

## Executive Summary

This study on the portability of applications developed in Gedae analyzes the work required to move an example application from simulation on a PC to running on a DSP board (the Mercury AdapDev™ system), and then to running on a multicore processor (the Cell Broadband Engine™ (Cell/B.E.)). We illustrate how the architecture considerations were taken into account when porting the application to each system and quantify both the work required to port the application and the performance of the application on each system.

## The Application

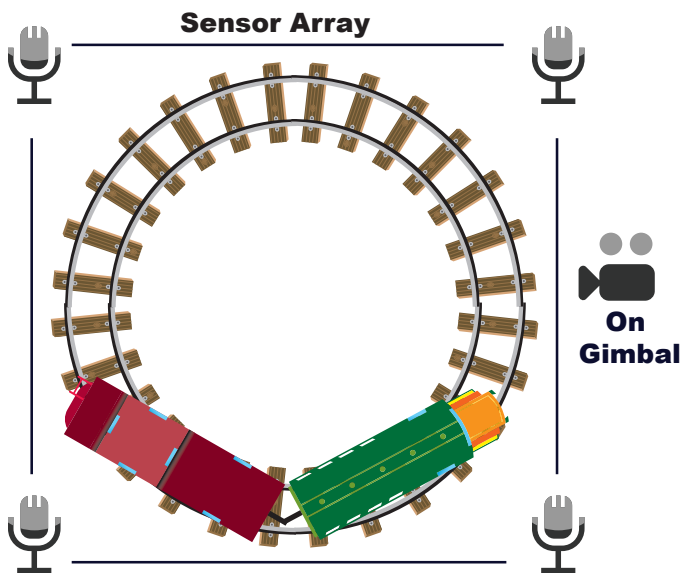


Figure 1: Tracking algorithm targeting train

The example application involves tracking a model train as it goes in a circular path around its track. The application uses input audio data from four microphones placed in a circle around the track to locate the train in the audio field. Using this location, the application pans and tilts a camera to point at the engine of the train. An illustration of this environment is shown in Figure 1.

The algorithm is based on RADAR technology. A beamformer correlates a linear array of RADAR sensors to identify a target based on a beam of high correlation. In this application, the array is circular,

so the high intensity in the correlation of the four channels forms a spot, as shown in Figure 2. After the spot formation forms this audio map, a detection algorithm identifies the high intensity peak corresponding to the train, and pan and tilt angles are computed to reposition the camera.

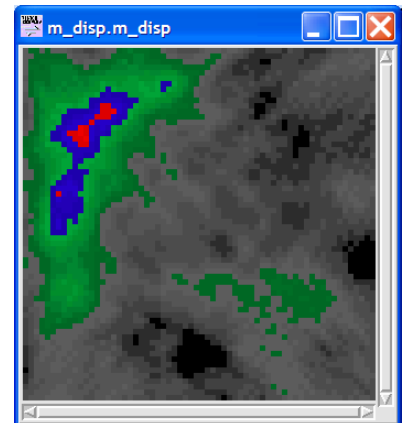


Figure 2: Spot formation

Because this application must continue to work in noisy environments, several approaches are used to reduce jitter and ensure smooth tracking. The input channels are run through low pass filtering to remove frequencies outside the desired band. In the detection algorithm, several peaks are identified and tested, and feedback is used to monitor the speed and direction of the train to help rule out spurious data.

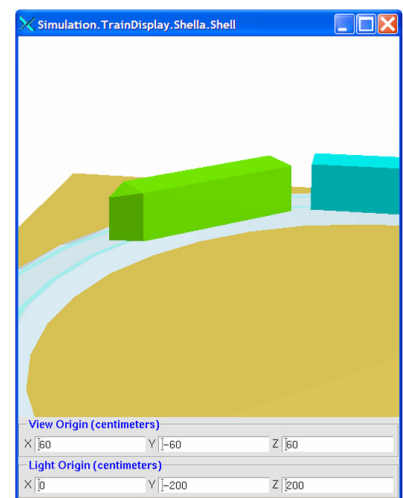


Figure 3: Sim rendering

## Simulation

The first testing ground for this application is in simulation. The environment of the train and camera are simulated, and the four channels of audio data for the microphone array are read from files. To show the results of the simulation, a 3-D rendering of the scene is presented from the view angle of the camera, as shown in Figure 3.

Using Gedae-Simulation, experiments are done on multiprocessor implementations of the application to prepare for moving it to hardware processing real-time data.

Once created, the code of a Gedae application does

not have to be changed in order to partition and map it to multiple processors. During simulation, several mappings to virtual processors are used in different configurations, and the results are analyzed in the Gedae Trace Table.

## Multiprocessor DSP Implementation

To transition this application to using real world data, we port it to the Mercury AdapDev system. The Mercury AdapDev system provides an Intel Pentium host and two quad DSP boards where each DSP is a 500MHz AltiVec processor. Physical components for the camera, gimbal, microphones, and audio digital converter (ADC) are assembled. The application is altered to remove the artificial audio source and scene

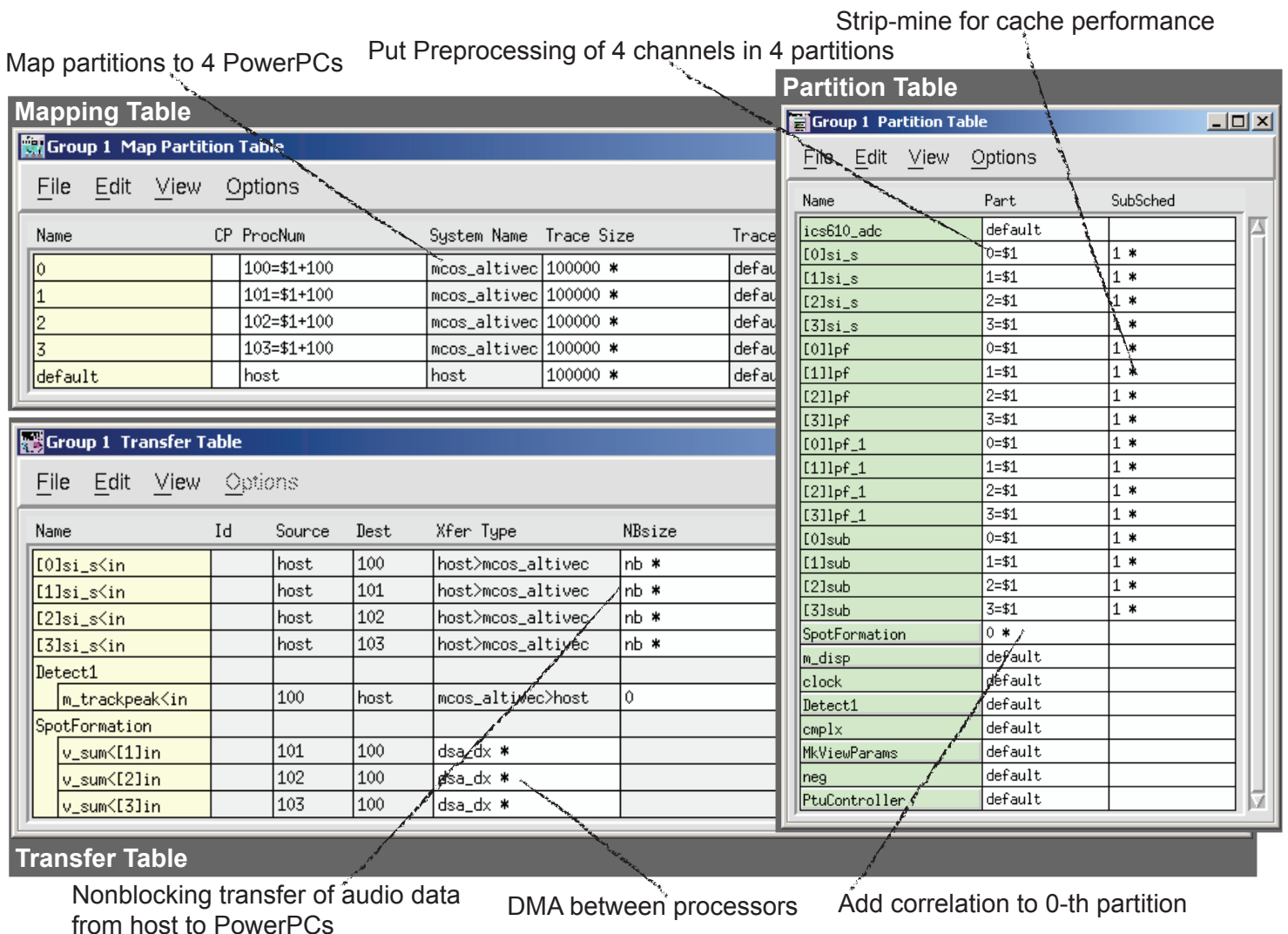


Figure 4: Mapping application to DSPs without changes to the code

rendering and replace it with an interface to the ADC (via PCI), the gimbal (via serial port), and the camera (via USB).

While the sources and sinks are replaced to use real world data, the algorithms and their coding did not need to be changed to create a real-time implementation. Using our experiments from the simulation, a partitioning and mapping scheme is entered into the Partition and Map Partition Tables shown in Figure 4. The communication protocols are tweaking using the Transfer Table, picking direct schedule access transfers (equivalent to DMA) and removing the blocking of the host-to-DSP transfers. Additionally, automated strip-mining is used to optimize vectorization and improve cache utilization. These changes, including both changing the graph to use real world data and setting the implementation parameters, took one day of effort and the resulting implementation is able to process three frames per second – sufficient to track the train at its maximum speed with a low number of errors or jitter.

### Multicore Implementation

To illustrate support for the Cell/B.E., this application is ported to the Sony Playstation 3. The Cell/B.E.

on the Sony Playstation 3 provides a dual threaded Power Processing Element (PPE) core as well as six Synergistic Processing Elements (SPE). The SPEs are very efficient vector processors but have very tight memory restrictions, with only 256KB of local storage and no cache. Programming the Cell/B.E. by hand requires careful management and planning of memory and data movement between the SPEs.

Gedae addresses the issues of memory management and data movement directly. The automated implementation of these issues simplifies development for the Cell/BE. After altering the application to use a USB-based ADC (the Playstation 3 does not have a PCI slot), the application was easily moved to the Cell/BE, and the process of optimizing it for the multicore architecture took two hours.

To optimize the application for the Cell/BE, the

Put preprocessing of 4 channels in 4 partitions

Map partitions to 6 SPEs      Strip-mine to reduce memory footprint

**Mapping Table**

Name	CP ProcNum	System Name	Trace Size
0	100=100+\$1	spu	1000 *
1	101=100+\$1	spu	1000 *
2	102=100+\$1	spu	1000 *
3	103=100+\$1	spu	1000 *
4	104=100+\$1	spu	1000 *
5	105=100+\$1	spu	1000 *
default	host	elinuxppc	10000

**Partition Table**

Name	Part	SubSched
quattro	default	
[0]Preprocess	0=\$1	1
[1]Preprocess	1=\$1	1
[2]Preprocess	2=\$1	1
[3]Preprocess	3=\$1	1
SpotFormation		
[0]s_ovrl_v	0=\$1	1 *
[1]s_ovrl_v	1=\$1	1 *
[2]s_ovrl_v	2=\$1	1 *
[3]s_ovrl_v	3=\$1	1 *
[0]v_selWinterpolateV_v	0=\$1	1 *
[1]v_selWinterpolateV_v	1=\$1	1 *
[2]v_selWinterpolateV_v	2=\$1	1 *
[3]v_selWinterpolateV_v	3=\$1	1 *
v_sum	4 *	1 *
v_sqr	4 *	1 *
v_integrate	5 *	1 *
v_m1	default	

Use 2 SPEs to perform 1st stage of correlation

Figure 5: Mapping application to SPEs without changes to the code

compute-intensive signal processing portion of the application is partitioned for the six SPEs. The memory footprint of the program and data is taken into account during this process, using the Schedule Parameters dialog to analyze the size of the threads that will be created for each partition. To reduce the size of the memory footprint, the automated strip-mining capability is used, allowing a set of audio vectors to be processed independently on each SPE instead of en masse. Additionally, a primitive that performs a column-wise sum of a matrix is identified as pushing the thread memory size over the limit. To fix the issue, the primitive is replaced with one that integrates a series of row vectors.

The Gedae Trace Table is used to analyze the performance when running on the Cell/B.E. During this process, one primitive is identified as being slow and is recoded to use a unity stride. Based on the processor load, the distribution of the work is also altered. After two hours, in the final optimization, four SPEs are used to do a majority of the preprocessing (one SPE per audio channel), including the band filtering of the frequency spectrum. The other two SPEs are used to combine the data in the correlation calculation of the spot formation. The PPE performs the detection algorithm and interfaces with the I/O devices. With this implementation, the application is able to process almost 15 frames per second on the Cell/BE, providing a much smoother tracking of the train.

## Analysis of Effort and Results

Coding for multiprocess and multicore systems using traditional methods depends on languages and compilers that are based on serial processors. Because the code is structured in this way, the software architecture - or how the software is broken up across processors - becomes buried in the code. Issues like the number of processors, their interconnection

and bandwidth, and memory size and structure get intertwined with the algorithms in the code. This entanglement makes it extremely difficult to port an application to a new system.

Gedae mitigates the risk of porting software by automating the incorporation of the software architecture. As illustrated by this tracking demonstration and shown in the table below, the tasks of moving from simulation to multiple DSPs and then on again to the Cell/B.E. are straightforward, taking a man-day or less to accomplish while still harnessing the power of the new hardware and achieving higher performance.

Target	Programmer Hours	Performance
Simulation	4 Weeks	-
MercuryAdapDev	6 Hours	3 Hz
Cell/B.E.	2 Hours	15 Hz

Hz: Millions of Iterations per Second

## Get Started Today

Gedae has developed a quick start package that includes a PS3 pre-loaded not only with the Gedae software but with the IBM SDK and Fedora Core Linux OS. The package includes an accelerated training program and sample programs.

Contact us today to start realizing the benefits of this powerful combination.

© 2007 Gedae Inc. All rights reserved. Gedae is a trademark of Gedae Inc. in the United States and/or other jurisdictions. All other markets and names mentioned herein may be trademarks of their respective companies.