ASPEN/EXPLORE: A multiprocessing assistant for electroacoustic music composers

Ambrose Field, Studio Director
Department of Music, University of York,
Heslington, York, YO10 5DD, UK

July 12, 1999

Abstract

ASPEN/EXPLORE is a productivity system for composers who need to conduct extensive digital exploration of a set of recorded sound materials. Electroacoustic composers often need to fine-tune the settings and parameters of their sound treatments. When computing power was limited, this process could involve much trial and error, and lengthy delays. However, with contemporary technology, musicians may now be spending longer tweaking settings and parameters than the computer requires to generate the results. This situation is clearly inefficient, and is set to get worse as low-cost desktop workstations become increasingly capable of 'faster than real-time' performance.

KEYWORDS: Electroacoustic composition, parallel computing, productivity system, scripting languages.
1 Purpose

This system aims to provide electroacoustic composers with some assistance in the more routine aspects of their work. Typically, an electroacoustic composer will follow a process whereby sounds are recorded or synthesised, manipulated by signal processing packages and mixed together. These stages do not necessarily have to be executed in this order.

ASPEN/EXPLORE is a system designed to help the sound-processing phase of the compositional process become more productive. As ASPEN/EXPLORE aims simply to give the composer more sounds to choose from, it is not a system for algorithmic composition per se. Recent trials using a parallel SGI Origin 2000 system show that productivity within the sound-design phase of the compositional process, can be increased by as much as ten times. This is compared with a composer executing the same tasks in an interactive, single processor environment.\(^1\)

ASPEN is an acronym for assisted sound processing environment. EXPLORE is a macro-package of ASPEN routines that enables extensive off-line exploration of sound treatments.

2 Implementation

The ASPEN/EXPLORE package has the following aims:

- To maximize the potential of parallel computer systems for performing routine compositional tasks. The user does not need to provide any special directives in order to enable the parallel execution of ASPEN code.

- To provide an extensible, unified off-line sound processing environment.

- To facilitate the integration of third-party software into highly customized work schemes.

Written in TCL with C extensions, ASPEN/EXPLORE currently runs on parallel computers under SGI’s IRIX operating system. The code is extensible, and contains the following components.

3 Construction

ASPEN/EXPLORE consists of a front-end where the compositional tasks are defined, a task decomposition system dedicated to obtaining optimum parallel efficiency and a back-end that currently runs a simple neural-network soundfile sorting program. The structure of ASPEN/EXPLORE is shown in figure one.

3.1 Front-end

The ASPEN/EXPLORE front-end consists of a scripting language, based on TCL. ASPEN is not a new ‘MUSIC-N’ type language or sound compiler. Instead, it assists users by enabling them to write compositionally useful scripts which use their own existing software tools. These scripts are run off-line as a batch job. Scripting systems which control third party tools are not a new idea [1] – operating system shells, or more sophisticated scripting languages such as TCL/TK, Perl, or JAVA can all perform this function. ASPEN/EXPLORE, however, is a scripting system dedicated to electroacoustic composition. With ASPEN/EXPLORE users may chain their different sound processing tools together, and the system will transparently arrange all file-handling. Also, it will schedule those tools for optimum execution in parallel.

In the following code example, the user has selected a file at random, added equalisation, band-pass filters and a vary-speed function taken from the commercial CDP sound processing kit [2]. Note how simple this script

\(^1\) The processor used in both cases was the MIPS R10000@195 MHZ.
is. File-handling is arranged by the ASPEN/EXPLORE sub-system. Optimum parallel execution of the resulting pipeline is also organised transparently.

# ASPEN/EXPLORE: Code example 1

# Set input/output files

workwith [pick *.aiff]
outputdir /usr/people/mydir

# Run CDP commands.
# Note that functions can be given as command arguments

eq H 1200 +12
lp6 [rand 120 300] -96
accel 0.5 [sound_len]

# Collect the output
collect newfile.aiff

3.2 Task decomposition

It is the job of the ASPEN/EXPLORE task decomposition module to allocate system resources so that parallel execution is possible. The task decomposition module contains three code units: a directory engine, a scheduler and a system-wide load balancer.

The directory engine looks for high-level parallelism in the job described by the ASPEN script. It does this by searching for task descriptions that do not create any data dependencies. If the results of one process do not need to access data from a previous task then that process is well suited for parallel execution.² Task decomposition is accomplished on a variety of different levels. Even within code that appears unsuitable for parallel execution, parallelism might still exist. For instance, in the example above, the ASPEN code describes a single, linear chain of work. Here, the output of one process must feed into the next. The user supplied a 4-channel file as input.

The directory engine will discover the following parallelism in this apparently sequential code. Firstly, and most obviously, each channel of the soundfile does not depend on the others within the processes described. The process chain is thus split by the system into 4 threads. Furthermore, I/O can be optimised as the chain of processing does not require random access to the original soundfile. Detecting this, ASPEN will assign the eq, lp6h, accel commands to separate sub-threads and create memory buffers optimised for this purpose. With the simple EXPLORE directives introduced below, it is possible to process whole directories of data with simple commands. Here the directory engine has less to do, as the opportunities for parallel execution are many in this situation.

Management of the threads created by the directory engine is done by the scheduler and load balancing modules. These modules interact with the underlying UNIX system in order to obtain efficient processing. The scheduler monitors load distribution across the processing array, system wait states and idle time. From this information, the scheduler can assess the best way of keeping the computer occupied. Under normal operation, it has been observed that the scheduler creates about twice as many threads as there are physical processors. The system thus responds well to I/O intensive soundfile processing, as a CPU can often compute results in one application whilst data is being fetched for another.

3.3 Back-end

When processing is complete, the ASPEN back-end is activated. This consists of a simple neural-network that can discriminate between basic sound morphologies. Via a phase vocoder based parameter extraction program, ASPEN can recognise simple motion trajectories and

²See [3] for details of high performance computing techniques.
classify sounds with pitch or noise-based spectra. Soundfiles can then be moved automatically into appropriate directories, to assist the composer in auditioning the results.

The ASPEN/EXPLORE subsystem also keeps a log of all operations undertaken. In a future release of the system, this history mechanism could be integrated into the neural-net in order to learn the working practice of the composer.

4 Third-party packages

The package manager provides an interface between the user's command scripts and third-party software packages. Essentially, the package manager generates information defining how third-party commands are to be invoked by the underlying UNIX system. Furthermore, third-party tools do not have to be modified to run within ASPEN/EXPLORE. New commands can be added through a simple adapter script which analyses the data output from a typical run of a new command, and works out the parameters necessary to run that command. The output from the adapter script is stored as a new package manager module. The user is able to edit this module to gain extra functionality.

5 EXPLORE

EXPLORE is a collection of macros which help composers to process large amounts of data, with very few commands. Currently, EXPLORE macros cover the following areas of sound processing: convolution, phase vocoder time stretching, repetitive mixing tasks, cross-synthesis, filtering, soundfile clean-up, texture layering, and event generation. EXPLORE also contains functions which enable investigative sound processing. For example, it is possible to sweep the parameters of any process over a range of values whilst keeping the same set of input files.

6 Future work

ASPEN/EXPLORE has already provided many of the sound transformations in the author's electroacoustic work Expanss Hotel. The first public version of the program will contain support for the LINUX operating system, and an interface to permit multiprocessing over a network of workstations. The code also requires optimization to prevent serial operations from impacting on currently running parallel processes.

7 References