Acceleration of PIAS by hardware support

May 2017

Some tasks of PIAS can be rather computation-intensive, such as the computation of intact stability (in particular if enhanced features are active, such as the shift of liquid method, and/or stability around the weakest axis) and damage stability, but also the generation of curved surfaces in Fairway. Time was that each new computer generation was faster than its predecessor, mainly thanks to processor clock frequency increase, but that has come to a halt a decade ago. Nowadays, CPU manufacturers try to stimulate performance gains by means of parallelization, so that multiple tasks can be executed simultaneously. Unfortunately these facilities do no offer free performance increase, on the contrary, the application software should be restructured to benefit from them. By 2005, some parts of PIAS have been restructured to benefit from the dualcore technology of those days, resulting in PIAS' dualthreading feature. Recently, PIAS has been further enhanced with support for the latest processor technologies. The *modus operandi* and some examples of performance gains are the subject of this document.

Definitions

- CPU. A CPU is the Central Processing Unit, it performs all operations and is the heart of the computer (in combination with memory). A CPU contains one or e more cores. And a computer contains one or more CPU's.
- Core. A core is an independent processing unit of a CPU. So, each core supplies computing power.
- Thread. A thread is an independent program part. So, each thread demands computing power.
- Multithreading means that a computer program splits itself into multiple threads, which can be executed simultaneously. So, multithreading comes from the demand side; each independent thread can run on a separate core, so multithreading requires more computing power from the CPU. For the benefit of increased program speed.
- Hyperthreading is a hardware technology that virtually splits each core into two. This is said to deliver on average some performance gain, but will not double the CPU's performance potential, because in reality the number of cores remains the same. For that reason hyperthreaded cores are reported by Windows as *logical processors*. The term *hyperthreading* is a bit confusing, because this technology operates on the supply side, so it is not related to the term *thread* in its usual meaning, on the demand side.
- Thread pool. Windows supports the use of multiple threads from its '95 version. However, this implementation had quite some overhead, where a part of the multithreading performance gain was lost. In later versions Windows offered a faster, but different, facility, called a *thread pool*.

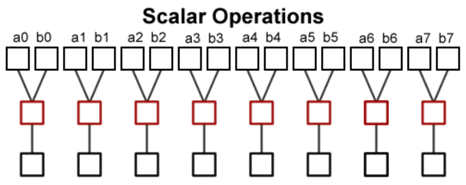
Multithreading in PIAS

It will be obvious that it will not be possible to make a computer program multithreaded by the touch of a magic stick, because tasks that are suitable for parallel execution have to be individually identified and multithreaded. And quite some tasks are by nature not fit for parallelization. E.g. an iteration algorithm, where each step is dependant from the results of the previous one, so the iteration steps can logically not be parallelized. Another example are the intermediate stages of flooding in damage stability, which can only be run *after* the final stage has been computed. So, intermediate stages cannot run simultaneous with the final stage of flooding.
Nevertheless, in quite some places in PIAS computation-intensive have been parallelized, such as:

- Deep down in PIAS some numerical integration procedures, where the determination of intersections and areas of frames are performed for all frames simultaneous.
- Computation of intact and damage stability for all angles of inclinations simultaneous.
- Tank sounding tables for all tanks simultaneous.
- Determination of damage case boundaries in PIAS' probabilistic damage stability module (Probdam) for all damage cases simultaneous.
- Computation for curved surface patches in Fairway, for all patches simultaneous.

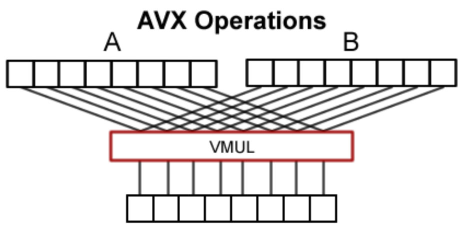
This list will be extended in the course of time. From now, in PIAS' software development parallelization will be applied where possible and opportune. By the way, in reality, the word "all", in the enumeration above, should be relaxed a bit. It would indeed be possible for a probabilistic damage stability calculation containing e.g. 500 damage cases, to create 500 threads and to offer them all to the CPU. However, in that case the CPU would spend many resources just to manage all these tasks, so it proved to be more efficient to limit the number of threads to the number of cores.

AVX support in PIAS



Besides multithreading, modern CPU's are equipped with yet another facility to give it more oomph, which is called AVX: Advanced Vector eXtensions. The background is that for a CPU arithmetic with real numbers are relatively time consuming. Notably multiplications and divisions take, relative to addition and subtraction, quite some time. This bottleneck is tackled by AVX, which can do these operation parallel for multiple numbers. The usual operation is depicted in the top figure right, where multiplying eight pairs of numbers takes 8 CPU cycles. With AVX, see the bottom figure, those eight multiplications can be performed in just one cycle.

Where applicable and opportune AVX will be used in the core of PIAS. A program as PIAS does not operate with massive arrays of elementary arithmetic, so it is not realistic to expect double or quadruple processing speeds just because of the application of AVX. Nevertheless, AVX offers some performance in PIAS, and it would be a waste not to benefit from it.



PIAS/ES

Applying multithreading is as a matter of speaking more a matter of art than of science, for the optimal solution can only be found for each individual application, by trial and error, assisted by some sound reasoning. For that case PIAS offers two speed enhancing packages:

- PIAS/ES 1, with original Windows threading facilities, limited to two threads (*dualtreading*).
- PIAS/ES 2, containing AVX and application of Windows thread pool technology. Optimized for 4 to 8 threads, limited to 8 threads (hence its name *octothreading*).

Speed gains in practice by applying PIAS/ES depend on quite some factors, such as processor type, nature of PIAS model (notably number of frames and compartments), and to a lesser extent networking and disc speed. A special word of warning has to be raised when applying a hyperthreading CPU, for this processor type has only **virtually** doubled its number of cores. As indication, below some measured timings (in seconds) are listed for Probdam computations (with its compartment method).

CPU	Generation of damage cases			Compute damage case boundaries and damage stability		
	Conventional	PIAS/ES 2	Ratio	Conventional	PIAS/ES 2	Ratio
Desktop, Intel i5, 3.3Ghz, 2012, 4 cores	105	28	27%	11245	3030	27%
Desktop, Intel i7, 3.6 Ghz, 2015, 4 cores hyperthreaded	92	22	24%	10041	2369	24%
The same as above, hyperthreading disabled	91	28	31%	9734	2823	29%
Laptop Intel i7-6500U, 2.5Ghz, 2017, 2 cores, hyperthreaded	111	47	42%	12065	5475	45%
Desktop Intel i9-7900X 3.3 Ghz, 10 cores hyperthreaded[1]	70	9	13%			

These timings are for a common PIAS ship, with damage cases up to 8 simultaneous damaged compartments, resulting in 525 damage cases, without so-called “external subcompartments”[2].
PIAS users interested in the gains of these new technologies in practice are advised to experiment a bit with their own vessels on their own hardware. SARC can facilitate this by supplying a PIAS/ES 2 version for a free evaluation period.

A final remark can be made on the experience that if PIAS is using all computer resources for its computations, other processes (such as email or MS-Word) might become a bit unresponsive. That effect can be relaxed by the *Maximum number of processors to be used by PIAS* switch in PIAS’ Setup.

Future developments

For the kind of performance issues as discussed here, no generic solutions or methods exist. Everything is a matter of experimenting and tuning on the particular hardware of the day. And solutions are not scalable; what works efficiently on a 2 or 4-core processor, might be much less efficient on a 16-core CPU. For that reason in PIAS/ES, multithreading is limited to 2 or 8 cores respectively. If future hardware developments for desktop PC’s tend towards 16 or 32 cores then PIAS will be tuned again on such platforms. Another future option would be to apply cloud-based computation servers, as offered today for e.g. CFD applications. There are some potential pitfalls in this solution — just substituting internally the number 8 by 128, if a cloud offers 128 cores will certainly not do — but at SARC we will keep a sharp eye on these developments.

[1] Included in this list August 2019.
[2] In the multithreaded execution of external subcompartments —which are processed with a more complex algorithm than ordinary subcompartments — there is still some room for a bit further optimization. Implementation is scheduled for the second part of 2017.