17 Lecture

The implementation of a pipelined algorithm can be:
→ Template based
→ Non-Template based

Template based algs, recognized many patterns, that are used and have to use this template, and fill the missing parameters.

Example: pipeline (...)

Now we have a template, I can have different possibilities to implement the channels (Pipes, FIFO, ...) Also, for I same implementation of the template, I need the parameters.

Within your file the parameters you have an implementation.

T_1 = map(T_2) — the pipe stages or a new Component

form(f, map) — the pipe

Parallel queues are one template.

If I want to implement the form of pipelines I have to map the the templates

form(pipe(f1,f2))

the number of arrows in the table pipeline

→ indicates the stage in a pipeline and that the number indicates a different work pipe parameters

T_1 = form(pipe(f1,f2)) — T_2 = pipe(f1,f2)

T_1 table pipeline, the pipe is transformed into the number of stages in the parallel queue.

T_1 T_2 T_3 T_4

the number of stages depend on the table pipe

MACRO DATA FLOW BASED:

Use tools, we need a special activity graph, a graph of dependencies.

We decide to get some optimizations. I define loops for (pipe(f1,f2)) and create the graph of dependency.

In each task I need to introduce a code that compute the running function.

form(pipe(f1,f2)), n=4

I could have 3 copies of the graph with f and g inside, each of them.

f, g are separate because they dependent on the data

f(i) — g(i) — dependency of the data (i.e. f(i) depends on g(i-1))

g(i) — f(i+1) — Pipe stages, compute at the same time

Now I implement this:

Dependency of TASK instructions

each thread has the code, while (true) { get (provide) function code (update) variables }

Implementation of single instruction

- pipe = (x, y, z) // Pipe of stages
- int const // pipe: input
- while (true) { // pipe: input
- int const pipeout // task data out put flow
- while (identity) { // task data input, task data output

So after implementation, we obtain this type:

Data space with data that I the pipe, I the pipe, I the pipe,