Report Number: WUCSE-2003-61

2003-08-22

The Greedy pipe Toolset Manual (Version 1.0)

Seema Datar and Mark A. Franklin

Chip Multi-Processors(CMPs) are now available in a variety of systems. They provide the opportunity to achieve high computational performance by exploiting application-level parallelism within a single chip form factor. In the communications environment, network processors (NPs) are often designed around CMP architectures and, in this context, the processors may be used in a pipelined manner. This leads to the issue of scheduling tasks on such processor pipelines. A tool and algorithm called Greedy pipe has been developed that determines the nearly optimal schedules for such multiprocessor pipelines. The tool offers a user friendly interface, with easily installable, portable and... **Read complete abstract on page 2.**

The Greedy pipe Toolset Manual (Version 1.0)

Seema Datar and Mark A. Franklin

Complete Abstract:

Chip Multi-Processors(CMPs) are now available in a variety of systems. They provide the opportunity to achieve high computational performance by exploiting application-level parallelism within a single chip form factor. In the communications environment, network processors (NPs) are often designed around CMP architectures and, in this context, the processors may be used in a pipelined manner. This leads to the issue of scheduling tasks on such processor pipelines. A tool and algorithm called Greedy pipe has been developed that determines the nearly optimal schedules for such multiprocessor pipelines. The tool offers a user friendly interface, with easily installable, portable and high-performance design. This report contains a description of the Greedy pipe tool set (release1.0), including the tool retrieval, installation and usage instructions.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2003-61

The Greedypipe Toolset Manual (Version 1.0)

Authors: Franklin, Mark; Datar, Seema

August 22, 2003

Abstract: Chip Multi-Processors(CMPs) are now available in a variety of systems. They provide the opportunity to achieve high computational performance by exploiting application-level parallelism within a single chip form factor. In the communications environment, network processors (NPs) are often designed around CMP architectures and, in this context, the processors may be used in a pipelined manner. This leads to the issue of scheduling tasks on such processor pipelines. A tool and algorithm called Greedypipe has been developed that determines the nearly optimal schedules for such multiprocessor pipelines. The tool offers a user friendly interface, with easily installable, portable and high-performance design. This report contains a description of the Greedypipe tool set (release 1.0), including the tool retrieval, installation and usage instructions.

The Greedypipe Toolset Manual (Version 1.0)

Seema Datar and Mark A. Franklin

Department of Computer Science and Engineering
Washington University in St.Louis, MO, USA
{seema,jbf}@ccrc.wustl.edu

Abstract

Chip Multi-Processors (CMPs) are now available in a variety of systems. They provide the opportunity to achieve high computational performance by exploiting application-level parallelism within a single chip form factor. In the communications environment, network processors (NPs) are often designed around CMP architectures and, in this context, the processors may be used in a pipelined manner. This leads to the issue of scheduling tasks on such processor pipelines. A tool and algorithm called *Greedypipe* has been developed that determines the nearly optimal schedules for such multiprocessor pipelines. The tool offers a user friendly interface, with easily installable, portable and high-performance design. This report contains a description of the Greedypipe tool set (release 1.0), including the tool retrieval, installation and usage instructions.

1 Introduction

The continuing increase in the logic and memory capacities associated with single VLSI chips along with the growth in the networking infrastructure has resulted in the development of Network Processors (NPs). Network processors are often combinations of Chip Multiprocessors (CMPs) augmented with special purpose logic, dedicated to efficiently performing networking functions yet flexible enough to respond to changes in protocol standards and functional requirements. Processors within NPs are often arranged in a pipelined manner. Packets of information arrive from the network and are classified and routed by a scheduler to one of a number of processor pipelines. A packet after being routed to a pipeline, will invoke one or more of the applications associated with the pipeline (e.g. routing, encryption, compression, etc.). After traversing the pipeline the packet returns to the scheduler where it is typically be sent to a switch fabric for transmission to the next node in the network.

Given a group of applications that have processor pipeline implementations specified as a series of sequential tasks, these tasks need to be assigned to the processors in the NP pipeline such that the system throughput is maximized. The tool and associated algorithm *Greedypipe*, obtains optimal application task assignments for such pipelines in reasonable time. The algorithm employs a greedy heuristic to schedule tasks derived from multiple application flows on pipelines with arbitrary number of stages.

2 Pipeline Task Assignment Problem

Network processors typically have multiple input flows where each flow represents a connection between designated source and destination in the network. Flows consist of a sequence of packets that are separated in time. Each flow has certain characteristics and processing demands requiring

certain application algorithms to be applied to the successive packets associated with the flow. It is assumed that the application algorithms consist of a series of sequentially ordered tasks and implemented on a pipeline of identical processors. Each processor in the pipeline operates on a packet, does some partial processing associated with the application (a particular task) and passes the packet to the next processor in the pipeline. The flow's processing requirements may have parts common with other flows and may share the processing on one or more of the pipeline segments.

The N incoming flows can be represented as the set F where

$$\mathrm{F} = \{F_1,\ F_2,\ F_3,\ \ldots\ ,F_N\}$$

with each incoming packet belonging to one of the N flows. The processing associated with each flow packet can be partitioned into an *ordered* set of M_j tasks, with T_{ij} corresponding to the application requirements of the flow where i and j respectively designate the task and flow number. Thus, for flow j:

$$T_j = (\ T_{1j},\ T_{2j},\ T_{3j},\ \ldots\ ,T_{M_jj}\ )$$

Each task belonging to a flow has a processor execution time given by:

$$t_j = \{t_{1j},\ t_{2j},\ \ldots\ ,t_{M_jj}\} \text{ where } j = 1 \text{ to } N.$$

The pipeline itself consists of R identical processors:

$$P = \{P_1,\ P_2,\ P_3,\ \ldots\ , P_R\}$$

The task allocation or assignment problem consists of mapping the full set of tasks onto the P processors of a pipeline in a manner that preserves task ordering within a flow and optimizes a given performance metric.

The following additional constraints also apply:

- The assignment process must maintain sequential task ordering.
- A task may only be assigned to a single processor.
- In situations where the same task is associated with multiple flows, there will be a single instantiation of the shared task and it will be assigned to a single pipeline stage. If this constraint for shared tasks is not desired, the tasks should be given different names.

3 Performance Metrics

The above defines the set of possible legal assignments. To determine an optimal assignment out of this set it is necessary to specify a performance metric. The metric of interest in the network processor environment relates to maximizing pipeline throughput (i.e., maximizing the number of packets per second that can flow through the pipeline).

We consider the case where there are one or more flows, flows that may share tasks, and a single pipeline. Assuming that the pipeline is synchronous (i.e., clocked), the clock period will be based on the time associated with the maximum stage execution time where that maximum is determined by the tasks assigned to each stage. The execution time for a single flow j, on a given stage k is given by:

$$s_{jk} = \sum_{i=1}^{M_j} X_{ijk} t_{ij} \tag{1}$$

where X_{ijk}=1 if the task is assigned to a stage and X_{ijk}=0 otherwise.

The maximum stage execution time for flow j across all the R stages is:

$$T_j = \max_{k=1}^{R}[t_{jk}] \;=\; \max_{k=1}^{R}\left\{\sum_{i=1}^{M_j} X_{ijk}t_{ij}\right\} \tag{2}$$

while the maximum stage execution time over all flows and stages is given by:

$$T = \max_{j=1}^{N} T_j = \max_{j=1}^{N}\;\left\{\max_{k=1}^{R}\;\left\{\sum_{i=1}^{M_j} X_{ijk}t_{ij}\right\}\right\} \tag{3}$$

To maximize packet throughput, the problem becomes one of finding a task assignment that minimizes T in Equation 3 since *packet throughput* $= 1/T$.

The next section describes application of the Grredypipe tool to different system configurations to obtain the optimal task allocation.

4 Example

To illustrate the operation of Greedypipe, we present a few examples with increasing complexity. The first example considered is a simple single flow system that has five tasks to be assigned to a three stage pipeline. The task details are as given in Table 1.

	Task 1	Task 2	Task 3	Task 4	Task 5
Flow 1 - Task Ids	T_1	T_2	T_3	T_4	T_5
Task Execution Times	2	4	3	1	3

Table 1: A Single Flow with five *ordered* tasks

The task Ids are represented using the notation T_i where i identifies a unique task. Figure 1 shows one of the possible assignments and the optimal allocation using *Greedypipe*.

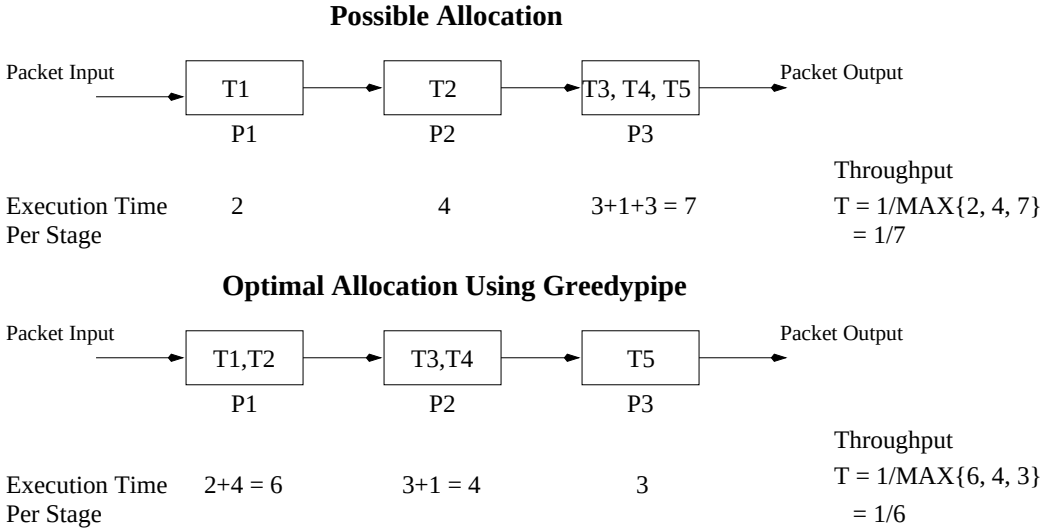

Figure 1: Example - Single Flow

The system throughput with the assignment using *Greedypipe* (Figure 1) is the same as is obtained if one performed a complete enumeration of possible assignments and picked the best one.

Next, we consider a case of increased complexity with two flows and a task shared (Task T_3) between the two flows. The tasks in these two flows (see Table 3) need to be allocated to a pipeline with three processors stages so that the shared task gets executed only on a single processor while still maintaining the task ordering associated with each of the flows.

	Task 1	Task 2	Task 3	Task 4	Task 5
Flow 1	T_1	T_2	T_3	T_4	T_5
Task Execution Times	5	4	3	1	3
Flow 2	T_6	T_3	T_8	T_9	
Task Execution Times	5	3	4	2	

Table 2: System configuration with Two Flows

Figure 2 shows one of the possible assignments and the optimal allocation using *Greedypipe*.

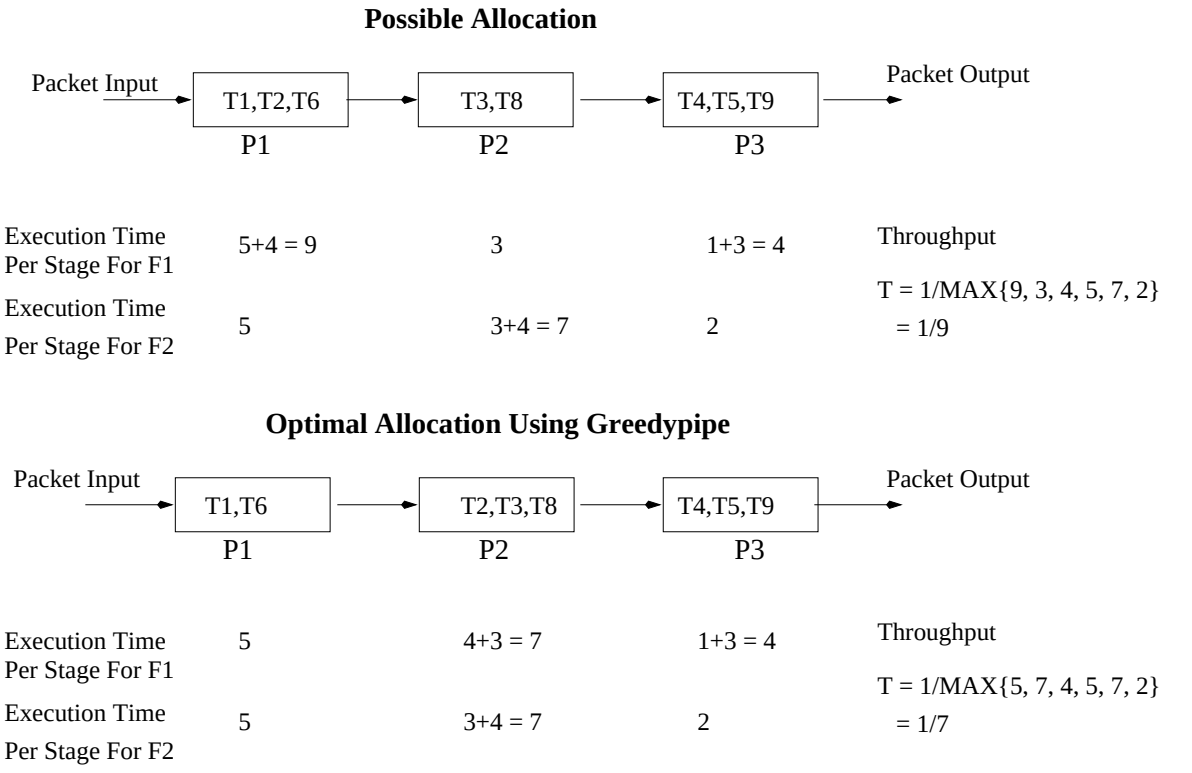

Figure 2: Example - Two flows with a shared task

In the above example, due to the presence of a shared task T_3, all the tasks before T_3 in both the flows need to get executed on or before the processor stage (P_2) that T_3 is assigned to. Again, the system throughput with the assignment using Greedypipe is the same as obtained using the complete enumeration of possible assignments.

5 Greedypipe Toolset

The Greedypipe tool uses an algorithm based in part on a greedy algorithm and thus does not guarantee an optimal solution, though it obtains an effective allocation quickly. Experimental results indicate that over a wide range of conditions 95% of the time, Greedypipe obtains schedules within 10% of optimal. The tool has been written using the C programming language. The installation comes in the form of a tar file called greedypipe.tar. To install the tool, extract the tar file and run the *make* command to build the binary *greedypipe*. The tool is enabled to accept the input parameters at the command prompt or as a configuration file to be placed in the same directory as the executable. To obtain the Greedypipe Toolset, e-mail your requests to seema@ccrc.wustl.edu or jbf@ccrc.wustl.edu.

5.1 Usage

The following command needs to be issued at the command prompt to run the *Greedypipe* tool

$ greedypipe

Greedypipe accepts the following command line arguments:

default	prints the help message
-f	reads the parameters from a configuration file named *config.txt*
-c	gets the input parameters from the command line

Table 3: Command Line Options For Greedypipe

The input parameters to be provided to the tool are the number of processor stages, the number of probable flows in the system and the task details for each of the flows.

5.1.1 Command Line Format

The input parameters to Greedypipe can be provided through the command line by using the following command

$ greedypipe -c

On issuing the above command the user is prompted to enter the system configuration interactively as shown in Figure 3.

The entries in **bold** are the user entered values. The tool expects an integer value to be entered for the number of processor stages, the number of flows and the number of tasks in flows. The minimum integer value for all the entities can be 1. The task details pertaining to each flow are entered in terms of task Ids and their respective execution times as

$TaskId_1 = Execution\ Time_1,\ TaskId_2 = Execution\ Time_2$,....

The task Ids need to have the format $Task\{xxx\}$ where 'xxx' is a unique integer identifying the task. The task Id can also be represented using a short-hand notation as $T\{xxx\}$. The execution time could be any decimal number to a precision of 6 decimal places. The order of the entered tasks also implies the sequence in which the tasks need to be executed for a flow. After getting all the system configuration parameters, the tool computes the optimal allocation and displays the task assignments for the pipeline. All the entries on the command line are case insensitive.

```
Please enter the number of stages in the pipeline : 3
Please enter the number of flows in the system   : 2
Please enter the number of tasks in Flow − 1     : 2

Please enter the task Ids in Flow − 1 with the corresponding execution times
(e.g. Task1=3.4,Task2=5.4,Task3=9.4)

Task1=2.3,Task2=4.5

Please enter the number of tasks in Flow − 2     : 2
Please enter the task Ids in Flow − 2 with the corresponding execution times
(e.g. Task1=3.4,Task2=5.4,Task3=9.4)

Task3=6.2,Task4=8.0
```

Figure 3: Sample - Interactive Command Line Option

5.1.2 Configuration File Format

The input parameters to Greedypipe can alternatively be provided through a configuration file, *config.txt*, by executing the following command

$ greedypipe -f

The configuration file should be placed in the same directory as the Greedypipe executable. The configuration file contains the system configuration details like the number of processor stages, the number of flows in the system, and the task details for each of th flows including their execution times. Following is a sample format of the configuation file.

```
# System Configuration File
Number of Stages = 3
Number of Flows =  2

# Task Details for Flows
# Each Task Identifier should be represented with a 'Task' appended
# by a unique integer associated with the Task Id
# The Task Identifiers and the corresponding execution times for a Flow
# should be represented as
# Flow[Integer]:Task[Integer]=Task Execution Time,
#                    Task[Integer]=Task Execution Time, ..
# Task Description for each Flow should be entered on a separate line
# Example :
# Flow1:Task1=3.0,Task2=4.0,Task3=5.3
# Flow2:Task4=2.3,Task5=7.4,Task6=6.5


Flow2:Task4=2.34367,Task5=7.4,Task6=6.5

Flow1:Task1=3.1,Task2=4.2,Task3=5.3
```

Figure 4: Sample - Greedypipe Configuration File

The contents of the file are not case sensitive. The lines in the file starting with the character '#' are considered to be comments. The system description primarily consists of three entries.

- The system description begins with the *Number of Stages* which denotes the number of processor stages in the pipeline and expects an integer value to be entered after the '=' delimiter.
- The second entry in the file is *Number of Flows* that needs an integer index specifying the number of flow classifications in the system, to be entered following the '=' delimiter.
- The third entry is a section in itself specifying the task details for each of the flows.

Each entry depicting the details for each flow should be entered on a new line and should comply with the following format.

$FlowId_1$:$TaskId_1$=$Execution\ Time_1$,$TaskId_2$=$Execution\ Time_2$,..

The flow identification needs to be entered in the format $Flow\{xxx\}$, where 'xxx' represents the unique integer identifying the flow. The flow Id can also be represented using a short-hand notation as $F\{xxx\}$. The flow identification is followed by the task identifiers with the respective execution times for each task. The sequence of task entries in a flow entry represents the order of execution of the tasks. If the task details for a flow cannot be entered on a single line, they should be entered on a new line in the following format.

$FlowId_1$:$TaskId_1$=$Execution\ Time_1$,$TaskId_2$=$Execution\ Time_2$,..
$FlowId_1$:$TaskId_3$=$Execution\ Time_3$,$TaskId_4$=$Execution\ Time_4$,..

5.1.3 Output Format

Given a set of input parameters for the system configuration, the Greedypipe tool determines and displays the optimal task assignments for the pipeline. Figure 5 shows a sample output format.

```
================================================================
                        Greedypipe Allocation
================================================================

 Processor 1 :  Task1  Task6
 Processor 2 :  Task2  Task3  Task8
 Processor 3 :  Task4  Task5  Task9
 Pipeline Throughput = 1/7.000000 = 0.1428
================================================================
```

Figure 5: Sample - Greedypipe Tool Output

The displayed output (Figure5) shows the tasks allocated to each processor stage and also the throuput of the system for the given allocation.

6 Greedypipe Example

This section gives complete examples of execution of the *Greedypipe* tool using both the command line option and the the configuration file option. The first example being considered has a system configuration with 3 processor stages and two flow classifications. The applications for the first flow are pipelined into two tasks while those for the second flow have been pipelined into three tasks. Figure 6 shows all the steps of the *Greedypipe* tool execution when the above system configuration information with the execution times for the slots is entered interactively through the command-line.

```
======================== Input Paramaters ==============================
Please enter the number of stages in the pipeline :        3
Please enter the number of flows in the system   :        2
Please enter the number of tasks in Flow − 1      :        2
Please enter the task ids in Flow − 1 with the corresponding execution times
(e.g. T1=3.4,T2=5.4,T3=9.4)
T1=2,T2=3.4,T3=5
Please enter the number of tasks in Flow − 2      :        3
Please enter the task ids in Flow − 2 with the corresponding execution times
(e.g. T1=3.4,T2=5.4,T3=9.4)
T4=6,T5=1.2,T6=4
============================System Configuration=======================
Flow 1            −   Task1       Task2       Task3
Execution Time −   2.000000    3.400000   5.000000
Flow 2            −   Task4       Task5       Task6
Execution Time −   6.000000    1.200000    4.000000
========================== Greedypipe Allocation ======================
Processor = 1 :  Task1  Task4
Processor = 2 :  Task2  Task5
Processor = 3 :  Task3  Task6
Pipeline Throughput = 1/6.000000 = 0.1667
=======================================================================
```

Figure 6: Sample - Greedypipe Tool Interactive Execution

The next example being considered has the input parameters provided through the configuration file. The pipeline has three processor stages and two flows. The applications in the first flow are pipelined into five tasks while those in the second flow are pipelined into four tasks. Figure **??** shows the contents of the *config.txt* file that describes the pipeline configuration details for the Greedypipe tool.

```
_____________________________________________________________
# System Configuration File
Number of Stages = 3
Number of Flows =  2
# Task Details for Flows
# Each Task Identifier should be represented with a 'Task' appended
# by a unique integer associated with the Task Id
# The Task Identifiers and the corresponding execution times for a Flow
# should be represented as
# Flow[Integer]:Task[Integer]=Task Execution Time,
#                    Task[Integer]=Task Execution Time, ..
# Task Description for each Flow should be entered on a separate line
# Example :
# Flow1:Task1=3.0,Task2=4.0,Task3=5.3
# Flow2:Task4=2.3,Task5=7.4,Task6=6.5
Flow1:Task1=5,Task2=4,Task3=3,Task4=1,Task5=3
Flow2:Task6=5,Task3=3,Task8=4,Task9=2
============================================================
```

Figure 7: Example - config.txt

Figure 8 shows the allocation generated by Greedypipe for the above system configuration.

```
========================== System Configuration =================

 Flow 1        -      Task1        Task2        Task3        Task4        Task5
Execution Time -   5.000000     4.000000     3.000000     1.000000     3.000000
 Flow 2        -      Task6        Task3        Task8        Task9        9
Execution Time -   5.000000     3.000000     4.000000     2.000000

========================== Greedypipe Allocation ================
  Processor = 1 :  Task1  Task6
  Processor = 2 :  Task2  Task3  Task8
  Processor = 3 :  Task4  Task5  Task9
  Pipeline Throughput = 1/7.000000 = 0.1428
============================================================
```

Figure 8: Example - Greedypipe Tool Output

More information regarding the employed *Greedypipe* algorithm can be found in the Technical Report *Task Scheduling of Processor Pipelines with Application to Network Processors* (Technical Report #WUCSE-2003-60), Washington University in St.Louis, Department of Computer Science and Engineering.