Document Classification

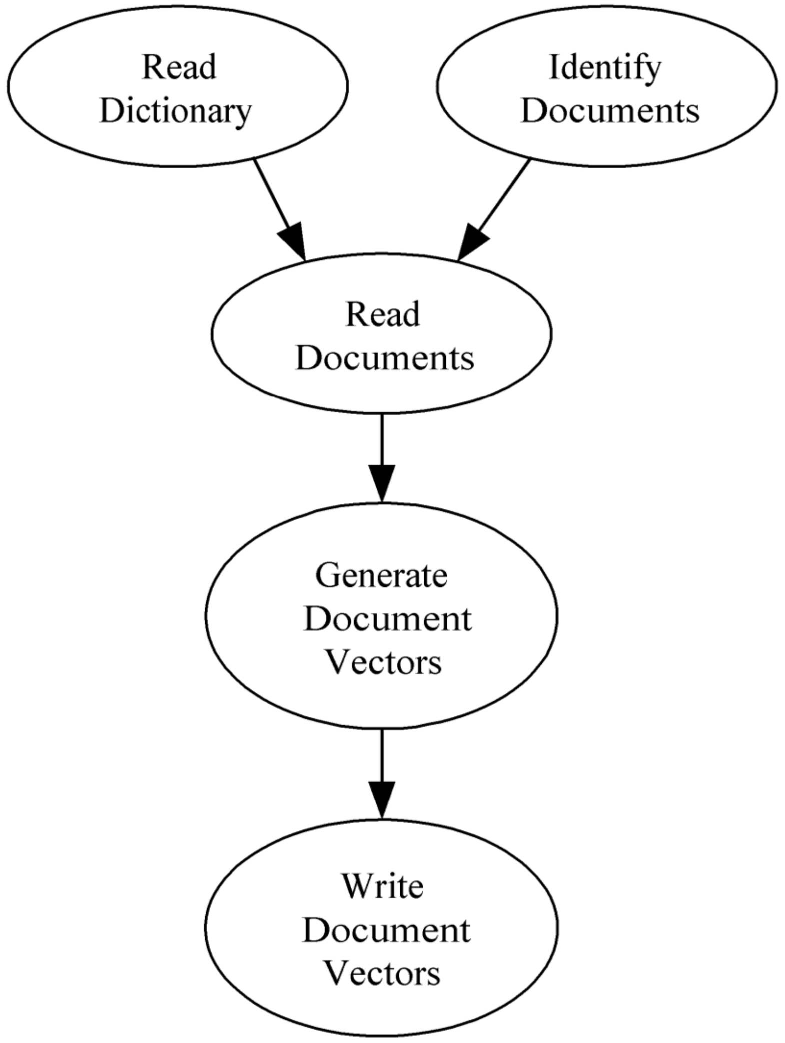
Objectives

- Search documents on WWW to find relevant information
- Implement manager-worker parallel model

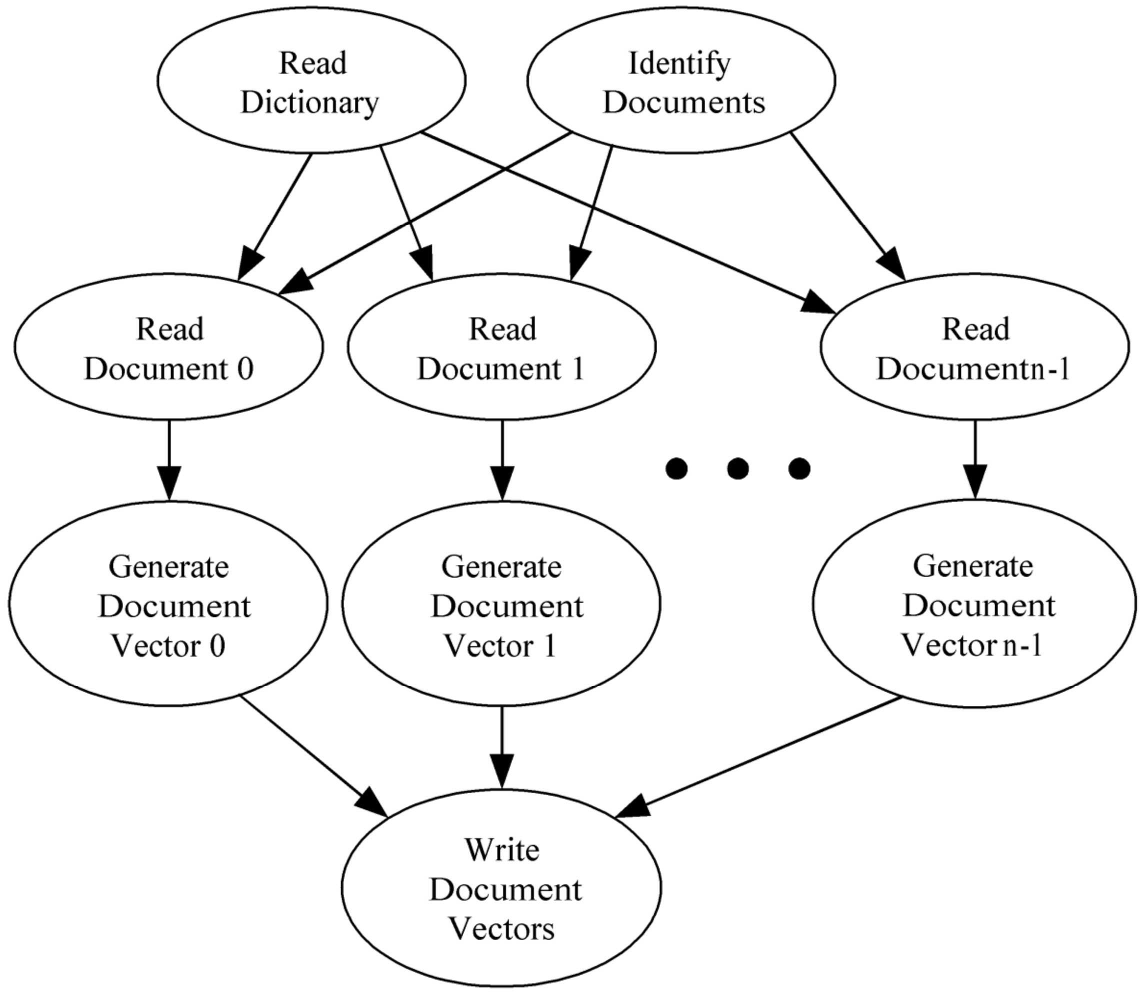
Document Classification Problem

- Search directories, subdirectories for documents (look for .html, .txt, .tex, etc.)
- Using a dictionary of key words, create a profile vector for each document
- Store profile vectors

Task Dependence Graph and Parallelization



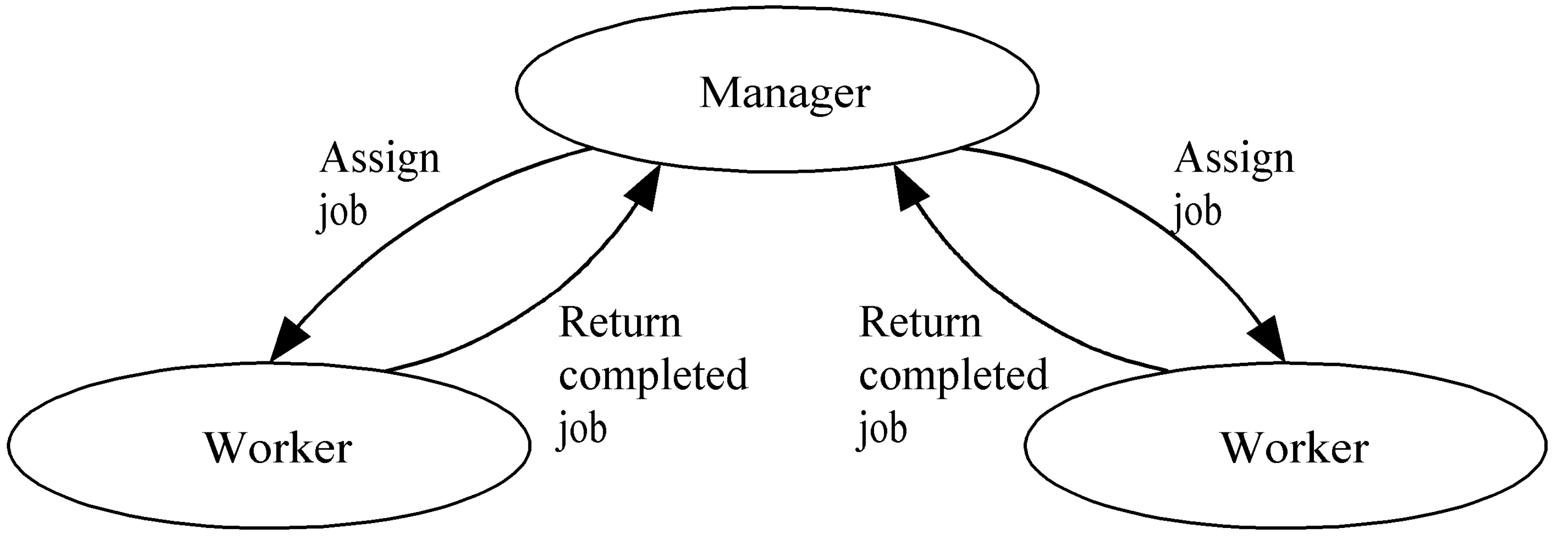
1. Most time spent reading documents and generating profile vectors
2. Create two primitive tasks for each document



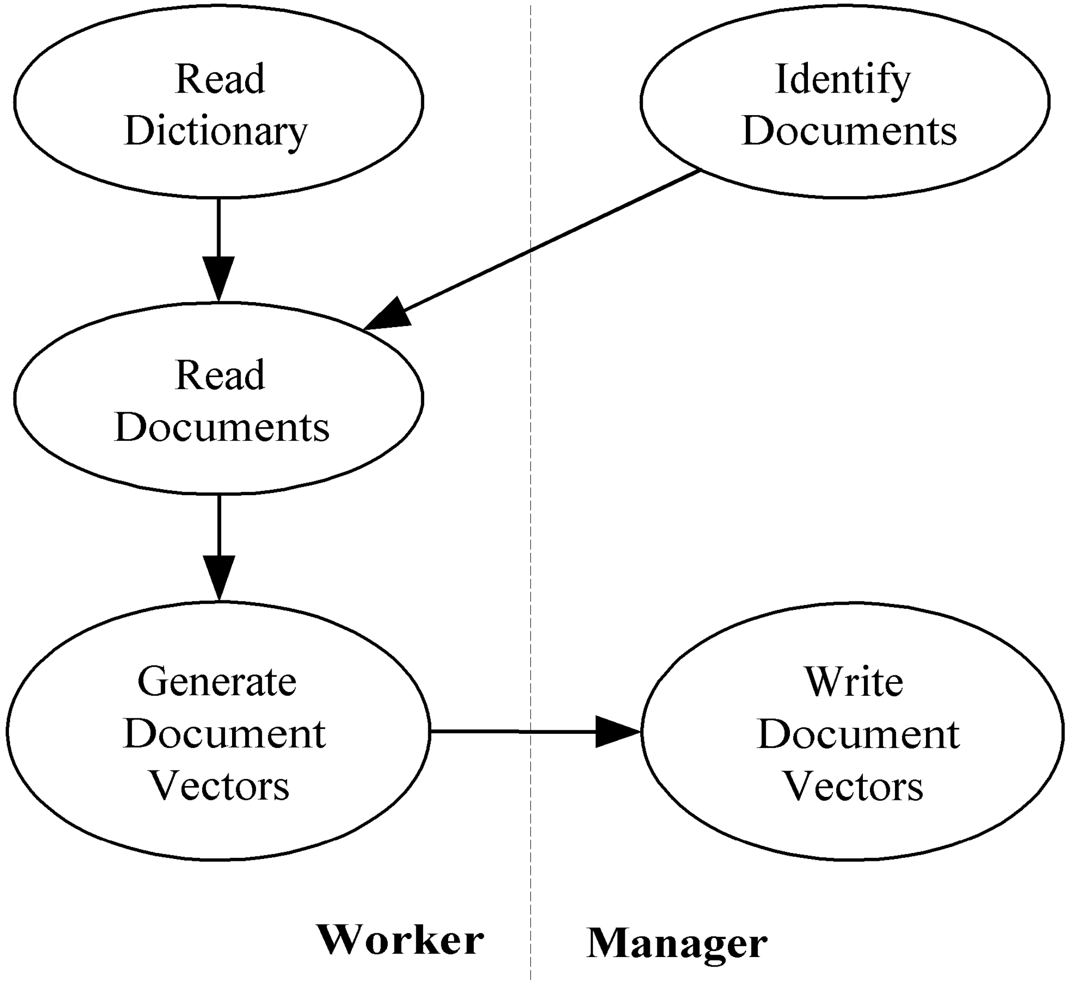
Reading and profiling of each document may occur in parallel

1. Number of tasks not known at compile time
2. Tasks do not communicate with each other
3. Time needed to perform tasks varies widely
4. Strategy: map tasks to processes at run time

Manager/worker Model



Roles of Manager and Workers



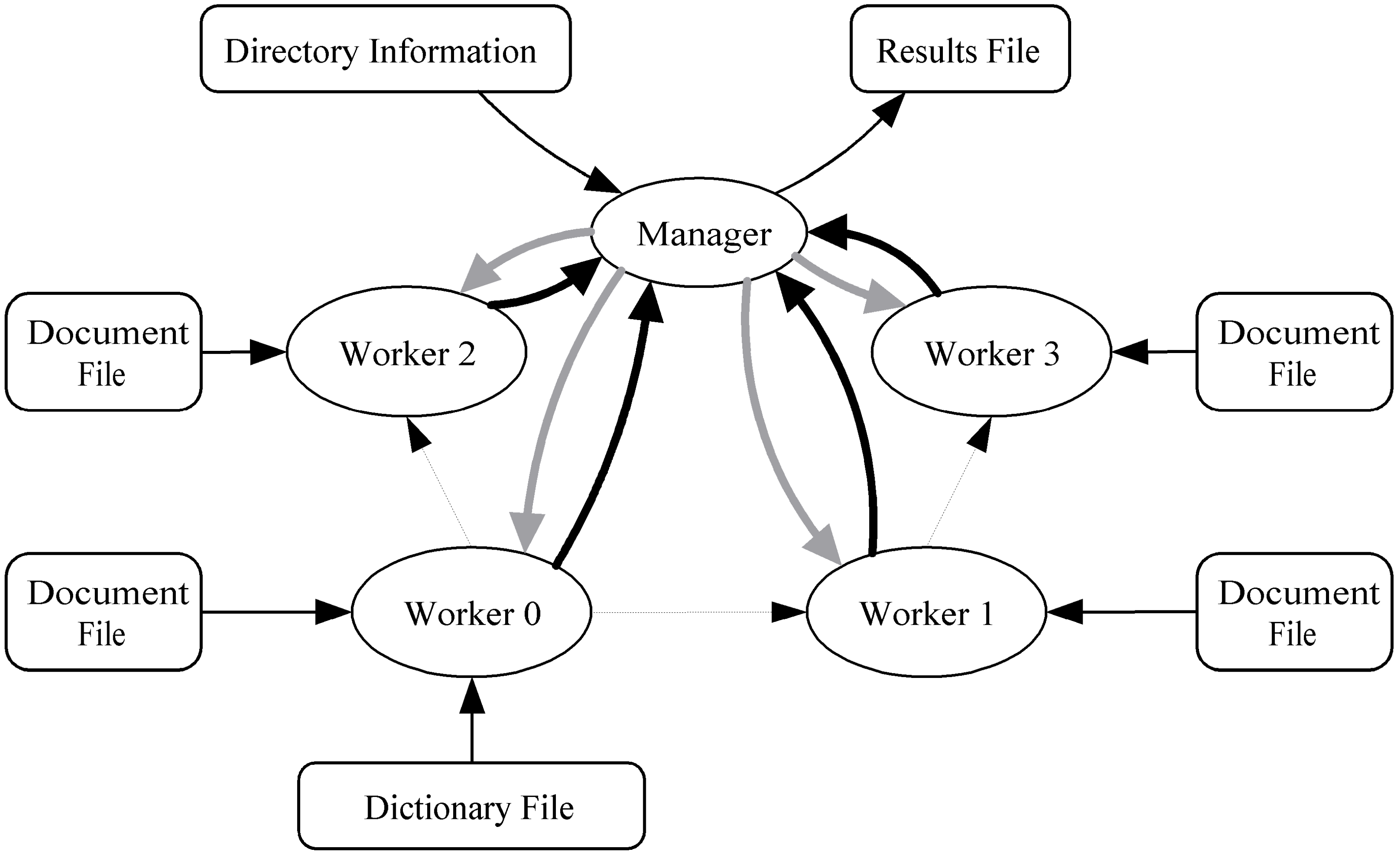
Manager Pseudocode

```
Identify documents
Receive dictionary size from worker 0
Allocate matrix to store document vectors
repeat
    Receive message from worker
    if message contains document vector
        Store document vector
    endif
    if documents remain then Send worker file name
    else Send worker termination message
    endif
until all workers terminated
Write document vectors to file
```

Worker Pseudocode

```
Send first request for work to manager
if worker 0 then
    Read dictionary from file
endif
Broadcast dictionary among workers
Build hash table from dictionary
if worker 0 then
    Send dictionary size to manager
endif
repeat
    Receive file name from manager
    if file name is NULL then terminate
    endif
    Read document, generate document
vector
    Send document vector to manager
forever
```

Task/Channel Graph



Creating a Workers-only Communicator

1. Dictionary is broadcast among workers
2. To support workers-only broadcast, need workers-only communicator
3. Can use MPI_Comm_split
4. Manager passes MPI_UNDEFINED as the value of split_key, meaning it will not be part of any new communicator

Workers-only Communicator

```
int      id;
MPI_Comm worker_comm;

if (!id) // manager
   MPI_Comm_split (MPI_COMM_WORLD,
      MPI_UNDEFINED, id, &worker_comm);

else // worker
   MPI_Comm_split (MPI_COMM_WORLD, 0,
      id, &worker_comm);
```

Expected Things

1. Pseudo code describing the parallel algorithm
2. Justification of choosed communication mode (block/non-block ?)
3. Performance table.

Reference:

W. Barry and M. Allen. Parallel Programming: Technique and Applications Using Networked Workstations and Parallel Computers. Upper Saddle River, NJ: Prentice-Hall, 1999.

Variations

- For undergraduate students

Develop a master/worker parallel program that find the smallest positive root of the equation:

$$f(x) = -2 + \sin(x) + sin^2(x) + sin^3(x) + \cdots + sin^{1000}(x).$$

The root r is the unique value between 0 and 1. The program should divide [0, 1] into several subintervals and create a set of tasks, one for each subinterval.