# GPU-MatchLocate1.0 (GPU-M&L)

Aug. 15, 2019

## 0.Author

Min Liu, China University of Geosciences (Beijing) (liumin@cugb.edu.cn)

## 1.References

1)Zhang, M., and L. Wen (2015), An effective method for small event detection: match and locate (M&L), Geophysical Journal International, 200(3),1523-1537.

2)Beaucé, E., W. B. Frank, and A. Romanenko (2017), Fast matched filter (FMF): An efficient seismic matched-filter search for both CPU and GPU architectures, Seismological Research Letters, 89(1), 165-172.

3)Liu, M. H. Li, M. Zhang, and T. Wang (2019), Graphics Processing Unit-based Match&Locate (GPU-M&L): An improved Match and Locate method and its application. (submitted)

## 2.Introduction

GPU-M&L (Graphics Processing Unit-based Match&Locate) is an improved Match&Locate (M&L). The GPU-M&L differs from the M&L in two ways: 1) adding weighting factor for each component of templates to improve the detection ability; 2) implementing the M&L method on GPU to accelerate the computation.

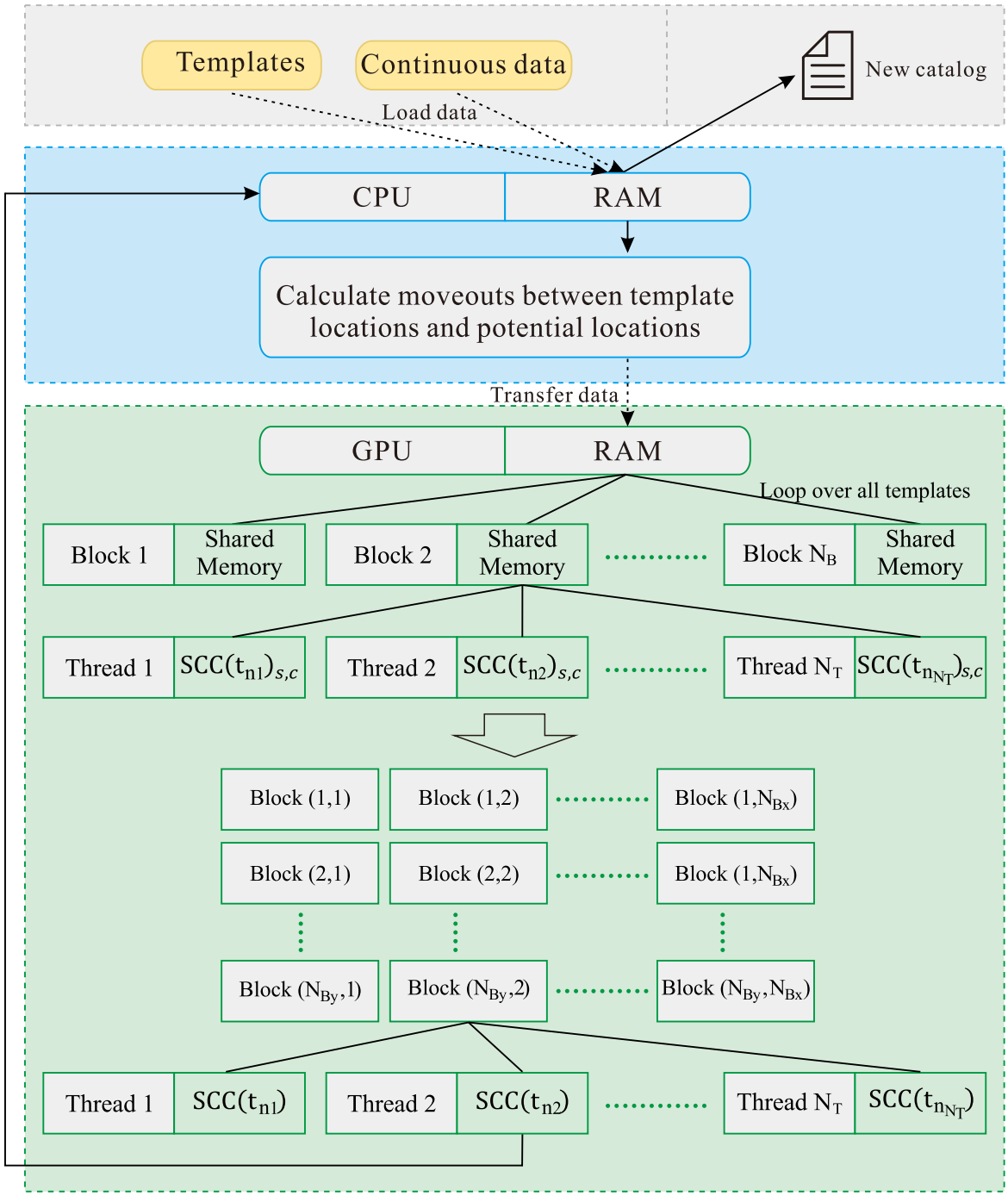


**Fig. 1.** Workflow of the GPU-M&L method. Templates and continuous data are first loaded into random access memory (RAM) of CPU, and moveouts between all template locations and potential locations are calculated. Next, templates and continuous data are transferred to RAM of GPU from RAM of CPU and then transferred to the shared memory of block. Two kernels of GPU

loop each template to compute SCCs and stack them based on the moveouts. Stacked SCCs are transferred back to the RAM of CPU. At the end, positive detections form a new catalog.

**3.Requirements**

ObsPy (https://github.com/obspy/obspy, for data downloading, processing and picking only)
SAC (http://ds.iris.edu/ds/nodes/dmc/software/downloads/sac, for data processing only)
TauP(http://www.seis.sc.edu/taup/index.html, for P/S wave arrival time marking only)
GMT (https://www.soest.hawaii.edu/gmt, for figure plotting only)
CUDA Toolkit (https://developer.nvidia.com/cuda-downloads, for compiling cuda code)
Pssac(http://gmt.soest.hawaii.edu/doc/latest/supplements/meca/pssac.html, for figure plotting only)

**4.Usage (Type "GPU_MatchLocate")**

-R (maxlat/maxlon/maxh)
-I (dlat/dlon/dh)
-T (template_window/before/after)
-D (INTD/threshold)
-N (n_templates)
-G (step/delay/segments)
-O (0 or 1)

**********Explanation**********

**-R:**
**(maxlat/maxlon/maxh)**
**(degree/degree/km)**
e.g., 0.03/0.03/3
maxlat: search range in latitude centered at template (degree)
maxlon: search range in longitude centered at template (degree)
maxh: search range in depth centered at template (km)

**-I:**
**(dlat/dlon/dh)**
**(degree/degree/km)**
e.g., 0.003/0.003/0.3
dlat: search grid size for latitude (degree)
dlon: search grid size for longitude (degree)
dh: search grid size for depth (km)

**-T:**
**NT: The cross-correlation window based on the marked t1 (P/S phase) in your templates (template_window/before/after)**
**(sec/sec/sec)**
e.g., 6/1/5
template_window: total length of template waveform (or cross-correlation window) (sec)
before: length of template waveform before the marked t1 (sec)

after: length of template waveform after the marked t1 (sec)

**-D:**
**(INTD/threshold)**
e.g., 2/9
INTD: keep one event within a certain time (sec)
threshold: User-specified detection threshold (multiple of the median absolute deviation)

**-N:**
**(n_templates)**
e.g., 10
n_templates: total events in your template catalog (catalog.dat)

**-G:**
**(step/delay/segments)**
e.g., 1/0.01/1
step: moveout for slide cross-correlation
delay: sampling rate of input data
segments: number of segments in your continuous data
[For dense search task, the easiest way to solve memory overload is increase segments]

**-O:**
**(0 or 1)**
0: don't output stacked cross-correlograms
1: output stacked cross-correlograms

**********

**5.Dependent files**
Template directory:
Template/yearmonthdayhoursminsec/net.station.component (e.g., Template/20190704213334.320/CI.SLA.HHZ)

Trace directory:
Trace/yearmonthday/net.station.component (e.g., Trace/20190704/CI.SLA.HHZ)

Slowness files:
Template/INPUT/yearmonthdayhoursminsec (e.g., Template/INPUT/ 20190704213334.320)

Template catalog:
catalog.dat

**6.Formats for input files:**
Slowness and weighting factor files:
net.station.component, P/S wave arrival time, horizontal slowness/vertical slowness, weighting factor
e.g., CI.SLA.HHZ 13.140 33.9890/6.2960e-02 0.08

Template catalog:
yearmonthdayhoursminsec, latitude, longitude, depth, magnitude, ref_latitude, ref_longitude, ref_depth
e.g., 20190704213334.320 35.6139 -117.5864 8.609 3.08 35.6139 -117.5864 8.609
Note: latitude, longitude and depth represent the spatial location of template, respectively. ref_latitude, ref_longitude and ref_depth indicate an user-specified center of searching grids, which is usually assigned as the location of template.

**7. Formats for output files:**
DetectedFinal.dat:
Num., year/month/day, hour:minute:second, lat., long., dep., mag., coef., mad., ref.
e.g.,1 2019/07/04 00:53:10.238 35.6298 -117.5933 10.32 1.22 0.1373 9.0200 20190704214704.040

**8.Demo of GPU-M&L**
#compile source code
$cd GPU-ML_release1.0/src
$make
$cd sacCC
$make

#download continuous data (User-specified)
$cd ../../Demo
$python data_download.py

#download templates and create routine catalog (User-specified)
$python template_download.py

#set common origin time
$cd ./Trace
$perl SACH_O.pl 20190704

#mark P/S wave arrival time and calculate slowness
$cd ../Template
$perl marktaup_p.pl
$perl marktaup_s.pl

#calculate weighting factor for each trace
$bash cal_weights.sh

#detect and locate events
$cd ../
$perl RunprocAll.pl

#generate new catalog
$cd ./MultipleTemplate
$perl MergeEvents.pl 20190704
$perl SelectFinal.pl 2019 07 04 Allevents

#compare waveforms between template and detection
$perl PlotEventWaveform.pl DetectedFinal.dat 1