CONCURRENT CSOUND\footnote{Durham Music Technology is a collaboration between the School of Engineering & Applied Science and the Department of Music at the University of Durham}:
PARALLEL EXECUTION FOR HIGH SPEED DIRECT SYNTHESIS

Nick Bailey
Alan Purvis
Peter Manning
DURHAM MUSIC TECHNOLOGY* School of Engineering & Applied Science University of Durham South Road
Durham DH1 3LE ENGLAND

Ian Bowler
Department of Music
University of Durham Palace Green Durham DH1 4RL ENGLAND

Michael Clarke, Music Department, Sunderland Polytechnic, ENGLAND

ABSTRACT: This group has already addressed the problem of increasing the speed of MIT’s CSOUND direct synthesis program by parallel execution after suitable pre-processing of the score file. This paper describes the impact of an advanced parallel architecture and presents some benchmark results indicating the speed increase achieved with real examples. A live demonstration of the system illustrates the order of speed increase which is achieved.

1. Introduction

It has already been shown that the speed of execution of GOUND can be considerably enhanced by running the program on an INMOS Transputer\cite{Transputer}, simply by virtue of this processor’s proficiency in general purpose, floating point calculations\cite{FloPy}. Using only one T800 floating point, 20MHz Transputer, the speed of execution is some 24 times as great as a desktop P.C. based on a 68000. An approximate load-balancing algorithm for mapping standard scores onto multi-processor arrays has also been presented, and a suitable topology which avoids communications bottlenecks to a large extent for many digital signal processing algorithms is available and tested\cite{Efficiency}.

2. Performance Evaluation

The performance index of a parallel computing system is calculated by dividing the speed-up relative to a single processor system by the number of processors in the network. A good parallel system will have a performance index close to 1, so that doubling the number of processors very nearly doubles the speed of execution.

Other authors have addressed the problem of the most appropriate scheduling techniques, but most have concentrated on real-time systems which present slightly different problems. Holm uses a similar scheduling algorithm to that used by the SORT program, where events are placed of "Fundamental Clock Pulse Lists"\cite{Holm}. MIDI "operating systems" which support multi-tasking and pipelining from a common controller have also been written\cite{MIDI}. Of greater significance to the CSOUND scheduling algorithm are the results of W.F.Walker\cite{Walker} which suggest that the scheduler may saturate for large numbers of processors, especially in a real-time context.

\footnote{Durham Music Technology is a collaboration between the School of Engineering & Applied Science and the Department of Music at the University of Durham}
\footnote{CSOUND is modified with the kind permission of Barry Vercoe, Department of Music, MIT}
The analysis presented here assumes that the processing time is broken into three elements: useful computation, while the processor is performing calculations which are directly involved in the production of sound samples; necessary computation, while the processor is busy performing buffering or communication subroutines which do not in themselves contribute to synthesis; and suspension, an "all else fails" state during which the processor awaits host I/O with insufficient buffer space to continue calculation. By manipulating the program source code and input data, information can be gained as to the relative time spent in each of these states.

Two scores and orchestrations were used to benchmark the system. Programme 1* is an example with many short events in the score and an orchestra which is computationally inexpensive; programme 2** has fewer events in the score, but has an orchestra which demands a great deal of computation.

2.1 Execution Profiling

The first experiment attempts to evaluate how much time is spent in I/O overhead, and how much is spent in useful computation. Programme 1 was compiled by three processors in four different contexts:

a) Normal execution – sound output file produced on disk;
b) Output suppressed with the -n (no output file) option in the CGOOUND command line;
c) Normal execution with patched code in SNDBUF.C – all of the buffering and data transfer take place, but the final write(1) statement is not executed so that the data is discarded;
d) As (c), but the score is sorted for ten processors and all score lines referring to processors which were not fitted removed.

The execution times are shown in Table 1.

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>Time/Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Normal Execution</td>
<td>497</td>
</tr>
<tr>
<td>(b)</td>
<td>No Output from Program</td>
<td>332</td>
</tr>
<tr>
<td>(c)</td>
<td>No Output to Disk</td>
<td>377</td>
</tr>
<tr>
<td>(d)</td>
<td>Sparse Score</td>
<td>93</td>
</tr>
</tbody>
</table>

Table 1. Experiment 1 Execution Times.

By far the longest execution time is taken by test (a), indicating that, even for a system expanded to only three transputers, the delay incurred in writing to the host's file system is already significant. When no output is produced by the program, as in test (b), the execution completes in 67% of the time taken when writing the sound-file to the disk. However, when CGOOUND is invoked with the -n command line option, the sound sample buffering and recombination code will never be invoked. The wasted 33% will be spent not just in a suspended state, but also in necessary computation.

* Extract from Elgar's "Enigma Variations" performed on a simple pipe-organ model. Plays for 12 seconds at 25KHz sample rate.
** Extract from J.S. Bach's "Die Kunst der Fuge" performed on a complex POF model. Plays for 7 seconds at 25KHz sample rate.

ICMC GLASGOW 1990 PROCEEDINGS 47
An estimate of the time spent in necessary computation can be obtained from test (c). In this case, the main program is instructed to produce sound output, but the sound output buffering software is modified so that all of the buffering and communication overhead is retained, but the sound samples are discarded at the last moment. From this benchmark, we deduce that 48% is spent in useful computation and 12% in necessary computation. Put another way, of the total time spent in the compilation of this example: 24% is spent suspended pending mass-storage operations; 67% is spent in useful computation; only 9% is spent in necessary computation. This is an encouraging indication of the ability of the Transputer to perform communication operations between concurrent processes without great impact on the general processor throughput.

A simulation was devised to estimate the behaviour of a larger system. The score is sorted using the standard utility, but specifying a pipeline of ten processors instead of three. The sorted score is then passed through a filter program, which removes those lines marked to be routed to non-existent processors. This produces a score with notes allocated sparsely amongst the processors (test (d)), which is what one would expect in a large system. If there is sufficient buffering to allow each processor to "run ahead" freely, the execution should take approximately 0.4 times as long on the ten-processor simulation as it does running in its entirety on three processors (writing the sample data to disk is suppressed as for test (c) in order to avoid introducing mass-storage delays). In fact, Table 1 shows that the execution time is cut to 20% of the original; the error is introduced by a lucky allocation of "easy" (short or computationally inexpensive) notes to the first three processors in the pipeline, but is nevertheless an encouraging indication that a much larger system could still show significant speed-ups, provided that a sufficiently fast disk drive is available.

2.3 Buffer usage

Having produced an estimated execution profile of the parallel CGSOUND package, memory usage must also be considered. The prototype three-processor system has 1MB of memory at each node. Of this, 256KB is allocated to the sound sample recombination buffers. The quantity of memory available for this purpose is defined by the manifest constant SBSIZE in file BUFS.H. The system normally operates with 32 sound buffers, each 8KB in length. Programme 2 was chosen to assess the significance of buffer length, because it contains fairly sparsely distributed score events, and each event is computationally expensive. It is this combination which places the greatest stress on the buffering mechanism. Table 2 shows the execution times for programme 2 compiling with three processors and various buffer sizes.

<table>
<thead>
<tr>
<th>SBSIZE</th>
<th>RAM Allocated</th>
<th>Time/Min.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>256KB</td>
<td>11.98</td>
</tr>
<tr>
<td>8</td>
<td>64KB</td>
<td>11.50</td>
</tr>
<tr>
<td>16</td>
<td>128KB</td>
<td>10.35</td>
</tr>
<tr>
<td>32</td>
<td>256KB</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Table 2: Experiment 2 Execution Times.
There is a 27% increase in compilation time with 4 sound buffers over the time required with 2 buffers. This is a significant change, although it is quite favourable compared with the performance of some algorithms mapped in corporate lots into a parallel environment.

2.3 Performance Indices

As a final experiment, programme 2 is compiled (and the output written to disk) using one, two and three processors, and the performance index calculated. The single-processor version of the program runs with the greatest efficiency; this is as would be expected, as it has a far lower communications overhead than the pipeline versions. The addition of an extra processor (with the attendant communications software) reduces the performance index to 0.90, but the addition of a third processor has a negligible effect upon the performance index. It is this stability in the performance index with the addition of an extra processor which indicates a successful mapping into a limited parallel environment.

3. Conclusions

A cost-effective solution to the problem of providing computer musicians with direct synthesis machines of reasonable performance has been achieved, the most significant limiting factor now being the rate of writing to the host machine’s file system under program control. The software and hardware package presented here compiles CSOUND source at a speed in excess of 100 times that of a conventional 16-bit microprocessor.

References


Fig. 1: Transputer CSOUND Profile