Effect of Complementary Processing on Navy Command and Control Software

Marion G. Ceruti, Ph.D.,
Senior Member, IEEE
Space and Naval Warfare Systems Center, D4121
53560 Hull Street
San Diego, CA 92152-5001, USA
+1 619 553 4068, ceruti@spawar.navy.mil

Ray C. Trout and Tse Lee
Complementary Systems Inc.
18100 Upper Bay Road, Suite 110
Houston, TX, 77058, USA
+1 281 335 9103, RayTrout@aol.com

Abstract
Complementary Processing, (CP), is a scheduling methodology to increase software execution speed. Experiments were performed to test CP with command and control software, specifically, to measure the effect of CP on the speed of processing Navy track-message data. A 37.5-fold increase was observed in this preliminary test. Recommendations for future tests are discussed.

1. Introduction
The Navy is investigating Complementary Processing, (CP), a scheduling methodology for increasing software execution speed [1], to evaluate its utility in improving the performance of command and control systems. The Global Command and Control System - Maritime (GCCS-M), includes software modules for tracking the ship positions. An experiment measured the speed of execution of the track-processing software both with and without CP.

2. Experimental methods
The experiment simulated the transmission of Link16 track messages from the Advanced Combat Direction System (ACDS) into the GCCS-M software using a network with TCP/IP. The Link16 transmitter was simulated by an IBM-compatible personal computer (PC) that was used to construct a data stream consisting of messages with information derived from actual GCCS-M track data. The PC fed this Link16 data stream into the JMCIS Hewlett Packard (HP) Model C110 computer running the HP-UX OS version 10.2.

These data were received in the GCCS-M HP machine and processed in the same manner as in an operational scenario. The GCCS-M software parsed and processed the incoming track data and stored the processed tracks in shared memory. An experimental measured the speed of execution of the track-processing software both with and without CP.

Data were collected to determine two time intervals for each track, \( t(B) \) and \( t(C) \). The baseline time interval for a track processed without CP is given by:

\[ t(B) = \text{time stamp upon data input} + \text{time stamp at output record} \]

\[ t(C) = \text{time stamp upon data input} + \text{time stamp at output record} \]
\[ dt(B(J)) = OUTPUT t(B(J)) - INPUT t(B(J)) \]  
(1)

Similarly, the test time interval for a track with CP is:
\[ dt(C(J)) = OUTPUT t(C(J)) - INPUT t(C(J)) \]  
(2)

where \( dt \) signifies the time difference and \( t(B(J)) \) is the notation for the time stamp of the data, e.g. INPUT \( t(B(J)) \) is the time the track was recorded in the test data generator and OUTPUT \( t(B(J)) \) is the time that the track was recorded as it was stored in shared memory with JMCIS and without CP software.

The CP software affixes a time stamp, called INPUT \( t(C(J)) \), to each track when it is stored in shared memory. CP also affixes a time stamp, OUTPUT \( t(C(J)) \), to each track when it is stored in shared memory.

The standard deviation, \( S \), of the sample was calculated according to the following formula:
\[ S = \left( \frac{\sum (x_i - <x>)^2}{n-1} \right)^{1/2} \]  
(3)

where \( x_i \) is the value of the \( i^{th} \) processing duration, \( <x> \) is the mean processing duration, and \( n \) is the sample size.

The CP software affixes a time stamp, called INPUT \( t(C(J)) \), to each track when it is stored in shared memory. CP also affixes a time stamp, OUTPUT \( t(C(J)) \), to each track when it is stored in shared memory.

The standard deviation, \( S \), of the sample was calculated according to the following formula:
\[ S = \left( \frac{\sum (x_i - <x>)^2}{n-1} \right)^{1/2} \]  
(3)

where \( x_i \) is the value of the \( i^{th} \) processing duration, \( <x> \) is the mean processing duration, and \( n \) is the sample size.

The PIF, the ratio of the mean processing durations without CP and with CP, is as follows:
\[ PIF = \frac{dt(B(J))}{dt(C(J))} \]  
(4)

where the brackets indicate the mean values from Table 1. The numerical value for the PIF was 37.5, which was obtained on the HP UX system, which does not support threads. CP is expected to increase processing speeds about 30% more than this in a thread-enabled environment.

4. Discussion and recommendations

An examination of the raw baseline data, INPUT \( B(J) \), and OUTPUT \( B(J) \), reveals dramatic evidence of the inefficiency of interrupt processing. The data for the first track were received at the HP's input port at \( t = 0 \) and were not fully processed and stored in shared memory until \( t = 660315 \) \( \mu \text{sec} \). During this time interval for processing the first track, the data for all of the other tracks were received at the HP input port. Therefore, while the CPU was processing computations associated with the first (and subsequent) track(s), it was interrupted multiple times. The last track data set was received at the input port prior to the time when GCCS-M completed processing the data for the first track and prior to storing the processed data of the first track in shared memory. The last track was not fully processed until all processing of prior tracks had completed, causing the last track to have the longest processing time.

The processing durations, \( dt(B(J)) \), exhibit a systematic increase with increasing value of \( i \). Thus, the mean value \( <dt(B(J))> \) is a function of the number of tracks, \( n \). This result is expected because the processing time increases with the number of interrupts, degrading the processing efficiency roughly in a linear manner:
\[ dt(B(J))i = m \cdot i + b \]  
(5)

where \( b \) is a constant, \( i \) is the number of the track, and \( m \) is the slope of the linear function. In this case, \( m \) was 21 \( \mu \text{sec} \), or \( -2 \mu \text{sec} \).

Equation (5) is valid in several segments over the range for which available shared memory can accommodate additional processed data sets (e.g. more tracks). Due to the limited shared memory, more than 95 baseline trials could not be processed successfully. Equation 5 is not valid in several trials in which the GCCS-M software appeared to take much longer to process the data than in most cases. Violations of (5) occurred to a lesser extent due to the periodicity fluctuations in data inputs generated by the PC.

The distributions of processing durations for both the baseline data and the CP test trials were observed to be highly asymmetric and skewed. For example, in the CP trials, the 72 values fell below the mean whereas 23 values were above the mean. This asymmetry also is expressed in the differences between the mean, the mode and the median for each set of durations, both \( dt(B(J)) \) and \( dt(C(J)) \). Whereas the result for baseline trials were consistent with theory, the values of the processing durations for CP trials exhibited much more variance than was expected based on the theoretical predictability of processing information frames and time periods. This is expressed in Table 1 by the large value of \( S \) equal to \( 25560.7 \mu \text{sec} \), which is almost 1.5 times larger than the mean processing duration for CP. \( S \) for baseline processing durations was considerably smaller. This discrepancy between theory and experimental observation requires further investigation.

The mean processing duration of the CP and the PIF are encouraging, but these results are preliminary. Additional tests are needed to explain the following observations:
- large standard deviation for CP processing duration,
- highly unusual distribution of CP processing durations with wide variance in mean, mode, and median values,
- discontinuities in linear function of baseline track processing durations vs. track number, and
- shared-memory limitations on the number of test trials.

For further directions regarding future research, see [1].

Reference