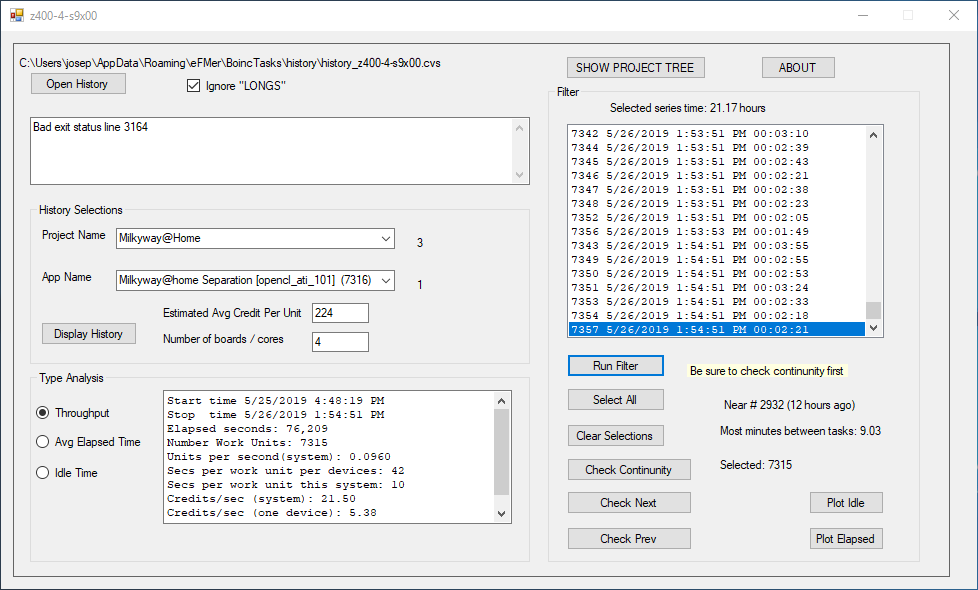
Boinc Tasks History Reader “BThistory”

Fred has graciously allowed my program BThistory to be promoted as an add-on to Boinctasks.

BTHistory reads one or more Boinctasks’s history files and allows data analysis for elapsed time, throughput and idle time. If more than one file is opened, then comparisons can be made between different systems. New or unknown applications are reported, highlighted and can be compared. The program is written in C# and compiled under Visual Studio 2017. One can download the executables or build the sources at the temporary location GitHub/BeemerBiker. Additional utility programs are included in the VS2017 solution and are explained below.

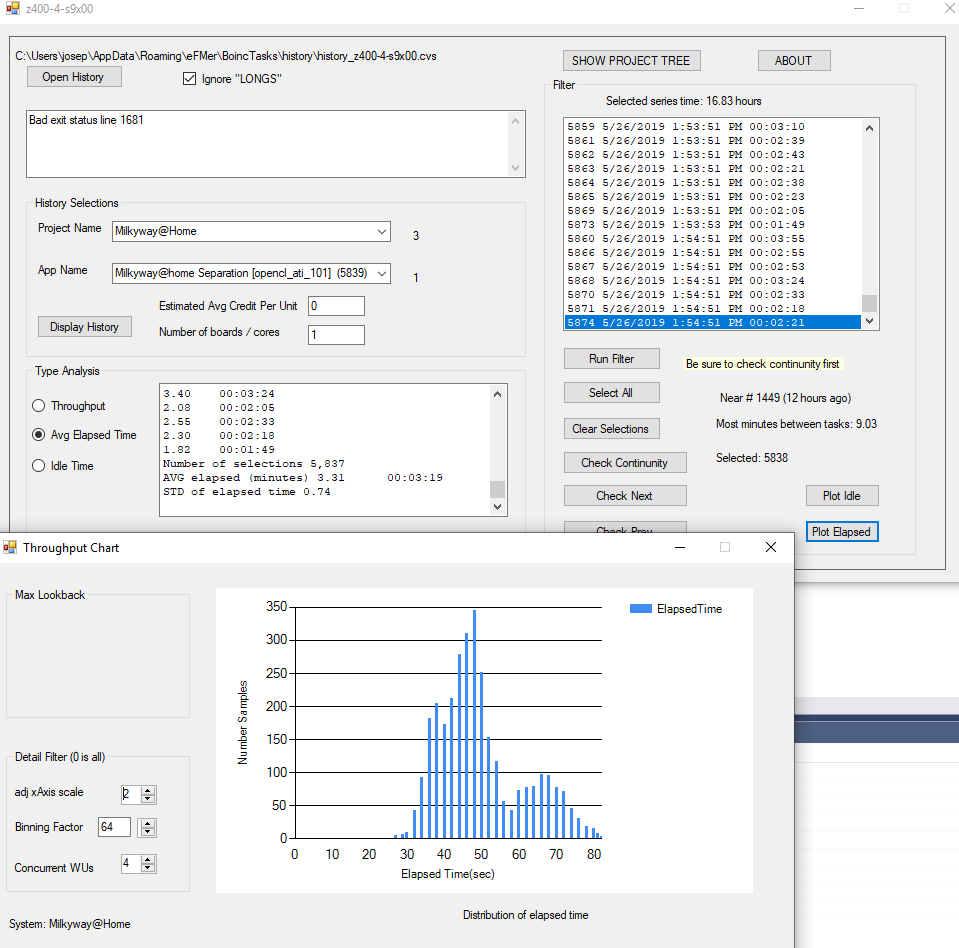
1. BTHistory main form and throughput analysis



The history file “z400-4-s9x00” has been opened and there are 3 projects. That can be selected (milkyway) and that project has only one app running on this system. The throughput filter is selected, all results were selected (except 3164 which had a problem) and the continuity check was done. Knowing that this system had 4 boards and average credit was 225 points, the results show about 5 credits per second.

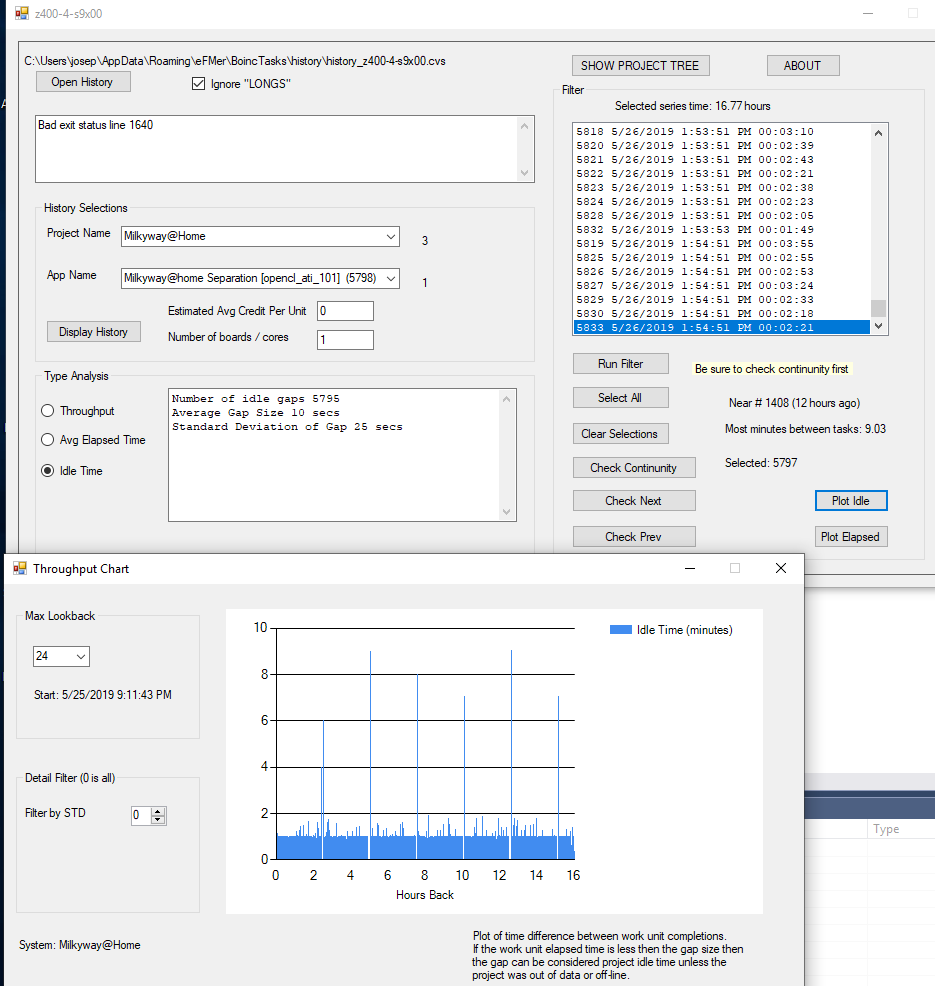
The “1” adjacent to the App Name box indicates there is only 1 app associated with Milkyway, at least on this particular system.

1. BThistory and Elapsed Time



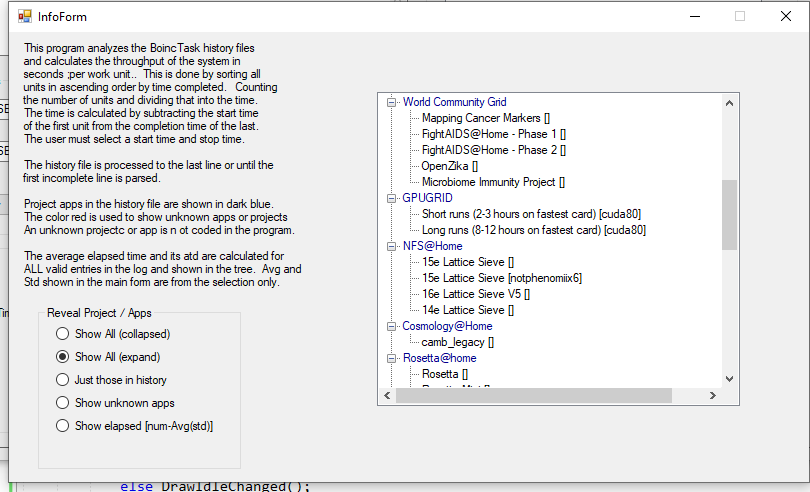
Elapsed time is in minutes, but the plot parameters were changed to show the effective ET since four tasks were being run concurrently each GPU. If there were 5 GPUs that mean of 50 seconds (as shown above) indicates the system with 5 GPUs will produce about (50 / 5 = 10 seconds per work unit)

1. BThistory Idle time



The Idle Time analysis is useful to show when projects run out of data or, as in the case above, the project fails to provide data until all the data has been processed. In that case, where the system is waiting for data to arrive, the gap is considered idle time. The above data shows that about every two hours there is a 6 to 9 minute gap before the server provides data.

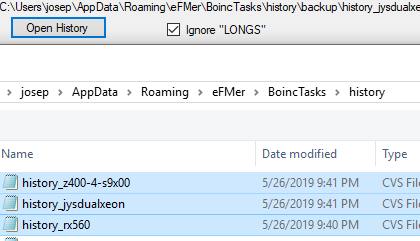
1. BThistory Project Structure



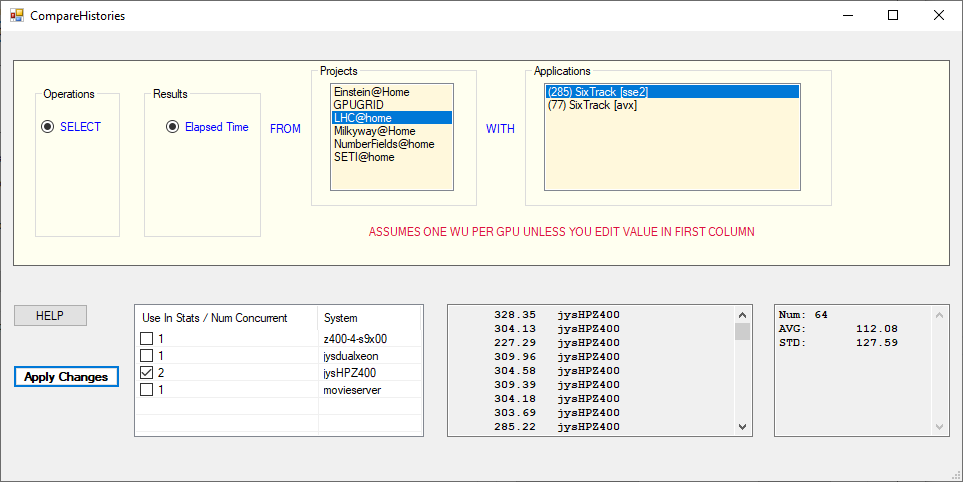
This shows which projects are in the BThistory database or are in the history file

1. BThistory Select multiple systems

If more than one history file is opened, then the BThistory produces a a comparison of the different systems. Typically the files of interest end in CVS and do not have phrase “\_long\_” in the filename. In the event that the “long” files do contain data then you should uncheck that exclusion.



5.1 BThistory Comparison



This feature allows comparison of systems across the same project and app. Currently only SELECT and REPORT are the only defined operations. You can use this tool to compare, for example, the computation of SETI using NVidia or AMD boards. Once the project and app are selected only those systems will be shown. Select (example above shows sse2) and the system you want to use in the comparison and then copy the results into notepad. Then select the avx app and the system that is desired and you can examine the statistics and compare to what you saved in notepad.

6 Other programs peripheral to BThistory

The BThistory program resides at

<https://github.com/BeemerBiker/Gridcoin/tree/master/BTHistoryReader>

However, the solution is at GitHub/BeemerBiker/GridCoin which will cause the following programs to be built in addition to BTHistory. All programs listed were built using VS2017 C# with the exception of the RPC library which is only used by BoincRpc.

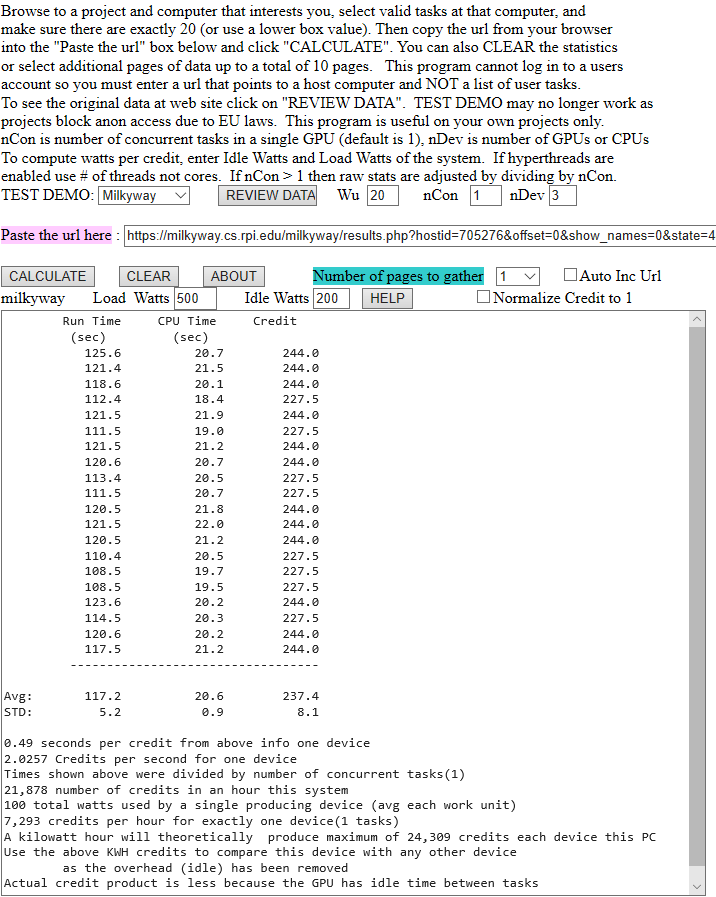
6.1 HostProjectStats

The is an aspx program that creates a web page. It can be compiled and executed on your windows system or you can run the program using most browsers by clicking on the link below

<http://stateson.net/HostProjectStats>

This program obtains elapsed time information from most Boinc Projects and, if you optionally know the load and idle wattage, it will calculate the average credit and wattage used to produce those credits for the system and the boards. This program requires that the data be available so it may not work with anonymous access unless the project had allowed it for the specified HOSTID.

As shown below, the project Milkyway has been browsed to and hostID 705276 selected. This is actually one of the top systems and is listed by default when first bringing up the program.



6.1.1 HostProjectStats default form.

6.2 Host Project Stats Help

The following text comes from the help form for this aspx application

**This program can be used to compare different GPUs and CPUs and obtain the cost in KWH you are contributing to the project.  
For this program to work you must be able access a BOINC project.  If you have enabled "Remember Me" on your project account  
then, except for WCG, you can access your own data.  Some projects have not restricted anonymous access so possibly you  
could access the data without signing in.  Normally, "Top participants" have allowed their statistics to be searched.  
Auto Inc Url:  automatically adds 20 to offset of url when checked.  Allows one to see 20 averages at a time  
Normalize Credit:  Make all credit 1.0 and adjust the run times proportionally. Useful for problem spotting.  
  
When this program starts up, Milkyway is selected and one of the "Top Computers" is preset into the URL.  If Milkyway is online  
then clicking CALCULATE will provide a demo for accessing statistics.  Click REVIEW DATA to see the actual data.  
If Milkyway is offline one will get an OOPS error as I don't do a lot of error checking.  If you wait too long then my website  
times your session out and OOPS shows up.  
  
To calculate the number of watts it takes to do a credit (cost of your contribution to the project) you must know how many watts  
your system is drawing.  That can be obtain using an inexpensive A/C ampmeter or wattmeter  
  
Load Watts:  number of watts the system is drawing (BOINC projects running)  
Idle Watts:    watts your PC uses when BOINC is running but not the projects (idle)   
nDev:            Number of GPUs being used by the project -or- number of threads being used  
                       Do not run CPU tasks at same time as GPU tasks and vice-versa when testing.  
nCon:            Number of concurrent GPU tasks. Set to 1 for any CPU testing  
Wu                Work units to obtain (leave at 20 unless you do not have that many)  
                        Note that WCG uses 15 on a page.  
  
Notes   
  
(1)    WCG cannot be accessed because they required username / password to be stored in a cookie and this program  
                        does not implement that scheme.  I tried using WebClient but failed.  
  
The following two notes assume 120 watts idle and 420 watts full load with 3 GPUs and only 1 task per GPU  
and A single watt is a joule of energy that is given up in 1 second of work.  
  
(2)    Example of computation of credit per second entire system.  
        Assume you have 3 devices and it takes 10 seconds to get a single  
        credit.  Your system then generates 3 credits in 10 seconds which  
        is 0.3 credits per second.  Multiply that value by the number of  
        current tasks each device is running (just 1) total of 0.3 credits.  
        If the system is run for a full hour at 420 watts then 0.3 \* 3600 gives  
        1080 credits per hour.  The 420 watt-hour can be converted to KWH  
        by dividing by 1000 for a total or 0.42 KWH to generate 1080 credits  
  
(3)    Computation of KWH (cost of contribution) for each GPU (or CPU)           
         Assume your system uses 120 watts at idle and 420 watts at full load  
         and has 3 GPUs working with one task per GPU.  We wish to calculate  
         how many credits in singe KWH on just the GPU:  
  
The 3 GPUs each consume 100 watts. Doing the same calculation as  
above, if the system is run for a fully hour at 100 watts then multiply  
0.1 credits by 3600 to get 360 credits for that 100 watt-hours.  
We can now calculate how many credits in a kilowatt hour by multiplying  
by 10.  This give 3600 credits per KWH per GPU.  This value does not  
include any idle wattage and can be used to compare to different GPUs  
Note that this is equivalent to powering the system for 10 hours as it will  
take 10 hours to utilize 1 KWH on each GPU.  
  
(4)     If concurrent tasks are run then I assume they are actually concurrent  
         and divide the run time by the number of concurrent tasks.  Theoretically  
         this is correct but in actuality just an approximation that gets worse as  
         more tasks are run supposedly concurrently.  I also do not account for the  
         number of invalid tasks nor the idle time between tasks which increases  
         as one attempts to run more concurrent tasks.  If the RunTime on  
         concurrent tasks is greater than the RunTime for a single task then  
         there is no benefit.  RunTime is divided by number concurrent to  
         make it easy to compare.**