Parallel Processing in APL+Win

The implementation of parallel processing capabilities in APL+Win is a 'work in progress'. The focus is to make it possible for the APL+Win application system programmer to directly use available hardware for parallel processing when it is deemed appropriate by that programmer without negatively impacting the performance of the APL+Win product for other users.

The ability of an APL+Win application system programmer to use parallel processing by allocating application system level 'work' (i.e. processing covered by an APL+Win user-defined function) to multiple machines or multiple CPUs is currently well supported by APLNext WebServices and APLNext Supervisor respectively. These 'out of process' implementations have many benefits including:

- Selective application of up-to-date technology based on application programmer choice
- No ill effects on existing application systems that do not benefit from the technology, because the feature is not imposed upon all programmers using APL+Win
- Superior flexibility and control options for the application system programmer, such as identification and allocation of hardware resources (e.g. min/max pool of machines or cores), process termination, exception handling and processing status feedback
- More memory and other machine resources simultaneously available to the parent APL+Win process
- Use of current technology inherently available in the operating system instead of 're-inventing the wheel' in APL, for example in the APLNext Supervisor implementation the operating system manages the assignment of the processing done by multiple threads to the number of programmer-selected CPUs.
- Extremely high performance 'out of process' interfaces exist in APL+Win such as the ActiveX interface

The next step in this effort will be to expose the mathematical and logical processing potential of graphics processing units (GPUs) to APL+Win application system programmers. This implementation has design objectives analogous to the APLNext WebServices and APLNext Supervisor components. Since GPUs generally support important basic operations, the use of multiple GPUs or CPUs has potential application to the 'primitive' APL functions and operators.

Consistent with previous implementations of parallel processing technology in APL+Win, this next step will use an 'out of process' methodology. The 'out of process' implementation means that the APL+Win application programmer must specifically invoke the tool instead of the APL+Win interpreter 'deciding' where the tool should be used. Initially, in order to achieve significant performance improvements, the technology will probably need to be applied to the application system as part of an APL2000 training and consultation process.

The initial 'out of process' implementation of multi-CPU/GPU processing for the 'primitive' APL functions and operators does not anticipate 'an automatic benefit in performance without requiring any APL re-coding'. Some reasons for this condition include:

- To maintain programmer choice in the use of the technology, the programmer must incorporate indicators in the application-level code when parallel processing is to be considered.

- Automatic performance benefits cannot be assured because specifying parallel processing globally is unlikely to yield overall performance improvements since low-level functions and operators may act upon arguments of varying characteristics.
- Incorporating selection criteria into existing low-level functions and operators of APL+Win to determine if parallel or serial processing should be used may significantly disadvantage those application systems which could not benefit from parallel processing. In order to 'do no harm' to the APL+Win product performance, it is important to avoid the intrusion of such switches and tests 'inside of the interpreter execution loop' whenever possible.
- Criteria to select parallel or serial processing have not been well established for low-level functions and operators. The best available method has been to 'try it both ways' which suggests that application system level programmers are best suited to this type of performance 'tuning' rather than hard-coded analysis added to low-level functions and operators in the APL interpreter.
- Customers have provided limited information to APL2000 about which (and how many) existing APL+Win application systems might benefit from parallel processing
- Overall application system performance is often more dependent on the high-level way the application process was programmed rather than on individual, low-level function and operator performance.
- Parallel processing CPU/GPU toolkits for the Windows operating system environment are now available with a proven record of performance benefits and these tool sets are inherently 'out of process' with respect to APL+Win.
- APL+Win is a general purpose programming language, whereas research performed by the IT community indicates that parallel processing performance benefits are applicable only to carefully selected application systems designed in an appropriate manner.
    - Operations processing data sets (e.g. arrays) that are not sufficiently large yield poor parallel performance results because the overhead in invoking parallel processing, even before data is marshaled to a CPU/GPU, is not justified by the work to be done on a small data set.
    - Operations processing data sets which are too large yield poor parallel performance results due to inevitable memory limitations.
    - Operations which do too little processing 'work' for each invocation yield poor parallel processing results because data marshaling costs between the main processing thread and the multiple CPUs/GPUs will exceed performance gains. This issue is of significant design importance, because it yields the conclusion, backed by practical experimentation, that applying parallel processing technology to individual primitive operations (e.g. APL reduction, inner/outer product, etc.) may not be generally productive. Instead, to achieve significant performance improvements, groups of such 'primitive' operations need to be combined and performed in parallel without intervening data marshaling. This issue is the origin of the probable need to reconfigure the application system code to replace blocks of APL primitive functions and operations with an 'out of process' module which performs multiple 'primitive' operations in parallel with data marshaling between the main process and the CPUs/GPUs occurring only at the beginning and end of this module rather than between each 'primitive' operation. This issue is also the origin of the probable need to involve an APL2000 training and consulting effort because the creation of such a module to perform multiple 'primitive' operations in parallel entirely within the CPU/GPU will involve programming techniques that may not be familiar to APL+Win application system programmers.

APLNext is currently performing practical research as described above. It is anticipated that demonstrations of this technology will be given at selected venues possibly starting as early as April 2013. Subsequently the technology will be made accessible in test versions of the APL+Win product, potentially in combination with a seminar illustrating the use of the features.

APL+Win customers can take part in this research. APL+Win application system programmers are encouraged to investigate their in-production application systems to record performance, identify areas with significant processing times and report their findings in this APL2000 forum topic. The APL+Win monitor function (□MF) is useful for that purpose as is recording of function and operator frequency of use, function argument characteristics such as shape and rank and hardware and operating system details. Some programmers may want to compare performance in selected areas of their application system with and without multi-core parallel processing using the APLNext Supervisor and report that information in the Supervisor forum.

Information from customers that would be useful includes how many in-production application systems:

- Have performance which is bound by mathematical or logical calculations as compared to flow control, memory limits, or data input/output?
- Have repetitive processes without side effects that are suitable for parallel processing?
- Perform mathematical operations on arrays with relatively many elements?
- Are amenable to reconfiguration to take advantage of parallel processing technology?

More information on parallel processing: http://en.wikipedia.org/wiki/Parallel_computing

Contact sales@apl2000.com to learn how APL2000 consultants can analyze your application system and either suggest or implement modifications to improve its performance.