DT Matlab Toolkit

DataTurbine Matlab Interface

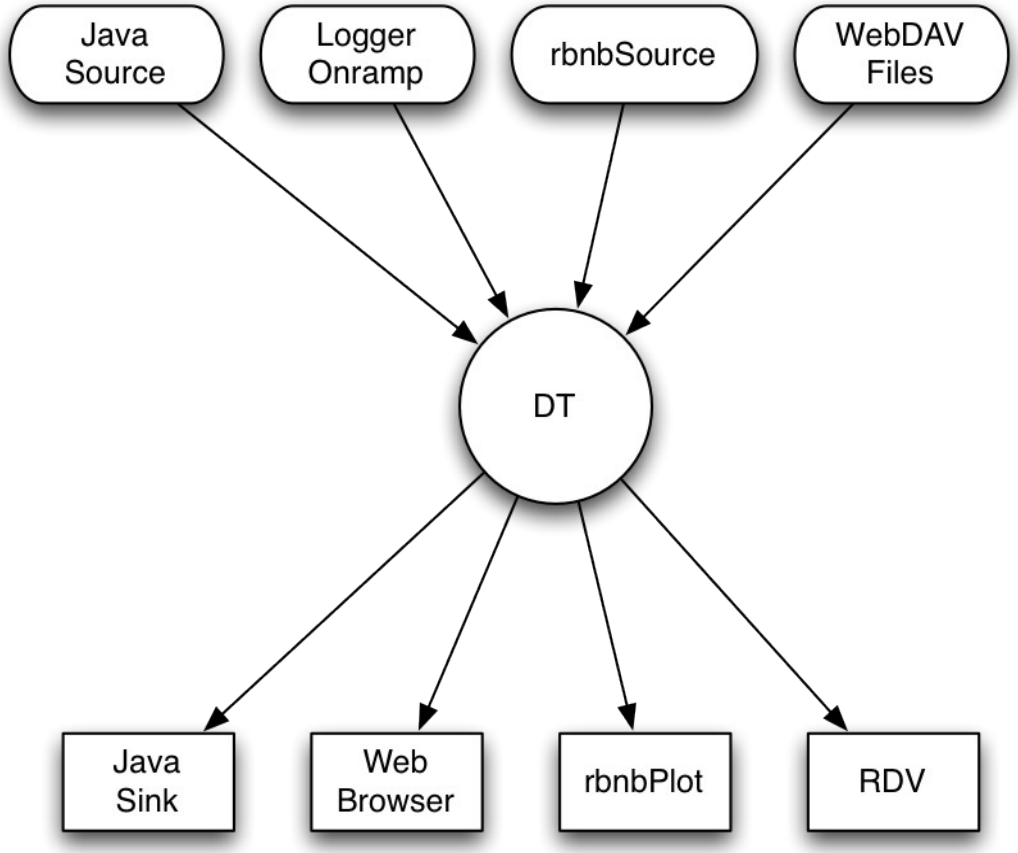
Matt Miller
Cycronix
April, 2013

DT-Matlab Toolkit

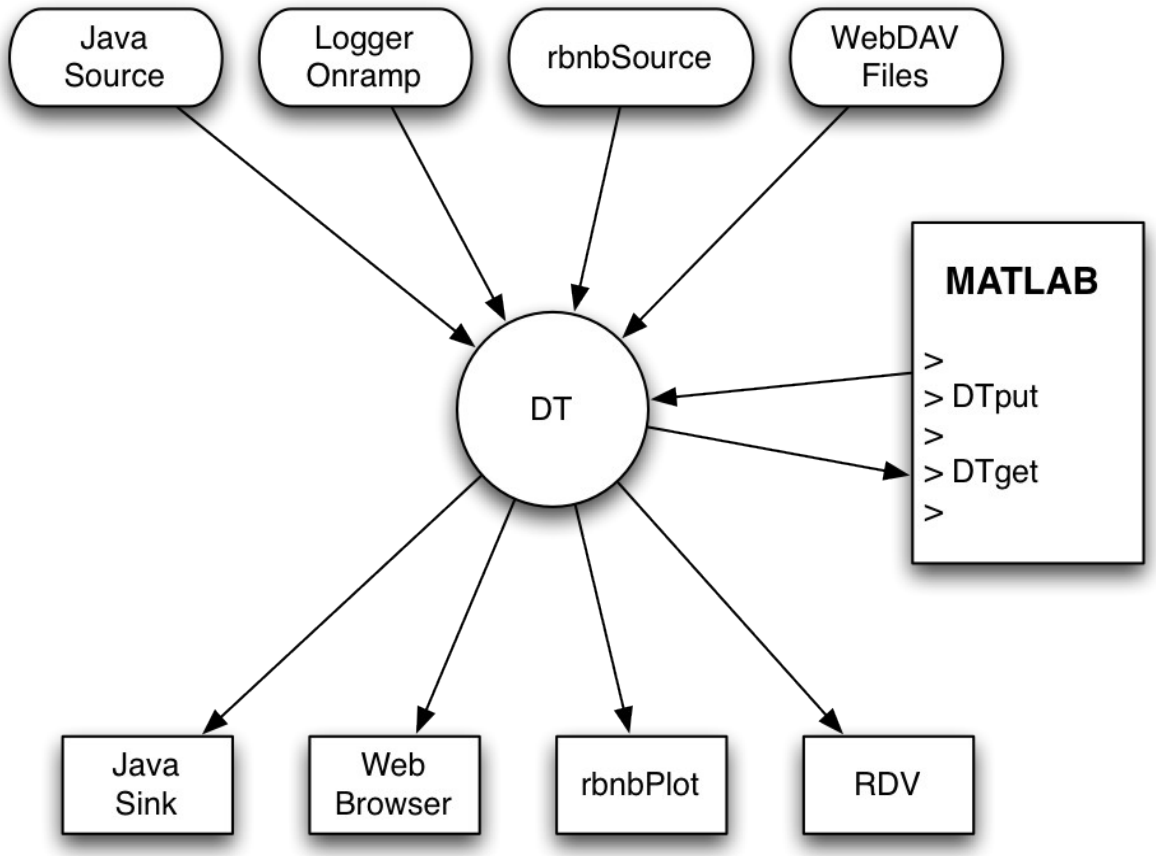
- New 2013
- High-level DT interface for Matlab
- Standard data structures
- Simple interface
- Foundation for DT to GCE Interface
- Download:

  https://bitbucket.org/MattMiller/matlab-dt-toolkit/downloads/DTMatlabTK.zip

DT is Hub to Sources & Sinks



Matlab is Another Source/Sink



Matlab Setup

- Java classpath
  - javaaddpath('/pathto/rbnb.jar')
- Toolkit path
  - Set Path, add folder(s) with DT M-files

Matlab – DT Components

- Low level M-files (old)
- Regression Test Scripts (old)
- DT-Matlab Toolkit (new)
- Built upon Java SAPI interface:

```
snk = com.rbnb.sapi.Sink();
snk.OpenRBNBConnection(DT.server,'dtSink');
cget = com.rbnb.sapi.ChannelMap();
```

Old - Low Level Scripts

- Matlab is inherently Java-compatible.
- Legacy Java-API scripts are available:
  - rbnb_open
  - rbnb_source
  - rbnb_sink
  - rbnb_put
  - rbnb_get
  - rbnb_cmap

Old - Regression Test Scripts

- DataTurbine self-check, 'testrbnb' calls suite of test scripts:

```
>> testrbnb
PASS: open/close test
PASS: testput1
PASS: testput2
PASS: testput3
PASS: testmulti
PASS: multiple ring buffer test
PASS: user data test
PASS: archive test
PASS: zero duration request test
PASS: zero duration request non-zero duration data test
PASS: monitor stream test
PASS: subscribe stream test
PASS: frame-based streaming from oldest test
PASS: time-based streaming from oldest test
PASS: subscribe from archive test
PASS: real-time (time-based) subscribe test
PASS: wildcard time-mode subscribe test
PASS: streaming from oldest (time) with duplicates test
PASS: streaming from oldest (time) with an odd frame test
PASS: many channels test
PASS: adding to newest data test
PASS: extend start request test
PASS: max wait test
PASS: detach source test
PASS: reatttach test
PASS: password test
```

New - Data Structures

- DTstruct
  - server, source, time
- DTchan
  - channel name, time, data

Data Structures

```
function DT = DTstruct()

% DT = DTstruct()
%   Return template DT structure:
%       DT.server = 'localhost';
%       DT.source = '_Metrics';
%       DT.start = 0;
%       DT.duration = 0;
%       DT.reference = 'newest';
%       DT.chan = '*';

DT.server = 'localhost';
DT.source = '_Metrics';
DT.start = 0;
DT.duration = 0;
DT.reference = 'newest';
DT.chan = '*';

end
```

```
function chan = DTchan()

% chan = DTchan()
%   Return default DTchan structure:
%       chan(1).name = 'MemoryUsed';
%       chan(1).time = 0;
%       chan(1).data = 0;
%       chan(1).type = 'int64';
%       chan(1).mime = 'binary';
%       chan(1).meta = 'null';

chan.name = 'MemoryUsed';
chan.time = 0;
chan.data = 0;
chan.type = 'int64';
chan.mime = 'binary';
chan.meta = 'null';

end
```

New - Functions

- DTput – put data to DT
- DTget – get data from DT
- DTfetch – fetch multiple DT
- DTnext – get 'next' data without overlap
- DTlist – get list of channels from DT

Functions - Help

```
>> help DTget
 [ DTgot nchan ] = DTget( DT )
    Get data from DT. Input & output is 'DT' struct.
    Optional 2nd output param is # of fetched chans
    DT structure parameters (with defaults):
        DT.server = 'localhost';
        DT.source = '_Metrics';
        DT.start = 0;
        DT.duration = 0;
        DT.reference = 'newest';
        DT.chan = '*';

    On input DT.chan is string (or struct array).
    On output, DT.chan is array DTchan structs:
        chan(1).name = 'MemoryUsed';
        chan(1).time = 0;
        chan(1).data = 0;
        chan(1).type = 'int64';
        chan(1).mime = 'binary';
        chan(1).meta = 'null';
```

Tutorial

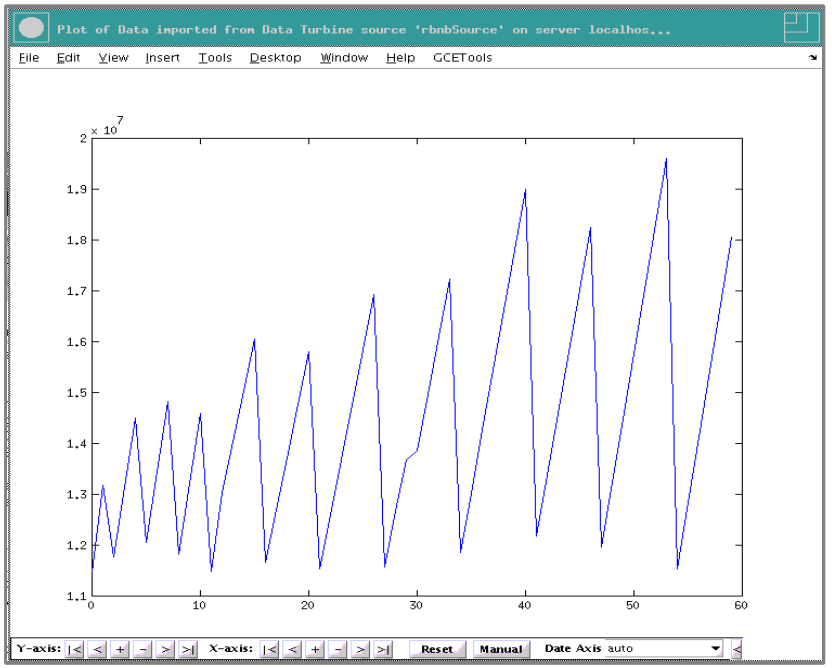
- Simple Scripts to Demonstrate Toolbox
- Run scripts, examine code
  - testplot
  - testput, testget
  - testnext
  - testTrout

testplot

```
% Loop plotting newest Metrics data (overlaps data)

% setup the DT structure:
dget.server = 'localhost';
dget.source = '_Metrics';
dget.start = 0;
dget.duration = 60;
dget.reference = 'newest';
dget.chan = 'MemoryUsed';

% loop for a minute:
for i=1:60
    dgot = DTget(dget);
    tref = dgot.chan(1).time(1);
    plot(dgot.chan(1).time - tref, dgot.chan(1).data);
    pause(1);
end
```



testput, testget

```
% testput.m
% Put single frame of data

% setup Channel array:
chan(1).name = 'c1';
chan(1).data = 1;
chan(2).name = 'c2';
chan(2).data = 2;

% setup the DT structure:
dput.server = 'localhost';
dput.source = 'mysource';
dput.chan = chan;

% put it
nput = DTput(dput);

fprintf('what we put:\n');
dput
dput.chan(:).name
dput.chan(:).data
```

```
% testget.m
% Get single frame from testput

% setup the DT structure:
dget.server = 'localhost';
dget.source = 'mysource';
dget.start = 0;
dget.duration = 0;   % single frame
dget.reference = 'newest';
dget.chan = '*';

% get it
dgot = DTget(dget);

fprintf('what we got:\n');
dgot
dgot.chan(:).name
dgot.chan(:).data
```

testnext

- Demonstrates DTnext
- Sequentially fetches playback or live data
- Note: GCE toolbox function “DTlatest”
  - similar function directly to GCE format

```
>> testnext
MemoryUsed, time: 1365087948.664, data: 15412688
MemoryUsed, time: 1365087949.665, data: 16858848
MemoryUsed, time: 1365087950.665, data: 11553000
MemoryUsed, time: 1365087951.665, data: 12879160
MemoryUsed, time: 1365087952.665, data: 14213736
MemoryUsed, time: 1365087953.665, data: 15558232
MemoryUsed, time: 1365087954.665, data: 16907472
MemoryUsed, time: 1365087955.665, data: 11777840
MemoryUsed, time: 1365087956.666, data: 13152512
MemoryUsed, time: 1365087957.666, data: 14487400
>>
```

testTrout

```
% testTrout.m
% subscribe to new data (no overlaps)

niter = 100;          % iterations to fetch
delay = 60;           % check once per minute

% setup the DT structure:
dget.server='144.92.62.139';    % address of DT server
dget.source='TroutBog';          % this source
dget.start=0;                   % align with newest
dget.duration=3600;             % hour of data each fetch
dget.reference='newest';        % start with most recent
dget.chan='*';                  % all channels

% crawl thru data and print status
for i=1:niter
    [dget, nchan] = DTnext(dget,delay);
    if(nchan)
        fprintf('%s, nchan: %d, npts: %d, time: %.0f\n', dget.source, ...
            length(dget.chan), length(dget.chan(1).data), dget.start);
    else
        fprintf('no data\n');
    end
end
```