# uLoadWatch: Monitoring Application System Load

**June 2018**

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139
project-pavlab/appdocs/app_uloadwatch

## 1 Overview

The `uLoadWatch` application monitors data produced by individual applications and reports a warning when the MOOS community begins to exhibit a strain in the CPU load. Load data is derived solely from two MOOS variables published by participating MOOS apps. For an example application `pFooBar`, the following two variables are published:

- `PFOOBAR_ITER_GAP`: indicates the ratio of (a) observed time between invocations to the application's `Iterate()` method, and (b) the scheduled time between iterations based on the configured `AppTick`. For example, for an application running at 4 Hz, a gap of 1.0 means things are running normally with 0.25 seconds in between iterations. A gap of 3 indicates that there was 0.75 seconds between iterations - an an idication that the CPU cannot keep up with the work being done in the iterate loop. This could, however, be due in part to other processes demanding CPU resources.
- `PFOOBAR_ITER_LEN`: indicates the ratio of (a) observed time between the begin and end of the `Iterate()` loop, and (b) again, the scheduled time between iterations based on the configured `AppTick`. In this case, a value of 1 is an indication that the work being done in the iterate loop is dangerously close to capacity.

The `ITER_GAP` variable indicates when strain is actually being exhibited. The `ITER_LEN` variable indicates the run-up to problems. The `uLoadWatch` application monitors the situation by registering for the `*_ITER_GAP` and `*_ITER_LEN` from all apps and doing three things:

1. Threshold detection. A user configured threshold for ITER_GAP values may be set, with a posting to LOAD_WARNING when it is exceeded.
2. An AppCast run warning will also be posted when the user configured threshold is exceeded. This will bring a critical load situation to the attention of an operator through the existing AppCast mechanisms.
3. An AppCast report is generated reporting the average and maximumn ITER_GAP and ITER_LEN noted thus far. This can be monitored via one of the AppCast Viewers to monitor the load fluctuation.

## 2 Configuration Parameters for uLoadWatch

The uLoadWatch application may be configured with a configuration block within a MOOS mission file, typically with a .moos file suffix. The following parameters are defined for uLoadWatch.

*Listing 2.1: Configuration Parameters for uLoadWatch.*

| | |
|---|---|
| breach_trigger: | The number of times a threshold can be breached before a warning is posted. The default is 1 meaning the first offense is forgiven, since it is not uncommon for some apps to have a longer initial gap as they perform one-time start-up work. |
| near_breach_thresh: | In addition to normal threshold breaches, the uLoadWatch app may also monitor for *near* breaches. A near breach occurs when the normal breach thresh is nearly met. For example, if the threshold for a certain app is 2.0, this means breach is declared if time between app iterations is greater than 2x the normal gap between iterations. If the near_breach_thresh is set to 0.6, a near breach is declared if the gap reaches 1.6$x$ the normal gap. |
| thresh: | A gap threshold for reporting a LOAD_WARNING. Each threshold line specifies the application name and the gap threshold value. |

### An Example MOOS Configuration Block

An example MOOS configuration block may be obtained from the command line with the following:

```
$ uLoadWatch --example or -e
```

*Listing 2.2: Example configuration of the uLoadWatch application.*

```
===============================================================
pHostInfo Example MOOS Configuration
===============================================================

ProcessConfig = pHostInfo
{
  AppTick   = 4
  CommsTick = 4

  thresh = app=pHelmIvP, gapthresh=1.5
  thresh = app=any,      gapthresh=2.0

  near_breach_thresh = 0.9   // default is off

  breach_trigger = 5         // default is 1
}
```

## 3 Publications and Subscriptions for uLoadWatch

The interface for uLoadWatch, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ uLoadWatch --interface or -i
```

### 3.1 Variables Published by uLoadWatch

The primary output of uLoadWatch is the occasional load warning plus the appcasting report. In Release 20.2, a few additional variables were added to summarize breaches and near breaches.

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility.
- LOAD_WARNING: A warning indicating an app has been detected reporting an gap greater than the configured threshold.
- ULW_BREACH: Posted with value `true`, if and only when a breach has been detected, for any app.
- ULW_BREACH_COUNT: The total amount of breaches detected, for all all apps.
- ULW_BREACH_LIST: The list of all apps for which at least one breach has been detected.
- ULW_NEAR_BREACH: Posted with value `true`, if and only when a *near* breach has been detected, for any app.
- ULW_NEAR_BREACH_COUNT: The total amount of *near* breaches detected, for all all apps.
- ULW_NEAR_BREACH_LIST: The list of all apps for which at least one *near* breach has been detected.

### 3.2 Variables Subscribed for by uLoadWatch

The uLoadWatch application subscribes all variables ending with the suffix _ITER_GAP and the suffix _ITER_LEN:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- *_ITER_GAP: A report from an application indicating its most recent observed gap between iterations.
- *_ITER_LEN: A report from an application indicating its most recent observed itertion duration.

### 3.3 Command Line Usage of uLoadWatch

The uLoadWatch application is typically launched with pAntler, along with a group of other shoreside modules. However, it may be launched separately from the command line. The command line options may be shown by typing:

```
$ uLoadWatch --help or -h
```

*Listing 3.3: Command line usage for the uLoadWatch tool.*

```
Usage: uLoadWatch file.moos [OPTIONS]

Options:
  --alias=<ProcessName>
      Launch pHostInfo with the given process
      name rather than pHostInfo.
  --example, -e
      Display example MOOS configuration block
  --help, -h
      Display this help message.
  --interface, -i
      Display MOOS publications and subscriptions.
  --version,-v
      Display the release version of pHostInfo.

Note: If argv[2] is not of one of the above formats
      this will be interpreted as a run alias. This
      is to support pAntler launching conventions.
```

## 4 Usage Scenarios the uLoadWatch Utility

A typical usage of the `uLoadWatch` utility is shown in Figure 1 below. If the mission is being monitored by a human (vs. an offline simulation), a load warning will present itself in any of the appcast viewing tools, e.g., `pMarineViewer`, `uMACView`, or `uMAC`. The `pNodeReporter` application also registers for `LOAD_WARNING` mail and will include the load warning in vehicle node reports. Depending on how `pMarineViewer` is configured, the vehicle label may indicate a load warning if it occurs.

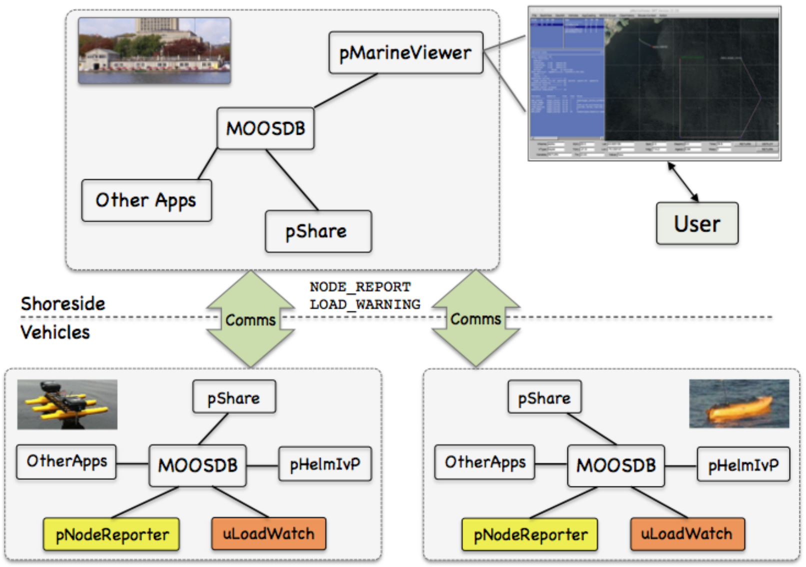

Figure 1: **Typical uLoadWatch Topology:** A shoreside or topside community is receiving information from several deployed vehicles, in the form of node reports, and AppCast reports and warnings. If a load warning is produced, it will (a) show up in any shoreside appcast viewer, (b) show up in node reports sent to the shoreside, and (c) show up in local vehicle alog files.

When running an offline simulation, the output of uLoadWatch is primarily found in the log files.

## 5 Terminal and AppCast Output

The uLoadWatch application produces some useful information to the terminal on every iteration of the application. An example is shown in Listing 4 below. This application is also appcast enabled, meaning its reports are published to the MOOSDB and viewable from any uMAC application or pMarineViewer. See the document for uMac for more on appcasting and viewing appcasts. The counter on the end of line 2 is incremented on each iteration of uLoadWatch, and serves a bit as a heartbeat indicator. The "0/0" also on line 2 indicates there are no configuration or run warnings detected.

*Listing 5.4: Example terminal or appcast output for* uLoadWatch.

```
 1  ===================================================================
 2  uLoadWatch henry                                            0/0(129)
 3  ===================================================================
 4  Configured Thresholds:
 5    ANY: 2
 6    PHELMIVP: 1.5
 7
 8  Application        AvgGap  MaxGap  AvgLen  MaxLen
 9  ----------------   ------  ------  ------  ------
10  PBASICCONTACTMGR   1.01    1.02    0.00    0.02
11  PHELMIVP           1.01    1.04    0.00    0.01
12  PHOSTINFO          1.01    1.01    0.00    0.58
13  PNODEREPORTER      1.01    1.02    0.00    0.01
```

```
14  UFLDNODEBROKER    1.01    1.01    0.00    0.00
15  ULOADWATCH        1.01    1.01    0.00    0.00
16  UPROCESSWATCH     1.00    1.02    0.00    0.00
17  USIMMARINE        1.07    1.15    0.02    0.17
```

The first few lines indicate any user-configured thresholds being applied to incoming reports. Here a LOAD_WARNING will be issued if the pHelmIvP has an iter gap above 1.5, and if any application has a gap over 2.0. The remainder of the report (lines 8-17 here) provide a live update of the average and maximum iter gap and iter length reported for each application reporting.

## 6 How to Modify a MOOS App to Produce Load Information

All the applications shown in Listing 4 above produce the *_ITER_GAP and *_ITER_LEN information by virtue of being an AppCastingMOOSApp. If you have an application that does not subclass this superclass, it is still pretty easy to add this to your class.

To add the ITER_LEN, and ITER_GAP information, follow the example of the pseudocode below. The code requires some memory between iterations of the timestamp of the prior iteration. So the code requires the additional member variable, added to the YourApp class definition, m_last_iterate_time, initialized to zero in the constructor.

*Listing 6.5: Adding appcast output for uLoadWatch.*

```
YourApp::Iterate()
{
  // Note the time the iterate loop started
  double start_time = MOOSTime();


  (The prior guts of your Iterate loop)


  // Note the time the iterate loop ended
  double end_time = MOOSTime();

  // Only report iter_gap, iter_len if app frequency is > zero
  double app_freq = GetAppFreq();
  if(app_freq > 0) {
     double interval = 1 / app_freq;
     string app_name = MOOSToUpper(GetAppName());89

     double iter_len = (MOOSTime() - start_time) / interval;
     Notify(app_name + "_ITER_LEN", iter_len);

     double iter_gap = (start_time - m_last_iterate_time) / interval;
     if(m_last_iterate_time != 0)
       Notify(app_name + "_ITER_GAP", iter_gap);
     m_last_iterate_time = start_time;
  }
}
```