UWM Milestone Report

Binary Inspiral Hierarchical Search Code

Bruce Allen, Duncan Brown, Jolien Creighton

1. Introduction

Architecture is based on a master and slave design.

Design is slave driven.

Slave:

- makes requests to master for templates, data
- does filtering
- returns events to master

Master:

- loops, servicing slave requests
- distributes templates, data to slaves
- records events returned by slaves

2. Master

Master completed in December 1999 (milestone 30th November 1999)

Code released in LAL version 0.2b (20th December 1999)

`packages/findchirp` `packages/comm` `packages/framedata`

Master is changing slightly as LAL standard evolves and slave is written

Master will use Cardiff template bank placement routines

Design Flexibility

- Communication routines isolated and general purpose (based on MPI)
- Input/Output routines isolated
- No "built in" assumptions about the nature of the waveforms/template bank

3. Architecture of Master

call template bank parameter creation routines
store template parameters

```
while (there are slave processes)

    wait for message from slave

    switch (message)
       case exchange inspiral templates
          send template parameters to slave
       case exchange data segment
          send data segments to slave
       case exchange event
          receive event from slave
       case slave finished
          decrease slave process count
```

4. Slave

Milestone for completion of slave 29th February 2000

Work on slave code is in progress.

We are implementing a slave that can perform an n level hierarchical search.

Slave uses a linked list to store coarse templates.

When an event is found at the coarse level, fine templates can be inserted into the list.

The slave

- transforms data segments to frequency domain,
- calls chirp generation function,
- filters data segments against generated chirp.

5. Slave Architecture

To allow an n level hierarchical search we define the level of a template to be its depth in the hierarchy

Zero for the coarsest templates down to $n-1$ for the finest templates

For a two stage search we would have two levels, 0 (highest) and 1 (lowest).

Data Structures

Template parameter structure and data segment structure has an INT4 to record level: `templateLevel` and `segmentLevel`

Initially all `segmentLevel` set to zero: can change during execution

`templateLevel` set to level of template in hierarchy: fixed

9. Current Status

Master has been written, but is evolving as slave is written.

Communication code has been written: `packages/comm`

Data buffering routines have been written: `packages/databuffer`

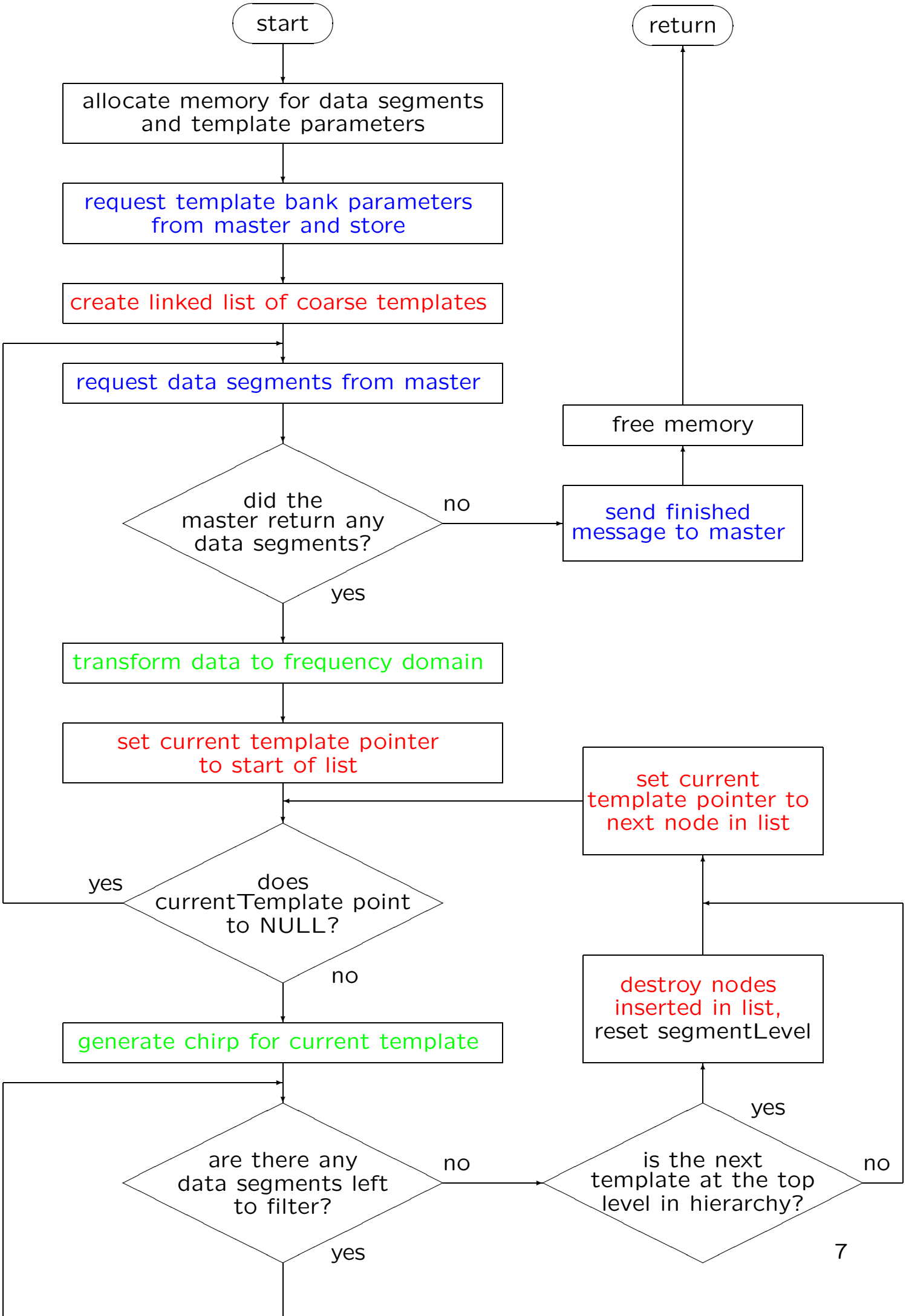
Core logic of slave code has been written:

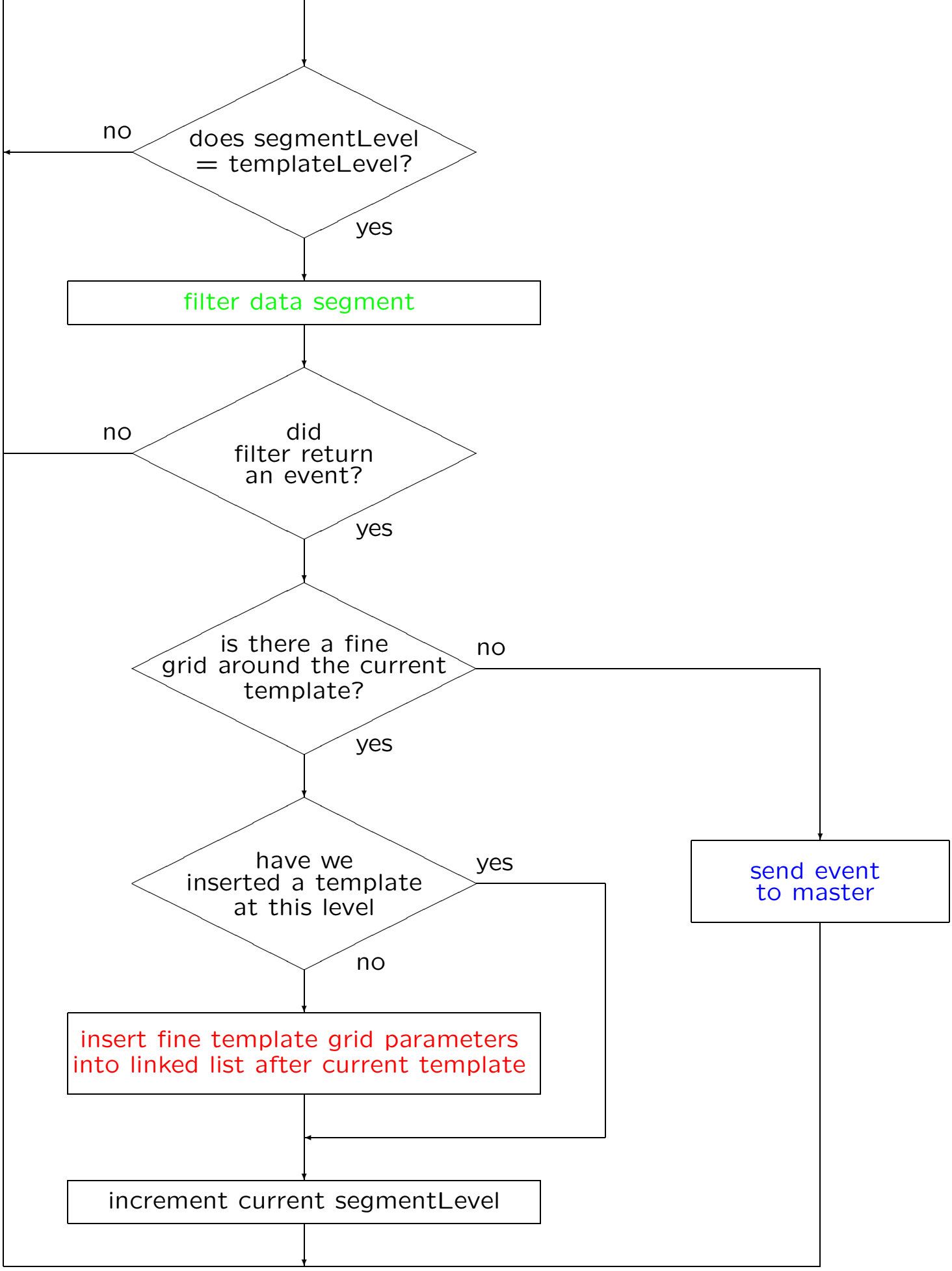
- linked list functions for creation and deletion of templates in list
- main loop of slave which forms hierarchical search engine

Currently under development are:

- Template parameter grid generation functions (Cardiff)
- Chirp generation function (Cardiff)
- Function to transform data segment to frequency domain
- Function to perform matched filtering

The last two functions are currently under development at UWM to be completed by the end of March.


start
allocate memory for data segments and template parameters
request template bank parameters from master and store
create linked list of coarse templates
request data segments from master
did the master return any data segments?
no
yes
send finished message to master
free memory
return
transform data to frequency domain
set current template pointer to start of list
set current template pointer to next node in list
does currentTemplate point to NULL?
yes
no
destroy nodes inserted in list, reset segmentLevel
generate chirp for current template
are there any data segments left to filter?
no
yes
is the next template at the top level in hierarchy?
yes
no


does segmentLevel
= templateLevel?
no
yes
filter data segment
did
filter return
an event?
no
yes
is there a fine
grid around the current
template?
no
yes
have we
inserted a template
at this level
yes
no
send event
to master
insert fine template grid parameters
into linked list after current template
increment current segmentLevel