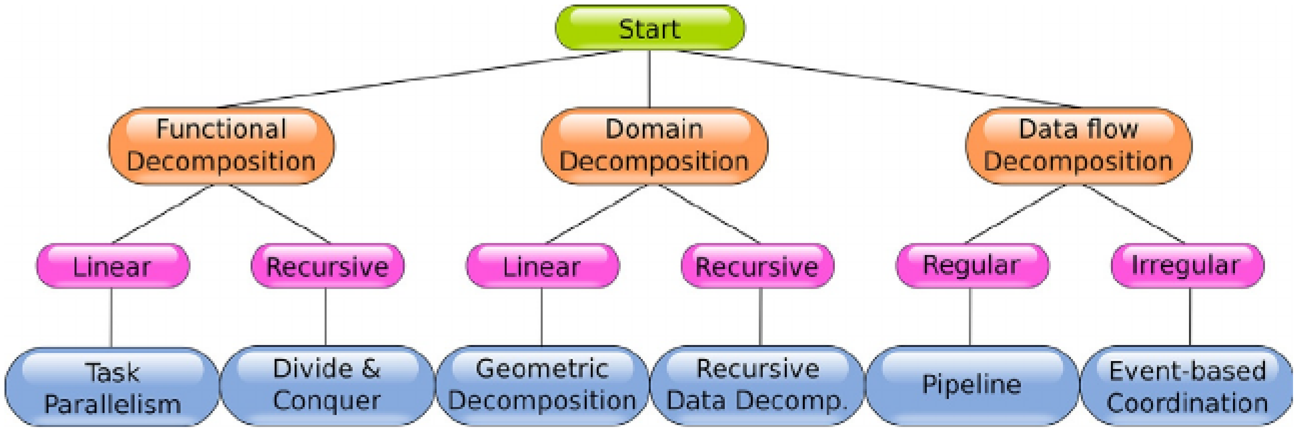
Pseudocode

```
int img[IMGY+2][IMGX+2];
int filt[IMGY][IMGX];
int n2 = n/2;
for(int x=1;x <= IMGX; x++) {
    for(int y=1; y <= IMGY ; y++)  {
        int newV=0;
        for(int i= -n2; i<= n2; i++)
            for(int j= -n2; j<= n2; j++)
                newV += img[ y - j][ x - i ] * k[n2 + j][n2 + i];
        filt[y-1][x-1] = newV;
        }
    }
```

Decomposing Application



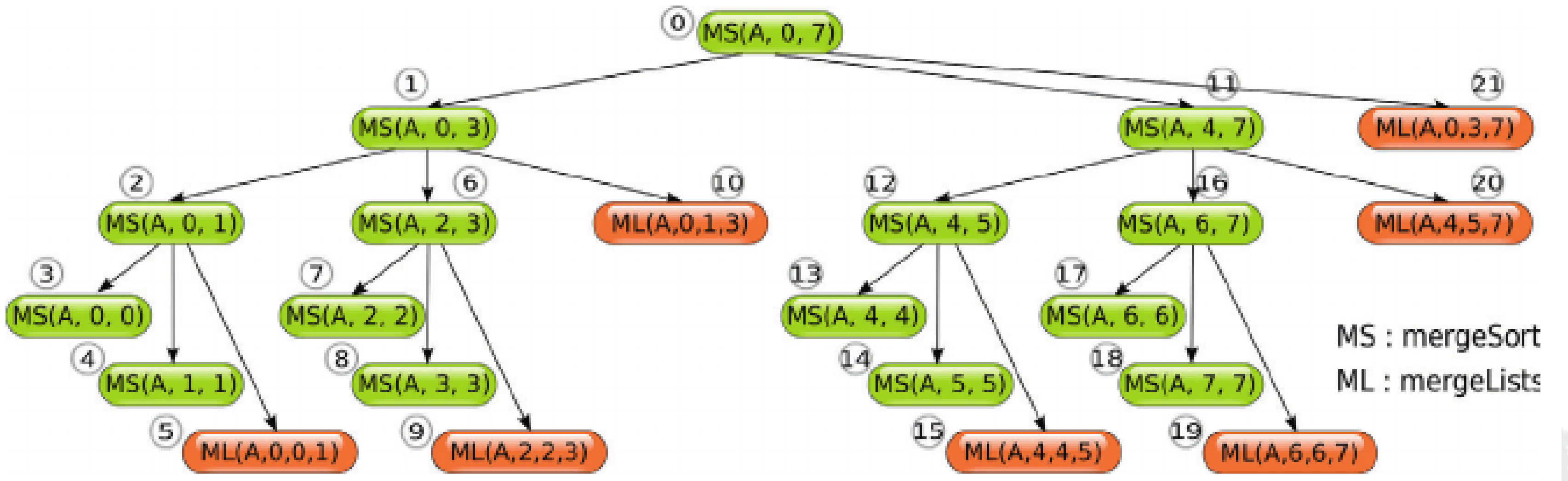
Decomposition

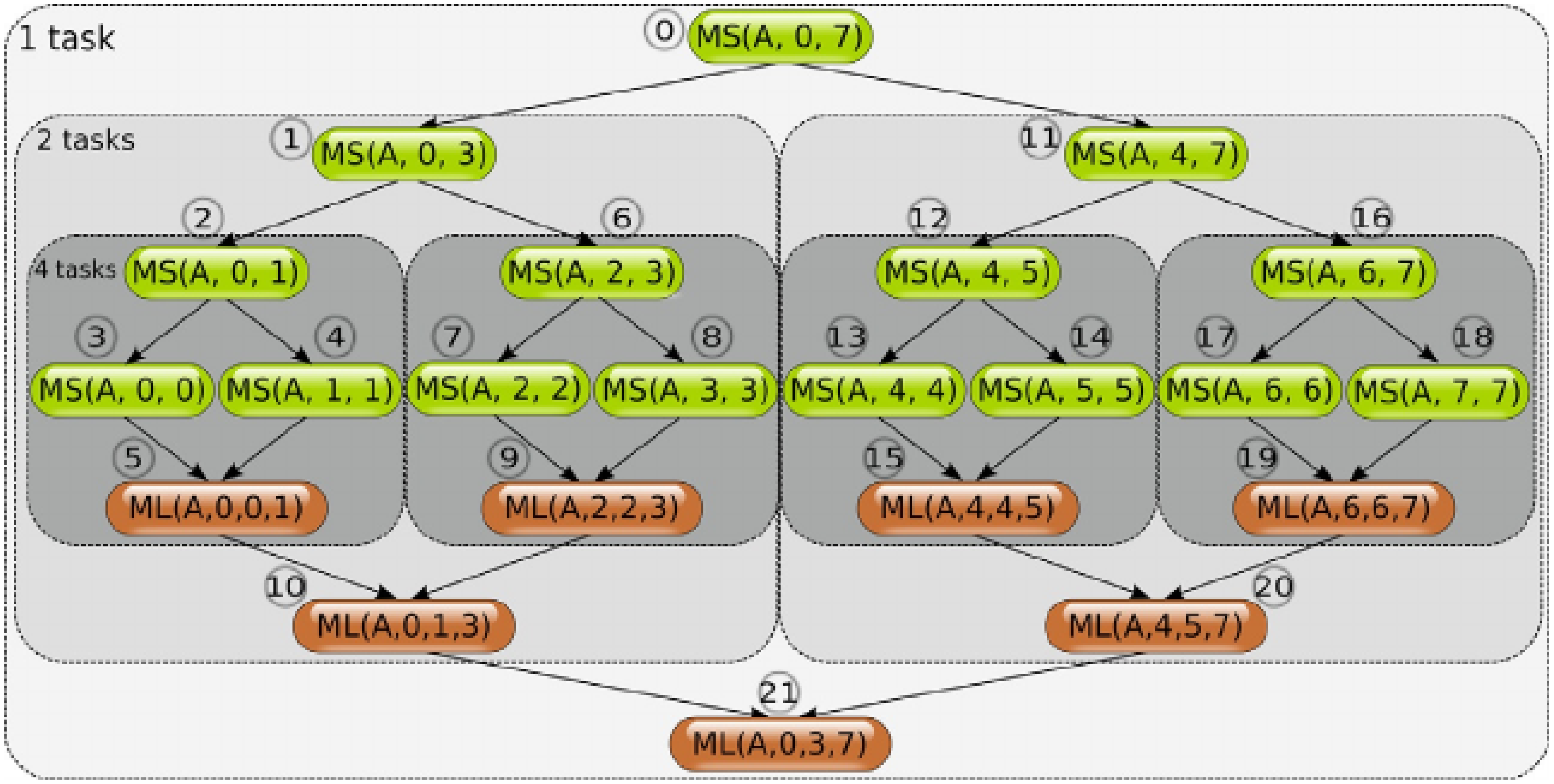


Types of Decomposition

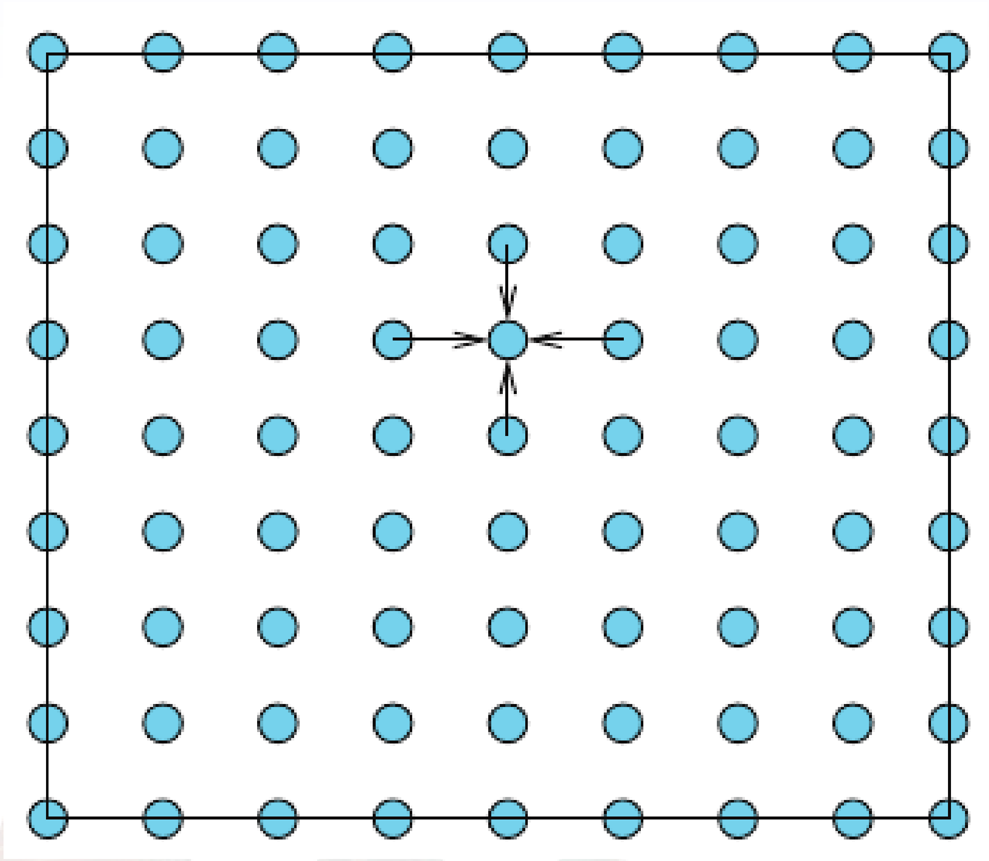
- Functional Decomposition
  - Task Parallelism
  - Divide & Conquer
- Domain Decomposition
  - Geometric
  - Recursive Data
- Data Flow Decomposition
  - Pipelining
  - Event Based

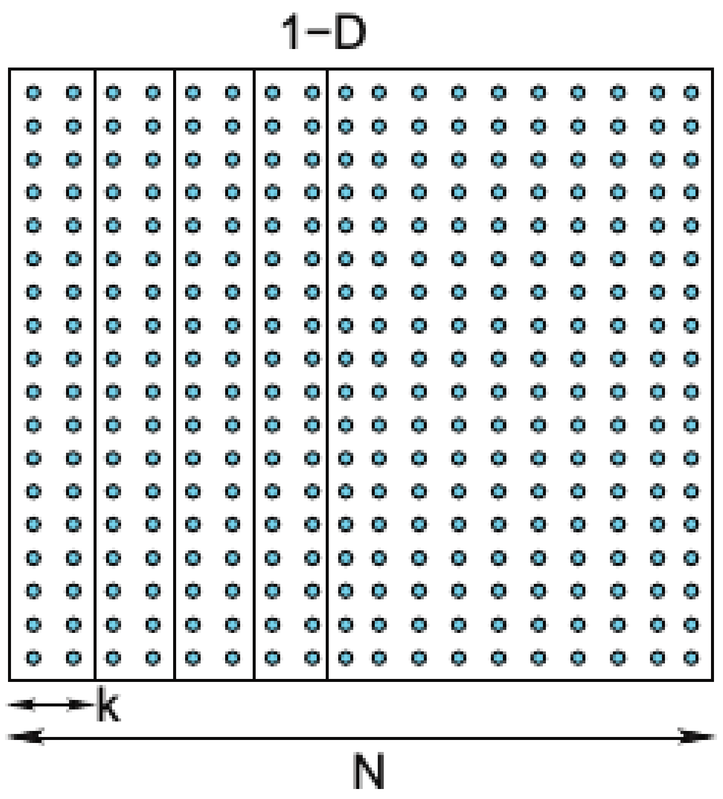
Divide & Conquer

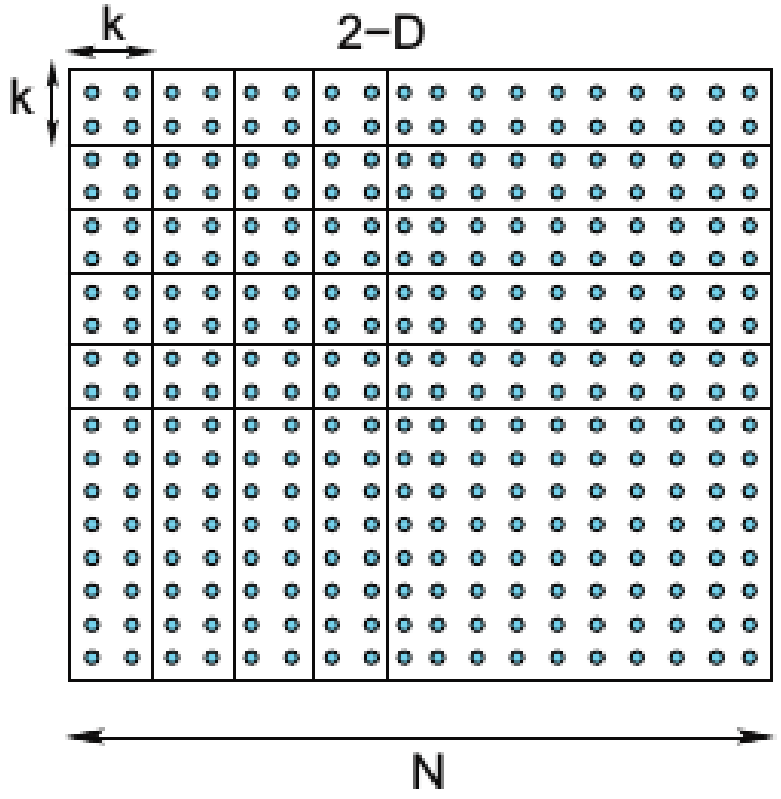



1 task
2 tasks
4 tasks
0
MS(A, 0, 7)
1
MS(A, 0, 3)
11
MS(A, 4, 7)
2
MS(A, 0, 1)
6
MS(A, 2, 3)
12
MS(A, 4, 5)
16
MS(A, 6, 7)
3
MS(A, 0, 0)
4
MS(A, 1, 1)
7
MS(A, 2, 2)
8
MS(A, 3, 3)
13
MS(A, 4, 4)
14
MS(A, 5, 5)
17
MS(A, 6, 6)
18
MS(A, 7, 7)
5
ML(A,0,0,1)
9
ML(A,2,2,3)
15
ML(A,4,4,5)
19
ML(A,6,6,7)
10
ML(A,0,1,3)
20
ML(A,4,5,7)
21
ML(A,0,3,7)

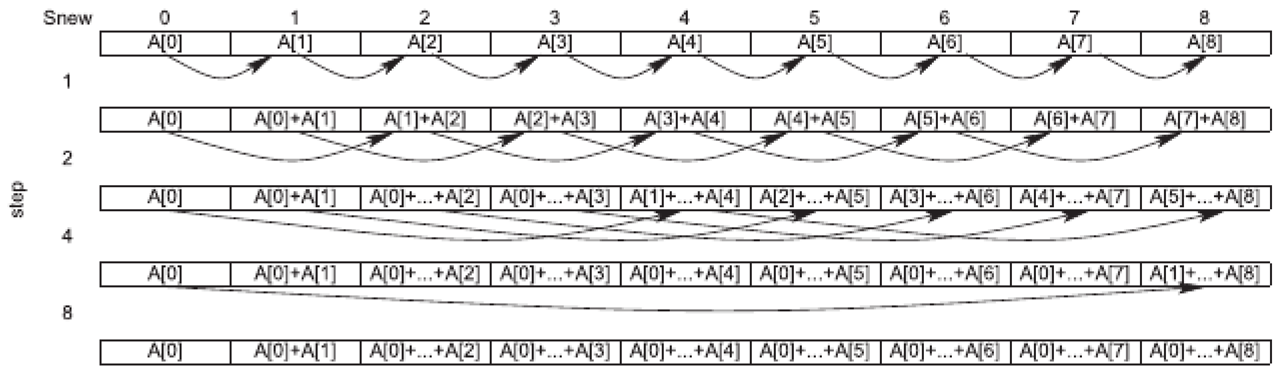
Geometry Decomposition




1−D
k
N

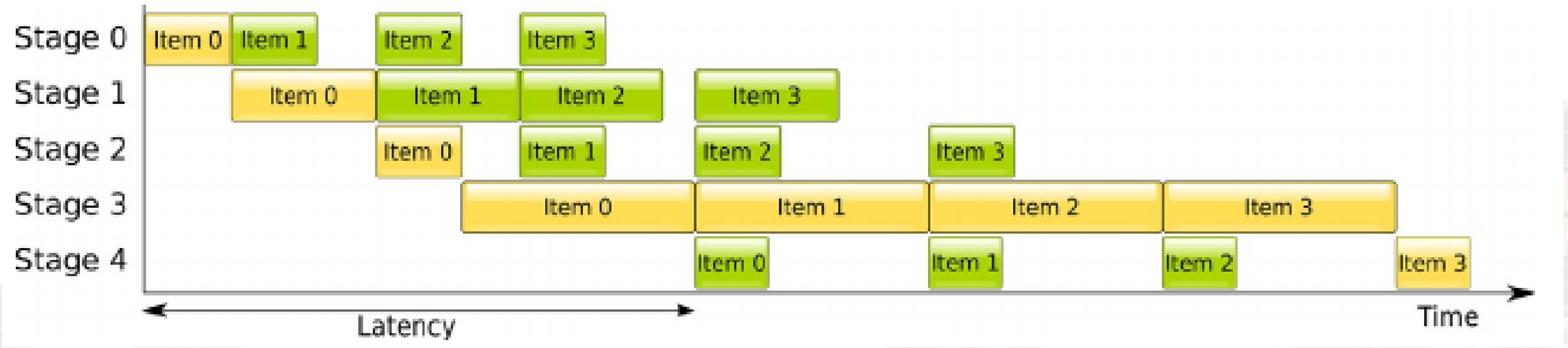

k
2−D
k
N

Recursive Domain Decomposition


Snew
0 1 2 3 4 5 6 7 8
A[0] A[1] A[2] A[3] A[4] A[5] A[6] A[7] A[8]
step
1
A[0] A[0]+A[1] A[1]+A[2] A[2]+A[3] A[3]+A[4] A[4]+A[5] A[5]+A[6] A[6]+A[7] A[7]+A[8]
2
A[0] A[0]+A[1] A[0]+...+A[2] A[0]+...+A[3] A[1]+...+A[4] A[2]+...+A[5] A[3]+...+A[6] A[4]+...+A[7] A[5]+...+A[8]
4
A[0] A[0]+A[1] A[0]+...+A[2] A[0]+...+A[3] A[0]+...+A[4] A[0]+...+A[5] A[0]+...+A[6] A[0]+...+A[7] A[1]+...+A[8]
8
A[0] A[0]+A[1] A[0]+...+A[2] A[0]+...+A[3] A[0]+...+A[4] A[0]+...+A[5] A[0]+...+A[6] A[0]+...+A[7] A[0]+...+A[8]

Pipeline Decomposition

Depends on Processor Architecture



Event Based

An event is a time-stamped message that can represent a status change in the state of a module, a trigger to change the state, a request to perform an action, a response to a previously generated request, or the like.

PROGRAM STRUCTURE

Globally Parallel, Locally Sequential (GPLS):

GPLS means that the application is able to perform multiple tasks concurrently, with each task running sequentially.

Patterns that fall in to this category include:

- Single program, multiple data
- Multiple program, multiple data
- Master-worker
- Map-reduce

Globally Sequential, Locally Parallel (GSLP):

GSLP means that the application executes as a sequential program, with individual parts of it running in parallel when requested.

Patterns that fall in to this category include:

- Fork/join
- Loop parallelism

SINGLE-PROGRAM, MULTIPLE-DATA

- **Program initialization:** This step usually involves deploying the program to the parallel platform and initializing the run-time system responsible for allowing the multiple threads or processes to communicate and synchronize.
- **Obtaining a unique identifier:** Identifiers are typically numbered starting from 0, enumerating the threads or processes used. In certain cases, the identifier can be a vector and not just a scalar (e.g., CUDA). Identifier lifetime follows the thread or process lifetime it corresponds to. Identifiers can be also persistent, i.e., exist for the duration of the program, or they can be generated dynamically whenever they are needed.
- **Running the program:** Following the execution path corresponding to the unique ID. This could involve workload or data distribution, diversification of roles, etc.
- **Shutting down the program:** Shutting down the threads or processes, possibly combining the partial results generated into the final answer.

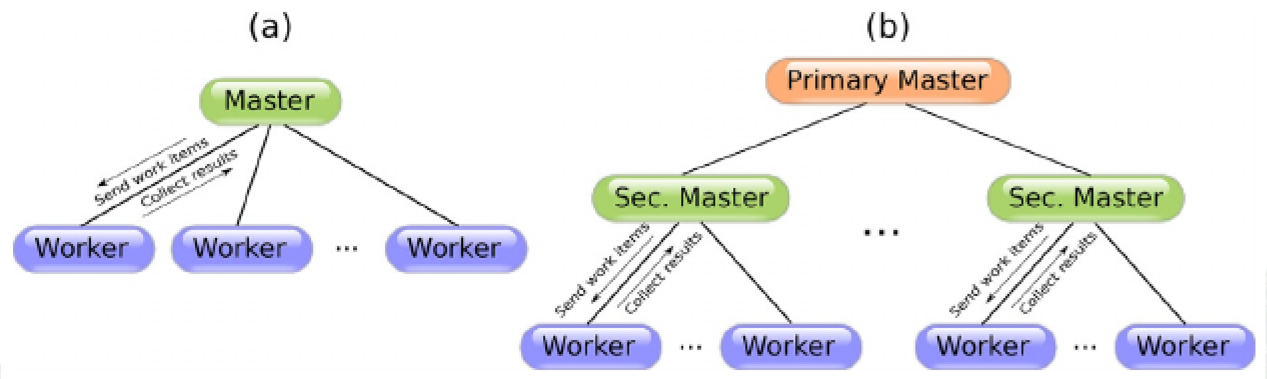
Multi Program Multi Data

The execution platform is heterogeneous, mandating the need to deploy different executables based on the nodes’ architecture.

The memory requirements of the application are so severe that memory space economy dictates a reduction of the program logic uploaded to each node to the bare essentials.

MASTER-WORKER

Handing out pieces of work to workers Collecting the results of the computations from the workers Performing I/O duties on behalf of the workers, i.e., sending them the data that they are supposed to process, or accessing a file Interacting with the user



MAP REDUCE

The map-reduce pattern, works in its generic form. A user program spawns a master process that oversees that whole procedure.

A number of workers are also spawned; they are responsible for

(a) processing the input data and producing intermediate results.

(b) combining the results to produce the final answer.

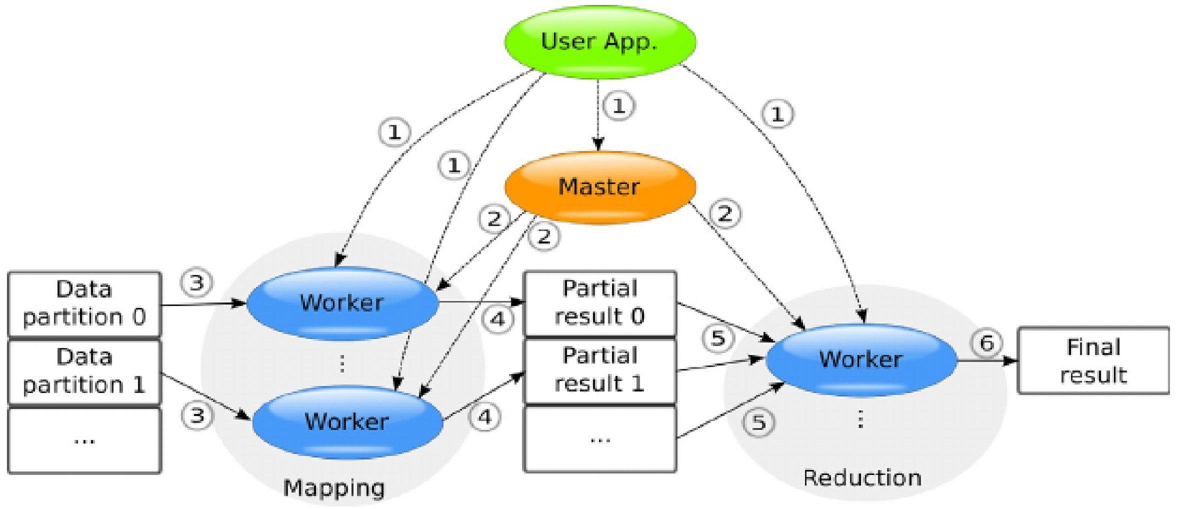


FIGURE 2.14

Generic form of the map-reduce pattern. The steps involve (1) spawning of the master and worker threads, (2) task assignments by the master, (3) data input by the workers performing the mapping, (4) saving of the partial results, (5) reading of the partial results from the "reducing" workers, and (6) saving the final result.

FORK/JOIN

The fork/join pattern is employed when the parallel algorithm calls for the dynamic creation (forking) of tasks at run-time. These children tasks (processes or threads) typically have to terminate (join) before the parent process/thread can resume execution.

LOOP PARALLELISM

This pattern is particularly important for the OpenMP platform, where the loops are semiautomatically parallelized with the assistance of the programmer.

The programmer has to provide hints in the form of directives to assist with this task.

Task

- Select an scientific HPC application
  - Compute Intensive
  - Data Intensive
  - Complex

- Write its **equation** and **working principle**
- Draw its **CDFG** based on **Decomposition Pattern**
- Propose a **Program Structure Pattern**