# MyAES 0.98 ( July 2021)


A page dedicated to version 0.98 because I think this version is a major update without bringing any big new features.

After versions 0.96 and 0.97, which had already been versions with big internal changes to make MyAES work better, to reduce memory usage, to work on resolutions below 256 colours, sometimes to speed up the kernel and sometimes unfortunately the opposite.

Here version 0.98 is an in-depth revision of the event management kernel which essentially aims at its performance. To achieve this, many tests and benchmarks have been carried out, if the previous versions were approximately at the level of a TOS or NAES, it was still quite far from XaAES or better from Magic.
The stated aim is clearly to compete with these two AESs in terms of responsiveness, and this is what we will see.

Of course version 0.98 brings its share of bug fixes that I won't list, many for the display, but also some new features.

## What's new


Support of thread creation by AES à la mode Magic via shel_write() it is thus to my knowledge the first AES supporting threads under Mint, I recall that this thread shares the memory area of the client, it has its own stack managed by the calling application, the calling application and the thread work in a concurrent way as 2 independent programs, it is thus very different from the gemdos functions Pfork() and Pvfork() provided by Mint, it is thus imperative to have a "safethread" code and to use libraries capable of that if the code used possibly can be used in a concurrent way like the functions mt_aes to call VDI and AES

Option "app_single" is an option which can be useful with the rather old applications which like to monopolize the mouse and were generally conceived under TOS, this option makes it possible to have only this application with the screen like under TOS, this option is to put typically in the file app_conf.cnf file followed by the name of the application to be managed, for example "app_single WORDPLUS" the wordplus application will be launched in this mode, applications of this kind include Pure Pascal, Degas Elite, Gembench (for a more accurate comparison with TOS, gembench can work without any major problem in a multi-application environment)



Option "focus_priority", allows to choose between 2 modes of operation of the AES, if the value is false then no preference is made between applications, this mode is similar to the other AES, if the value is true then there is priority to the application that is in the foreground, the other applications are slowed down voluntarily in the AES (outside the priority is not modified), MyAES considers that the application in the foreground must have the maximum of available resources and that a background task does not have to load the system excessively. By default this variable is true, this can be set in myaes.cnf: "focus_priority=true".

## Technical improvements

MyAES 0.98 fixes a number of bugs, but mainly improves speed:

- Mainly speed of the kernel that handles events, it has been deeply modified, which has led to a long absence of updates to stabilize this new kernel, on some extreme cases it can be more than 50 times faster than the previous version and more commonly about 2 times, The idea of the improvements is mainly to not have any important slowdown by the AES for the management of the events during an intensive use typically for games or GEM video players style, even if in my opinion the best way is the use of a dedicated thread via shel_write(MT_THREAD....) so far only supported by Magic and present in this version. I have created some specific tests to study this kernel, these results are presented later. MyAES is probably the fastest AES in this

field on 68030 at 32Mhz, 68060 and faster, in fact it stands out the faster the machine is, I could not make a comparison on 68000 8Mhz machine for lack of configuration machine for testing, it will anyway be significantly faster than TOS or Emutos, note however that its use is not trivial, MyAES behaves like a server for the most part which means that a small part of the resources are taken even if the AES is not used, if this remains modest on a 68060 we note that the more modest the machine the more important this proportion is so we can estimate that in terms of loss of power linked to this mode is around 7% on 68030 at 32Mhz and around one percent on a 68060.

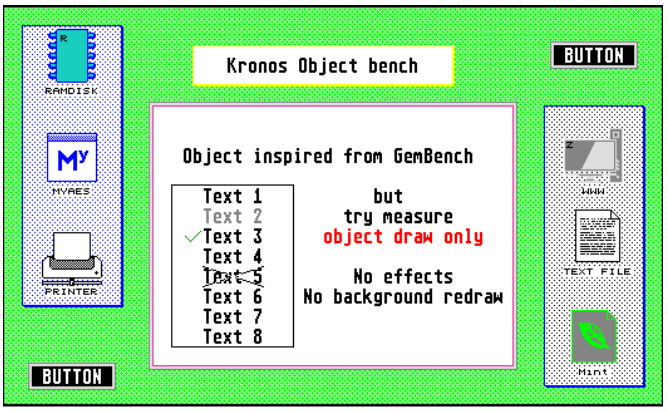
- Slight speedup in the GEM object display routine (objc_draw), according to my tests it is equivalent to the best AES in the field but it is true that a large part of the execution time is done in the VDI in this case.

# Speed comparison

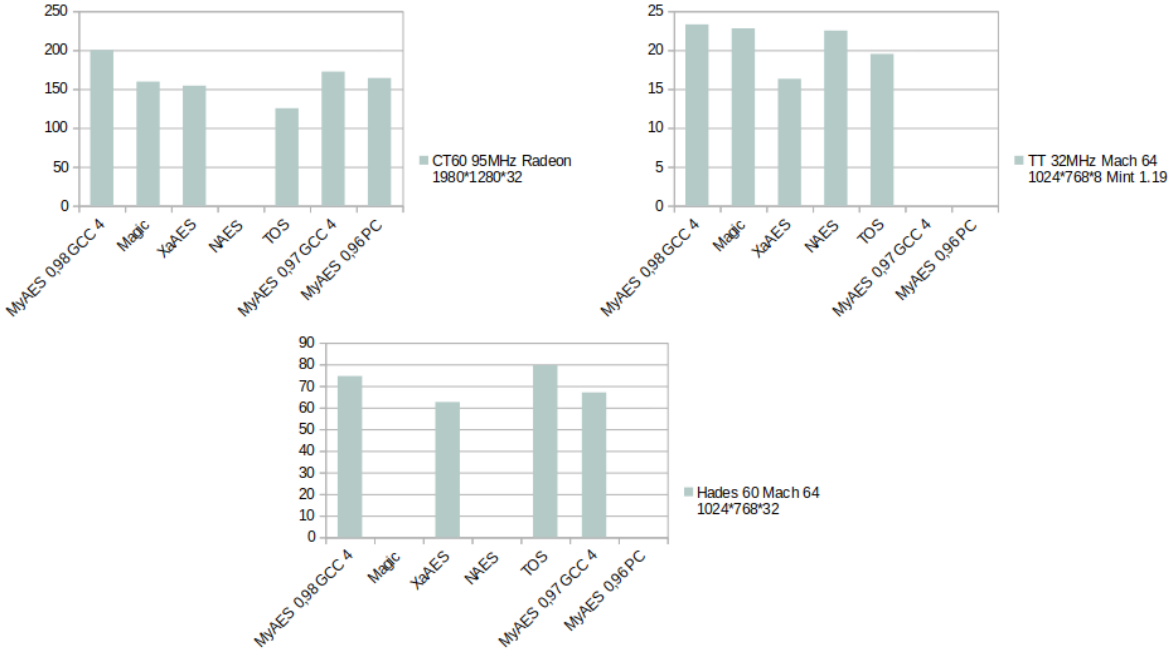
In order to perform the tests, a new version of Kronos has been released, it integrates the display of a GEM dialog box via objc_draw() and 3 event_multi() tests that are supposed to return immediately, one of the tests is the method officially used by SDL for the GEM driver and uses simultaneously appl_write and evnt_multi.
**If any values are missing it is because the AES has not been tested on the machine**, see nothing else, I would particularly like to thank 2 people for these tests: Mr Piotr Mietniowski who has a considerable number of different configurations (CT60, Hades 60, TT30) and Mr Guillaume Tello (TT30 and CT60), I noticed considerable differences in speed between Guillaume's TT and Piotr's, the configurations are very different, on each graph I try to present the results of only one machine at a time so as not to put forward results that could lead to erroneous conclusions, As far as Mint configurations are concerned, only the AES changes, when it comes to Magic I cannot guarantee that the configuration is totally equivalent, I have had results over time that can be fluctuating, I have kept by default in the case of Magic the best results.
I compiled MyAES with 3 different C compilers (PureC, GCC 3 and GCC 4), depending on the machine, the configuration of each of these 3 configurations can be faster than the others, on Aranym JIT typically the GCC 3 version will be the best followed by PureC and GCC 4, on Piotr's machine GCC 4 is the most efficient, while on Guillaume's TT with Mint 1.12 the PureC version is the fastest, the best thing is to test with Kronos, but in all cases MyAES 0.98 gives excellent results.

## ▸ Test display objc_draw()

This test counts the number of dialog boxes displayed per second, the dialog box displayed is the one below.




## ▸ Comparison objc_draw()

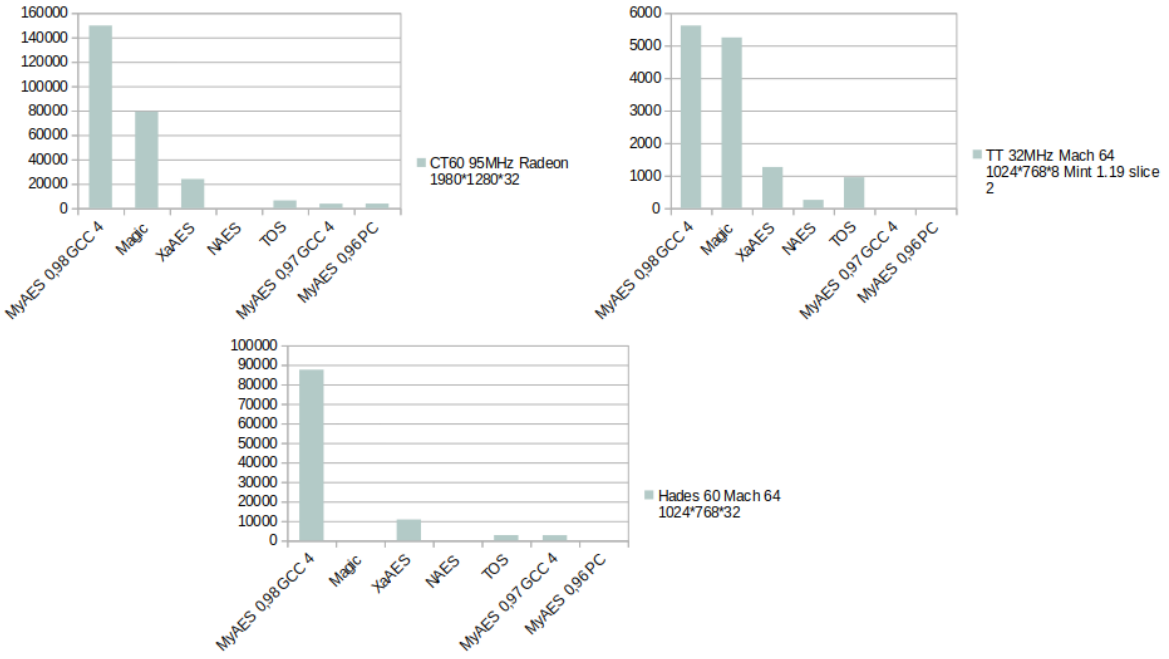



*Test obj_draw, number of dialog boxes displayed per second*

From one AES to another, the differences in display speed remain fairly small on the slower display machines and more variable on the faster display machines without being able to draw any definite conclusions. Most of the time spent on display is in the VDI routines not modified by AES. MyAES is always close to the best result, on CT60 it is the fastest (no reference with NAES), on TT it is equal to NAES and Magic, on Hades 60 the TOS is the fastest, XaAES is slightly behind, the TOS depending on the machine and the version has more variable results. Note that MyAES 0.97 and 0.96 do not display correctly the colours of the monochrome icons if they are not in black and white

## ▸ Comparaison of evnt_multi response time with the requested timer at 0 and message search

This test shows the number of calls that can be made per second by calling the evnt_multi routine with a timer limit of 0 ms + message search.




*Test evnt_multi(MU_MESSAG+MU_TIMER(0)), number of calls per second*

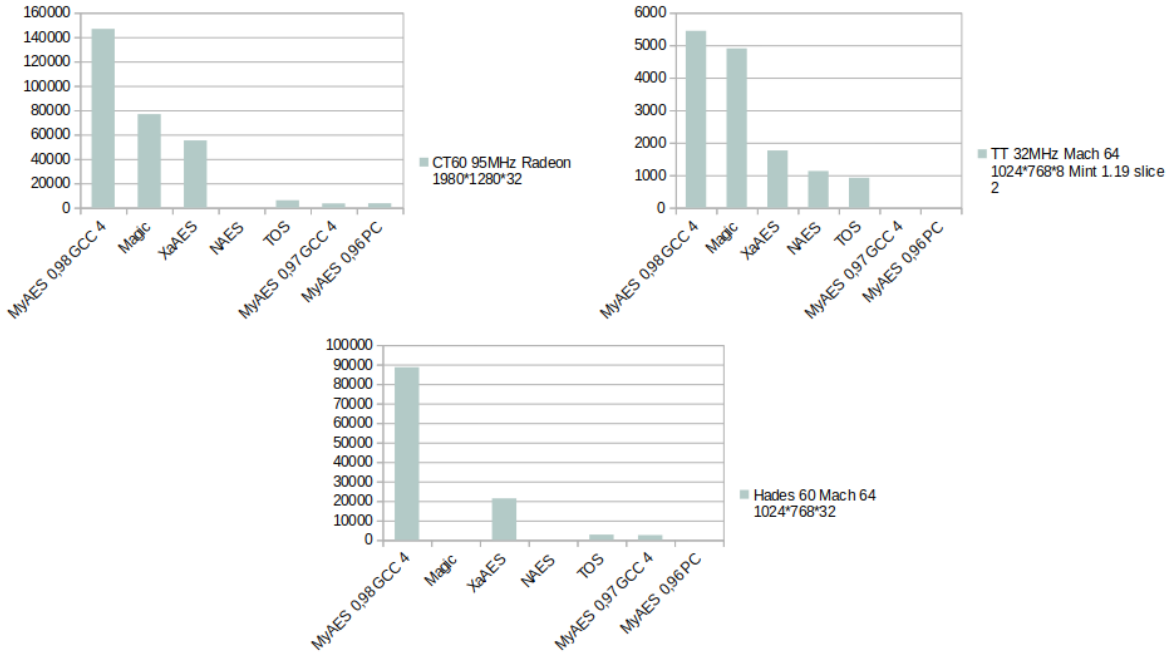
Magic was until now the reference for the speed of processing the verification of the presence of a message with a request for immediate return if there is no message thanks to the imposition of a 0ms "timer" event. We can see on TT that MyAES 0.98 is very slightly faster than Magic, on 68060 the gap widens considerably. We can see the huge difference between the 0.98 version and the

previous versions, which are particularly inefficient with performances in the order of TOS or even worse. XaAES is not particularly efficient on this test even if it is significantly better on 68060. NAES (only tested on TT) is particularly slow, much slower than TOS, which is probably why the GEM version of SDL has abandoned this method.

## ▸ Comparison of evnt_multi response time with mouse click released + timer 0 + message search

Normally if a mouse click in the released state is requested then the evnt_multi routine should return if the mouse is in that state, there is some interest in adding this request as we will see below on the results. This test shows the number of calls that can be made per second.

This mode of operation is used by the SDL library that I patched, the sources of this version are available on http://ol.lutece.net/ .

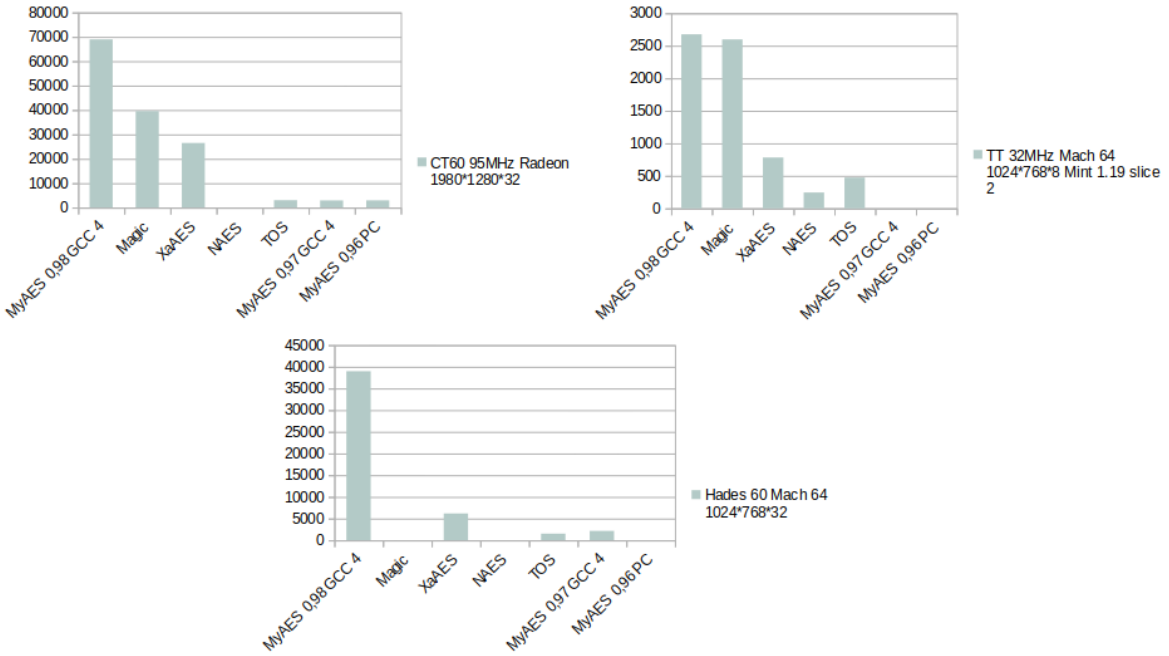



*Test evnt_multi (Mu_MESSAG+MU_TIMER(0)+MU_BUTTON), number of calls per second*

As curious as it may seem, XaAES and especially NAES see their performance improve compared to the previous test, while MyAES 0.98 and Magic see their performance decrease slightly, but these two AES remain significantly faster, especially MyAES 0.98 on 68060

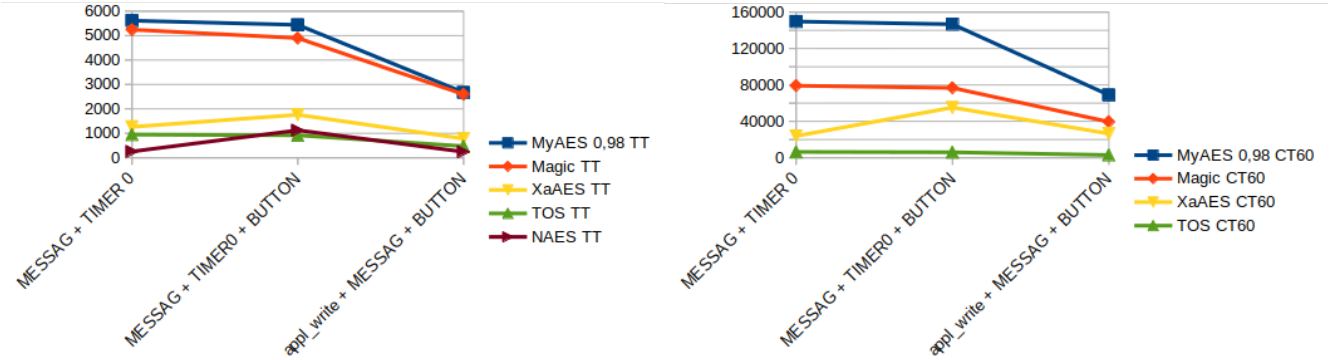
## ▸ Comparison of the response time of the AES requested by appl_write + evnt_multi (MU_MESSAG + MU_BUTTON)

This test displays the number of calls that can be made per second when the application sends a message to itself and then asks to search for this message using evnt_multi(). This mode of operation is realized by the nice SDL library port in official version written by Patrice Mandin.



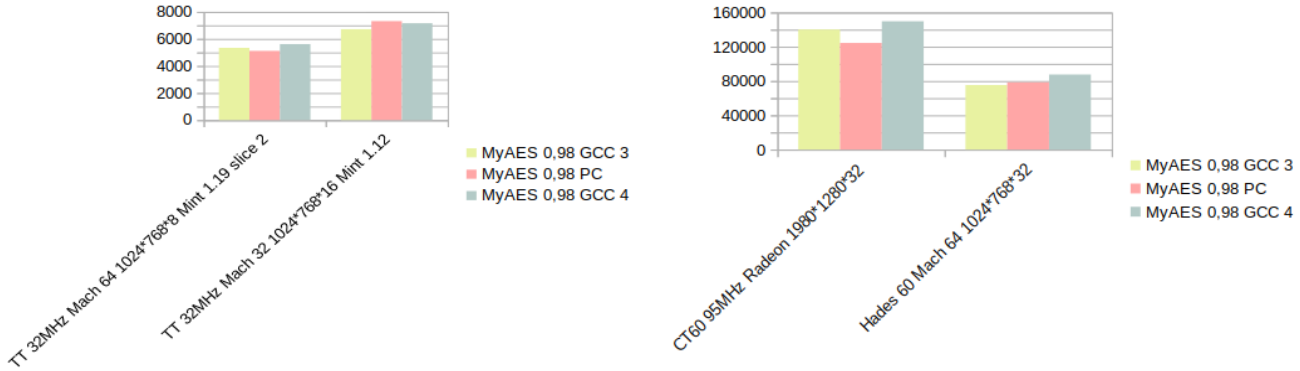

*Test appl_write + evnt_multi(MU_MESSAG+MU_BUTTON), number of calls per second*

In view of the results, I don't really see the point of such a method, MyAES 0.98 and Magic are significantly more efficient but their performance is still divided by about 2, as can be seen on the image below:



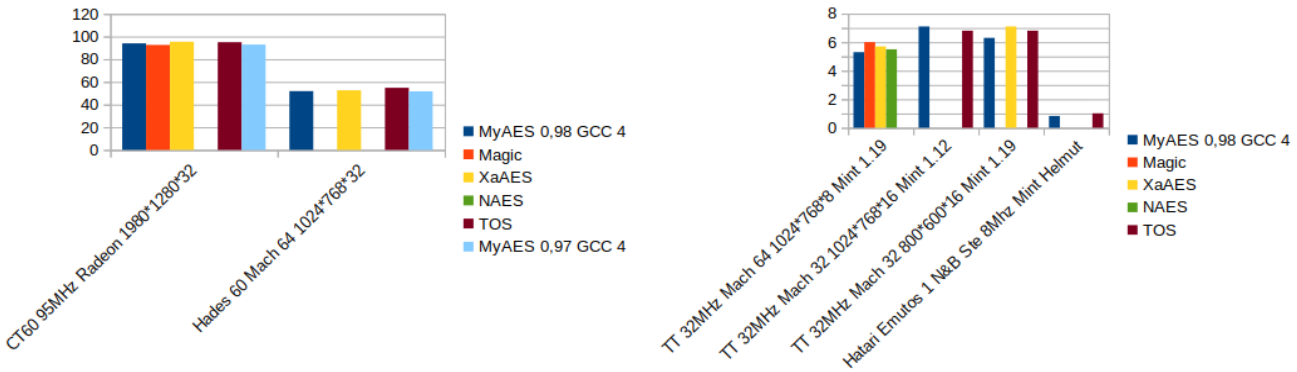

*Comparison of the 3 methods of using evnt_multi, number of calls per second*

MyAES offers identical versions compiled with 3 different compilers (PureC, GCC 4 (the only one able to produce native code for coldfire processor) and GCC 3), following these tests and some others, the choice is not obvious as we can see on the following picture, it is possible that some parts of AES prefer a compiler rather than another in terms of performance, here we focus on the event part regarding the responsiveness.
Here we compare the evnt_multi test with a timer at 0 on two different TTs with different compilation of MyAES 0.98, depending on the case, no compiler dominates, the differences if they are not negligible remain generally of a relatively limited impact.




*Test evnt_multi(MU_TIMER+MU_MESSAG), compiler comparison, number of calls per second*

MyAES is presented as a server, so we can ask ourselves the question of the load of this one on the reduction of the computing capacity of the machine, as well as the use of a multitasking system compared to a non-multitasking system like TOS. To try and understand this problem we will use a result from Kronos which calculates an Opengl scene in 24 bit offscreen and purely 68000, during this calculation there is no access to the system by Kronos, if we compare the results we should have an idea of this load.

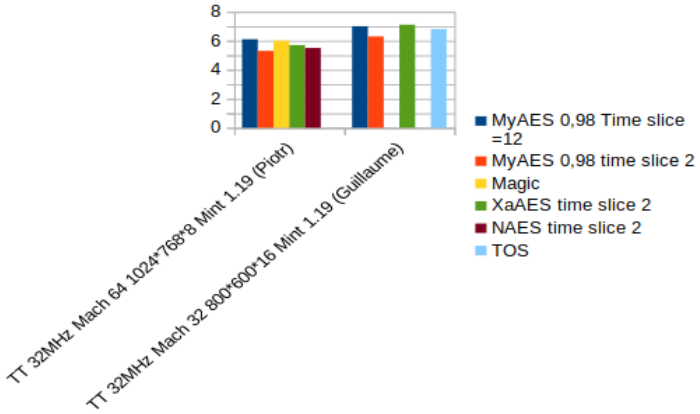



*Comparison of the performance of the calculation according to the configuration used*

Under 68060 either on CT60 or Hades 60 the results are very close between the configurations of the order of +-1.5% difference between the multitasking systems, on the Hades 60 the SWR results are higher than MyAES 0.98 by about 6% (maybe due to the services on the way on the Mint configuration)

On TT 32Mhz with Mint 1.19 or Magic, MyAES 0.98 is noticeably slower by 8 to 13% compared to XaAES or Magic, whereas under Mint
1.12 the TT seems much faster and MyAES is about 4% faster than the TOS. Finally under the Hatari emulator with an 8 Mhz STE emulation MyAES 0.98 with Mint "Helmut" loses 22% of power compared to TOS but if you use Mint 1.19 the drop is 50%!

A small research allowed us to determine an important parameter to return to a speed equivalent to the Mint "Helmut" version, it is enough for that to define in mint.cnf: KERN_SLICES=12 (manages the time between the changes of task) with that setup the results are very close to the "Helmut" version, the real time remains completely usable, this same modification allows on TT with Mint 1.19 to have an availability equivalent to Magic, this point remains to be studied during a next version for modest configurations, in the meantime you can use this parameter if you want to use MyAES 0.98 without losing performance.




*Impact of the KERN_SLICES parameter on the computer's computing performance*

It can also be noted that a multitasking system does not necessarily cause a slowdown compared to the TOS and that sometimes performance is improved!

# The future

I think that on the subject of the speed of the event handler the development is closed except bug or encountered problem except that it would be interesting to reduce a little the CPU load on 68000.

*The areas of improvement still envisaged are:*

- Reduce the processor load on smaller configurations
- Reduced memory requirements in 68000 version
- Bug fixes among other things some applications can't make the editable fields of objects work (like Devpac) if you have bugs to report you can report them on Github https://github.com/Landemarre/MyAES/issues
- Support new resource file format
- .....

Thank you for your attention

Olivier Landemarre