Reference Design Guide

1200-Port Media Server

Optimizing the Performance of High Density Systems Built Using Intel® NetStructure™ DM/V and DM/V-A Series Boards on the Windows* 2000 Operating System

Contents

Executive Summary

In today's economy, delivering high-density, scalable, and reliable communication solutions to market quickly is a necessity for original equipment manufacturers (OEMs), integrators, and ISVs so they can keep the competitive edge they need for continued success.

With the demand for enhanced services growing, large enterprises and service providers are continually breaking system density barriers. In response to rising customer needs for low cost, high-density media servers, and to answer critical density and performance questions, Intel has tested a 1200-port media server solution while optimizing system software for ultra high-density systems. As a result customers can now confidently build a wide variety of sophisticated high-density messaging, call center, conferencing and switching applications.

This *Reference Design Guid*e will describe the high-density media server testing environment used and give performance results as well as discuss the building blocks used to create a high-density system. It will explain the performance guidelines for creating such high-density systems and how the performance was achieved using the Intel® NetStructure™ DM/V and DM/V-A Series of boards, the Windows* 2000 operating system with Service Pack 2, and software and applications created for telco-grade, mission critical media servers.

I. Introduction

The market conditions of the post-dot.com burst and cut throat competition on basic services has left service providers with reduced revenues, network over-capacity, and significant capital constraints. They are looking for new sources of revenue while keeping their costs low. Meanwhile, medium and large enterprises are looking to improve customer satisfaction, retention, and increase employee efficiency while reducing operational and infrastructure costs.

As a result, both enterprise and service providers are looking for innovative enhanced services using open, standards-based building blocks. In addition, both customer types want solutions that are flexible, quickly integrateable, offer a lower cost of ownership, and yet still provide a strong potential return on investment (ROI). To satisfy these criteria and still reach the density levels required to support the growing needs of businesses, communications networks and solutions must ensure scalability without adversely affecting or degrading system performance.

Media servers are the platform to deploy these enhanced services. In addition, with the emphasis on controlling costs, the focus today is on high-density, scalable media servers with smaller footprints.

Intel provides open, standards-based building blocks to develop and deploy cost-effective high-density media servers. The Intel® NetStructure™ DM/V Voice and DM/V-A Multifunction Series support voice processing, speech recognition, and conferencing, and occupy a single PCI or CompactPCI* slot. Multiple boards from these series can be installed in a single computer, reducing the amount of hardware space needed for scalable, high-density solutions.

To reduce the customer's time-to-market, Intel has developed and tested a 1200-port media server system using the DM/V Voice and DM/V-A Multifunction Series boards, Windows 2000 with Service Pack 2, and Intel® Dialogic® System Release 5.1.1 for Windows. Through vigorous testing and analysis focused on scalability and performance in ultra high-density solutions, Intel engineers successfully established the optimal board parameters that let customers confidently push the density barriers in a single system to new heights. This *1200-Port Media Server Reference Design Guide* is the documentation of their process and the subsequent performance guidelines.

Applications for a High-Density Media Server

- Unified messaging — handles voice, fax, and regular text messages as objects in a single mailbox that a user can access either via email or telephone. If a user's PC has multimedia capabilities, the user can open and play voice messages. Fax images can be saved or printed. A user can access the same mailbox by telephone, and the email text will be converted to audio files and played back.
- Network call center/contact center — a central place where customer and other telephone calls are handled by an organization, typically with some computer automation. Call centers are deployed by any organization that uses the telephone for generating sales or providing service.
- Interactive voice response (IVR) — a software application that accepts a combination of voice telephone inputs and touch-tone keypad selections and provides appropriate responses in the form of voice, fax, callback, email, and other media. IVR is usually part of a larger application that includes database access.
- Voice portal — a Web site or other service that a user can reach by telephone for information such as weather, sports scores, or stock quotes. After requesting information by speaking or pressing keys, the voice portal responds with voice information using text-to-speech (TTS) or possibly with an email message.
- Conferencing — lets several parties be added to a phone conversation.
- Prepaid/debit card — a customer purchases in advance a certain amount of calling time or a dollar amount of credit towards making phone calls, and the value of the card decreases as calls are made. The accounting is typically done in a remote switch, which the user dials to make calls.
- Voice mail — lets a user receive, edit, and forward messages to one or more voice mailboxes.
- International callback — a system for avoiding regular phone company long-distance charges by having a call initiated from within the United States with the originating caller joining in as part of a conference call.
- Gateway switch — A gateway is a network point that acts as an entrance to another network. A gateway is often associated with both a router, which knows where to direct a given packet of data that arrives at the gateway, and a switch, which provides the actual path in and out of the gateway for a given packet.

II. The High-Density (1200-Port) Media Server

Scalability and performance are the key customer problems Intel focused on for building, testing, and optimizing a 1200-port media server reference system. How could Intel add more ports without affecting the high-performance its customers expect and demand? Systems must be scaleable for the customer who wants to expand from one T-1 or E-1 line to forty in a single box, without any performance degradation.

The latest performance results indicate that a system with ten Intel® NetStructure™ DM/V Voice Series boards (1200 channels), performing simultaneous play/record functions, has a CPU loading of about 20% without any performance degradation as density goes up.[1] CPU utilization has a linear correlation to the number of channels in service, demonstrating that the architecture has excellent scalability.

III. Methodology

In configuring a 1200-port media server architecture, Intel engineers selected the Intel® building blocks and third party components that could offer higher than normal performance and lower CPU utilization rates. The server was configured with ten Intel NetStructure DM/V Voice Series boards (DMV1200-4E1-PCI / DM/V1200-4E1-CPCI) for network connectivity and voice processing, which support up to four digital network interfaces and up to 120 ports of voice processing capability. Similar tests were performed with ten DM/V-A Multifunction Series boards (DM/V1200A-4E1-PCI), which support up to four digital network interfaces with enhanced media capabilities per board, and include support for continuous speech processing. For a detailed features list, read the datasheet online at http://www.intel.com/network/csp/products.

The boards were loaded into a chassis built with an Intel® Pentium® III processor, 1000 MHz CPU, and running on the Windows 2000 operating system with Service Pack 2. Using Windows 2000 and the Intel Dialogic System Release 5.1.1 for the Windows operating system gave developers the high availability and high performance features required for carrier-grade telco, service provider, and large enterprise solutions.

The Windows 2000's efficient threading model was utilized in the applications to fine-tune the system level performance. Testing results indicated acceptable system level scalability and stability. The CPU load on this ultra-high density system was low, leaving plenty room for customers to build their applications and achieve solid performance results.

The media server was tested for scalability and performance. Intel engineers developed a basic messaging application and tested it to a load of 1200 channels, each channel performing continuous and simultaneous play and record. System software, including libraries and firmware, were optimized; each was pushed to its limits to determine system flexibility and what resources would be strained at certain density levels. Once identified, optimizations were made and this process continued until the appropriate density levels were reached with no performance degradation.

Various programming models were tested as well. Synchronous versus asynchronous, single thread versus multi-thread ó the engineers tested various scenarios until they determined the combinations for optimal performance. Based on 1200 channels, Intel engineers experimented with the number of channels per thread (i.e., 120-ports per thread verses 240) Testing determined that, in a 1200-port configuration, 120-ports per

thread was the most efficient if developers used the multiple-threaded asynchronous programming model with sr_waitEvtEX() to handle events. The single-threaded asynchronous model, described in detail later in this document, is recommended for maximum CPU utilization rates; however, the difference between the 120-ports per threaded model (the multiple-threaded asynchronous model) and the single-threaded asynchronous model is minimal.

The asynchronous multiple-threaded programming model is the recommended model as it showed a lower CPU utilization rate than the extended asynchronous model; and the system demonstrated good scalability with a linear correlation between the CPU utilization rate and number of ports.

The methodology used to create a 1200-port media server is documented for programmers to take into account when writing their applications.

- Play/Record Termination
- Programming Model
- Using Concatenated/Index Play Instead of Issuing Multiple dx_play() Commands
- Calling dx_stopch( )
- Optimizing Performance of Devices
- Device Initialization

These guidelines are covered in greater detail in the **Performance Guidelines for High Density Systems** section of this guide.

High Density Media Server Testing Environment

- Hardware: Intel Pentium III processor, 1000 MHz processor CPU
- Operating System: Windoes 2000 with Service Pack 2
- Development Software: Intel Dialogic SR 5.1.1 for Windows
- Chassis: Transduction Berta 1000
- RAM: 512 MB SDRAM PC133
- Hard Drive: Seagate ST330620A-30GB IDE (7200 rpm, 8.5ms Seek Time)
- Communications/telephony boards: Up to 10 Intel NetStructure DM/V or DM/V-A Series boards: DM/V1200A-4E1-PCI or DM/V1200-4E1-CPCI. Although the results described in this document come from a system with E-1 boards, similar programming guidelines and results

IV. Development Environment

To create a media server that exceeds the density barriers of prior systems, Intel engineers selected the Intel and third-party products that are geared toward the development of high availability, high performance, carrier-grade media servers.

Windows 2000 Operating System with Service Pack 2

The 1200-port media server was built on the Windows 2000 operating system with Service Pack 2 using the Intel Dialogic SR 5.1.1 for Windows, which supports increasingly high-density, high-performance communications systems. The modular architecture supports rapid development cycles and boasts a well-defined, standard set of interfaces designed to ensure hardware compatibility. Not only has the density been dramatically increased, but also better system performance has been achieved. The high density and performance of these systems makes possible the development of a wide variety of sophisticated messaging, call center, conferencing and switching applications.

Intel® NetStructure™ Voice and Multifunction Series Boards

The DM/V and DM/V-A board series are based on the powerful DM3 architecture — a flexible, modular, and open architecture platform for developing leading-edge call processing applications and supporting the high-density, high performance needs of media servers. Developers can develop systems with densities ranging from 48 to 1200 ports of T-1/E-1 network connectivity, plus on-board voice, speech, and conferencing media capabilities æ all in a single chassis.

The DM/V-A board series features innovative continuous speech processing technology, a DSP-based signal processing solution optimized for speech recognition that enables a friendly user interface and seamless integration of speech recognition software from the

DM/V Voice Series

- Supports up to 120 ports of voice processing and four digital network interfaces in a single slot and scales up to 1200 ports per system
- Supports patented perfect call outbound call progress analysis that accurately discriminates human speech from recorded human voice and network noise
- Downloadable signal and call processing firmware provides the flexibility to enhance applications as needs change
- Deploys on either industry-standard PCI or CompactPCI form factor
- Allows a choice of T-1 or E-1 digital network interfaces with internationally approved CAS and ISDN Primary Rate
- Unified call control access through the Global Call interface provides worldwide application portability and helps shorten development time

DM/V-A Multifunction Board Series

Contains all the features and functionality of the DM/V board series, plus:

- Supports continuous speech processing
- Provides an onboard high-density conferencing solution that can be used to deploy network-grade conferencing systems
- Supports G.726, GSM, and TrueSpeech* voice coders

leading speech technology vendors. On-board conferencing offers one of the industry's most advanced feature sets, providing a pleasant conferencing experience for the end-user. An optimized state-of-the-art algorithm prevents noise build-up and echo in the conference. It also equalizes participant voice volumes, and offers optional DTMF clamping to limit audible enter and exit tones. On-board conferencing enables Intel customers to deploy network-grade conferencing systems with comparable features, audio quality and density as typical proprietary solutions at significantly reduced costs.

Software

System Release 5.1.1 for Windows unlocks rich multimedia resources (voice, fax, speech, and conferencing) and a full complement of digital network interfaces on the Windows 2000 operating system for the high-density Intel DM3-based board products. In addition, SR 5.1.1 for Windows enables solutions using the CompactPCI* form factor, supporting peripheral hot swap (PHS) SNMP, on-demand diagnostics, single board start/stop operation, firmware tracing, faster system download and initialization, and more.

The remainder of this *Reference Design Guide* details performance guidelines, testing environment, and performance results for the 1200-port media server. Readers who want additional information about creating a 1200-port media server are encouraged to contact an Intel technical sales representative or an account manager.

V. Performance Guidelines for High Density Systems

When programming applications for high-density systems containing DM/V Voice and DM/V-A Multifunction Series boards, Intel suggests several guidelines to maximize the performance of the system. These guidelines are discussed in the following sections:

- Play/Record Termination
- Programming Model
- Using Concatenated/Index Play Instead of Issuing Multiple dx_play() Commands
- Calling dx_stopch( )
- Optimizing Performance of Devices
- Device Initialization

Play/Record Termination

Terminating Events

Use digit mask termination **(DX_DIGMASK)** in the Termination Parameter Table data structure **(DV_TPT)**, to terminate events instead of getting digits via the Call Status Transition data structure **(DX_CST)** and stopping I/O functions on the channel via a **dx_stopch( )** call. The **DV_TPT** structure specifies the termination conditions.[1]

Maximum Function Time

Use maximum function time **(DX_MAXTIME)** in the Termination Parameter Table data structure **(DV_TPT)** instead of specifying the number of bytes allocated for recording using io_length in the I/O Transfer Table data structure **(DX_IOTT)**. The DV_TPT structure specifies termination conditions for recording.[2]

Maximum function time is equivalent to play/record duration.

Programming Model

To maximize performance, use the asynchronous single-threaded programming model using the **sr_waitevt()** function.[3] The testing data listed below was collected with a testing application that uses the Standard Template Library's mapper class, which has a better matching algorithm than linear search. Similarly, developers using this model must devise a better event- and handle-matching algorithm.

The following topics provide more information about the programming model.

Using the Asynchronous Single-Threaded Model

The asynchronous single-threaded model is the recommended programming model for this type of system. The application is then responsible for retrieving and handling the events in a suitable manner. For example, applications that interact with a database may need to dispatch events to another thread so that lengthy operations do not block the event polling thread. This approach allows you to implement the event-handling scheme to achieve optimal performance.

Number of Threads

The best combination of threads and channels used depends on the application. As stated above, we recommend a single thread for the handling of DM3 events. Fine-tuning the programming model to achieve the best performance may be necessary. For example, separate threads dedicated to system I/O may help system performance. On the other hand, if the application uses the record/play to/from memory feature, adding such a dedicated thread may not improve overall system performance.

In general, for Windows 2000 systems, our results show that fewer threads yield better results if there is no extensive disk activity. However, this is only a generic recommendation. Again, you should always fine-tune your applications according to their specific characteristics to achieve the best results.

Number of Processes

There are two considerations:

1. **Determine the correct number of processes for your system based on your service availability requirements**. You should do this because separate processes run independently. If one process encounters an error condition or is out of service, the rest will still be available. However, if the same process uses all resources and this process goes out of service, the whole system will be unavailable.
2. **More processes will add overhead to the system and thus reduce overall system performance.** In general, you should try to avoid large numbers of processes in one system. The overhead for the operating system to handle multiple processes can affect system performance.

Therefore, you need to maintain a balance based on service availability and overall system performance. Try different combinations to see what works best.

NOTE: The same device cannot be accessed via multiple processes.

Minimize Disk I/O

To maximize performance in high-density systems, minimize disk I/O operations (for example, play and record from memory). Intel provides an API that can play from and record to memory instead of disk file[4]. Intel recommends that the application manage its disk I/O in a way to minimize disk I/O activity during operation to achieve better system performance.

Using Concatenated/Index Play Instead of Issuing Multiple dx_play() Commands

Instead of using consecutive play commands, you can use the **DX_IOTT** input transfer table to describe a single data transfer from memory blocks or a custom device. **DX_IOTT** is a link list that can be used to indicate which file or segment of a file (using **lseek()** to set the offset) is to be played in a sequence. This will reduce the number of **dx_play()** commands issued. This is also known as index play or concatenated play. A similar principle applies to record.[5]

Calling dx_stopch( )

For the best performance, **dx_stopch( )** should be used in asynchronous mode rather than synchronous mode. Issuing this function call in asynchronous mode will not block and wait for termination of the function, and will allow the processing of other devices and events while the specified channel terminates. For applications to achieve the best possible performance and throughput, asynchronous programming techniques should be used when the option is provided.[6]

Optimizing Performance of Devices

If you open and close devices or your application uses Global Call to dynamically open and close devices as needed, application performance may be affected. To optimize performance, we recommend that you open all DM3 devices during application initialization and keep them open for the duration of the application. All devices should be in an idle state before they are closed, and all devices should be closed at the end of the application.

Device Initialization

The following recommendation from the *Compatibility Guide for the Intel Dialogic R4 API on DM3 Products for Windows* should be used as a guideline for device initialization. Please refer to the Compatibility Guide for details.

The **xx_open( )** functions for the Voice (dx), Global Call (gc), Network (dt), and Fax (fx) APIs are asynchronous in this release of R4 on DM3, unlike the standard R4 versions, which are synchronous. This should usually have no impact on an application, except in cases where a subsequent function calls on a device that is still initializing, that is, is in the process of opening. In such cases, the initialization must be finished before the follow-up function can work. The function will not return an error, but it is blocked until the device is initialized. For instance, if your application called the following two functions:

dx_open()

dx_getfeaturelist()

The **dx_getfeaturelist( )** is blocked until the initialization of the device is completed

internally, even though **dx_open( )** has already returned success. In other words, the initialization **dx_open( )** may appear to be complete, but, in truth, it is still going on in parallel. With some applications, this may cause slow device-initialization performance.

Fortunately, you can avoid this particular problem quite simply by reorganizing the way the application opens and then configures devices. The recommendation is to do all **xx_open( )** functions for all channels before proceeding with the next function. For example, you would have one loop through the system devices to do all the **xx_open( )** functions first, and then start a second loop through the devices to configure them, instead of doing one single loop where a **xx_open( )** is immediately followed by other API functions on the same device.

With this method, by the time all **xx_open( )** commands are completed, the first channel will be initialized, so you won't experience problems. This change is not necessary for all applications, but if you experience poor initialization performance, you can gain back speed by using this hint.

VI. High-Density Media Server Testing Environment

This section describes how Intel built the media server used for the performance tests.

Hardware and Software

This section describes what was used in testing the high-density media server. The specifications are given for the PCI form factor.

PCI Form Factor

Hardware: Intel Pentium III processor, 1000 MHz processor CPU

Operating System: Windows 2000

Development Software: Intel Dialogic SR 5.1.1 for Windows

Chassis: Transduction Berta 1000

RAM: 512 MB SDRAM PC133

Hard Drive: Seagate ST330620A-30GB IDE (7200 rpm, 8.5ms Seek Time)

Boards: Up to ten Intel NetStructure DM/V Voice and DM/V-A Multifunction Series boards: DM/V1200A-4E1-PCI or DM/V1200-4E1-CPCI. Although the results described in this document come from a system with E-1 boards, similar programming guidelines and results should also apply to T-1 boards.

VII. Performance Results

This section describes the performance characterization application that was tested and gives performance measurements. The performance characterization application was developed following the guidelines listed in *Section VI. High Density Media Server Testing Environment.* The application covers

- call control
- play and record

Call Control

The call control application was designed to achieve the highest possible call control completion rate. The call control completion rate was defined as the number of calls that were completed within an hour for one simple E-1 board without inter-call delay and intra-call delay.

Design Considerations

The application used here was designed with the following considerations:

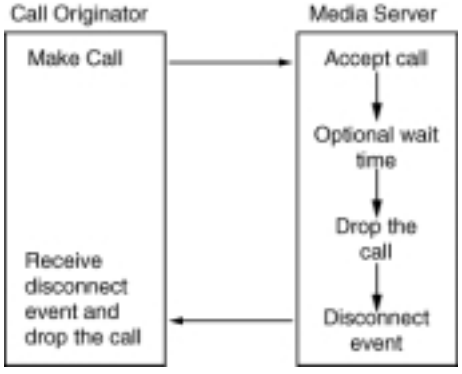
- Minimized disk I/O: all plays and records are done through memory
- The application was based on R4 APIs
- The application was written using the extended asynchronous mode
- No call setup precedes play or record

Call Control Sequence

In this call control application, the system does the following:

1. Both sides open channels
2. The incoming side waits for the call
3. The outbound side makes the call
4. The call is connected
5. The inbound side detects the connection and subsequently drops the call
6. The outbound side receives the disconnect event and drops the call

The following picture shows the sequence of the system's response to an incoming call.



1200-Channel Call Control Results

The call control completion rate defined above was measured approximately: 1.1 million calls per hour for ten E-1 boards (120 channels). In other words, it takes approximately 4 seconds to finish one call for each channel.

This number was obtained using five boards on the inbound side and five boards on the outbound site, connected by crossover cable, on the same ten-board system.

Play and Record

Play and record are the basic functions of any media server application. High performance on a high-density system is critical for a successful media server application. The sample application stressed the system by doing simultaneous play or record on all channels in the testing system.

CPU utilization rate and scalability of the system were measured. Two different event-processing models have been used. The first is the asynchronous single-thread model with **sr_waitEvt()** and the second is the extended asynchronous model with **sr_waitEvtEx()**. The results show a significant difference between the two – especially for high-density systems.

The current implementation of **sr_waitEvtEx()** does a linear search to match events with each device handle array supplied with each call to **sr_waitevtEx()**. With **sr_waitEvt()**, there is no such overhead – the Standard Runtime Library (SRL) simply returns the next available event on any device to the application. The application is free to implement its own event matching mechanism. In our case, we used the Standard Template Library (STL) map class, which has a more efficient implementation than just doing a linear search, and so this gave a better result.

This section provides test results that show the right combination of threads and channels for the extended asynchronous model with **sr_waitEvtEx().**

The data for the asynchronous single-thread model (which we recommend) shows that for a ten-board system, play and record will utilize 24% and 27% CPU time – versus 36% and 38% for the extended asynchronous model. And the system demonstrates good scalability: the correlation between the CPU utilization rate and number of channels is linear.

The following sections and figures report performance measurements from a system built as described in *Section VI. High Density Media Server Testing Environment.*

System Scalability: Play CPU Rate

Figure 1 shows the comparison between the recommended asynchronous model with **sr_waitevt()** and the extended asynchronous model with **sr_waitEvtEX()**. The recommended model clearly yields better results.

The system scales well for play CPU rate. It linearly correlates to the number of oards in the system. Figure 1 shows the CPU rate for an application running on one, two, four, six, eight, and ten boards.

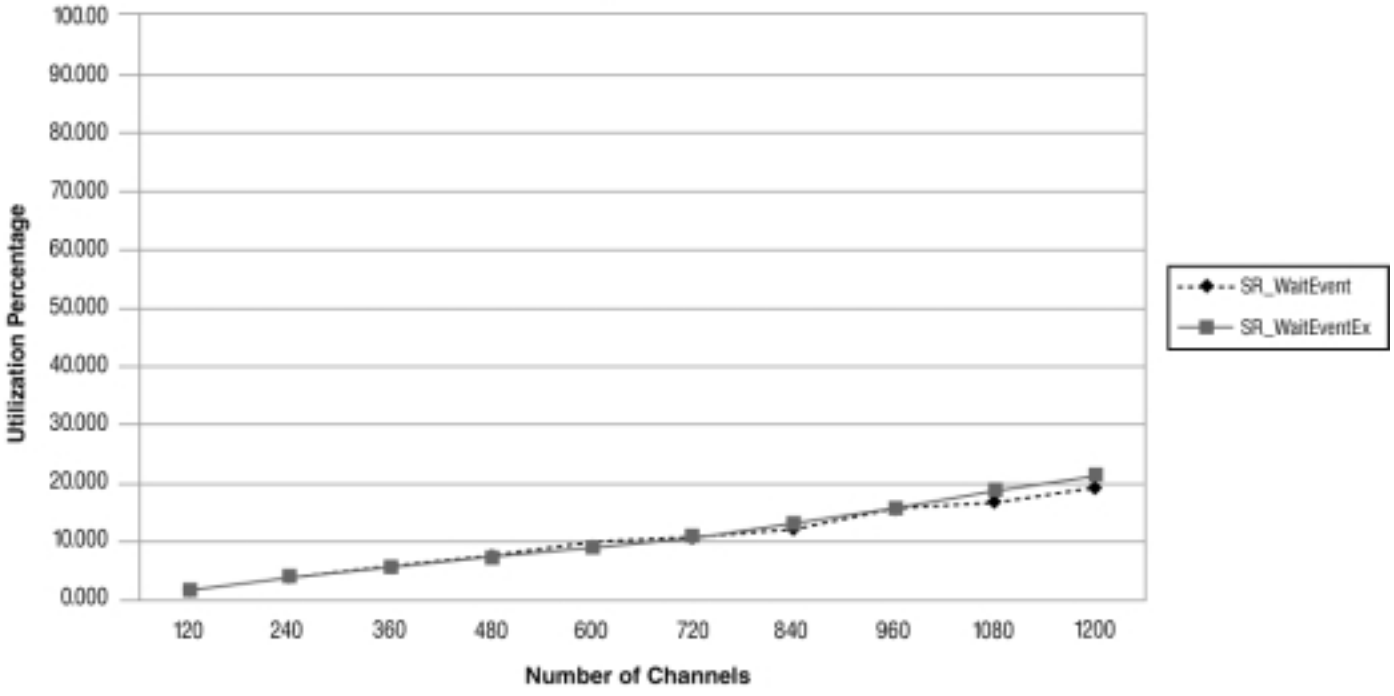


Figure 1: Comparison of PLAY CPU Rate and Scalability for Different Programming Models

System Scalability: Record CPU Rate

Figure 2 shows the comparison between the recommended asynchronous model with **sr_waitevt()** and the extended asynchronous model with **sr_waitEvtEX()**. The recommended model clearly yields better results.

The system scales well for record CPU rate. It linearly correlates to the number of boards in the system. Figure 2 shows the CPU rate for an application running on one, two, four, six, eight, and ten boards.

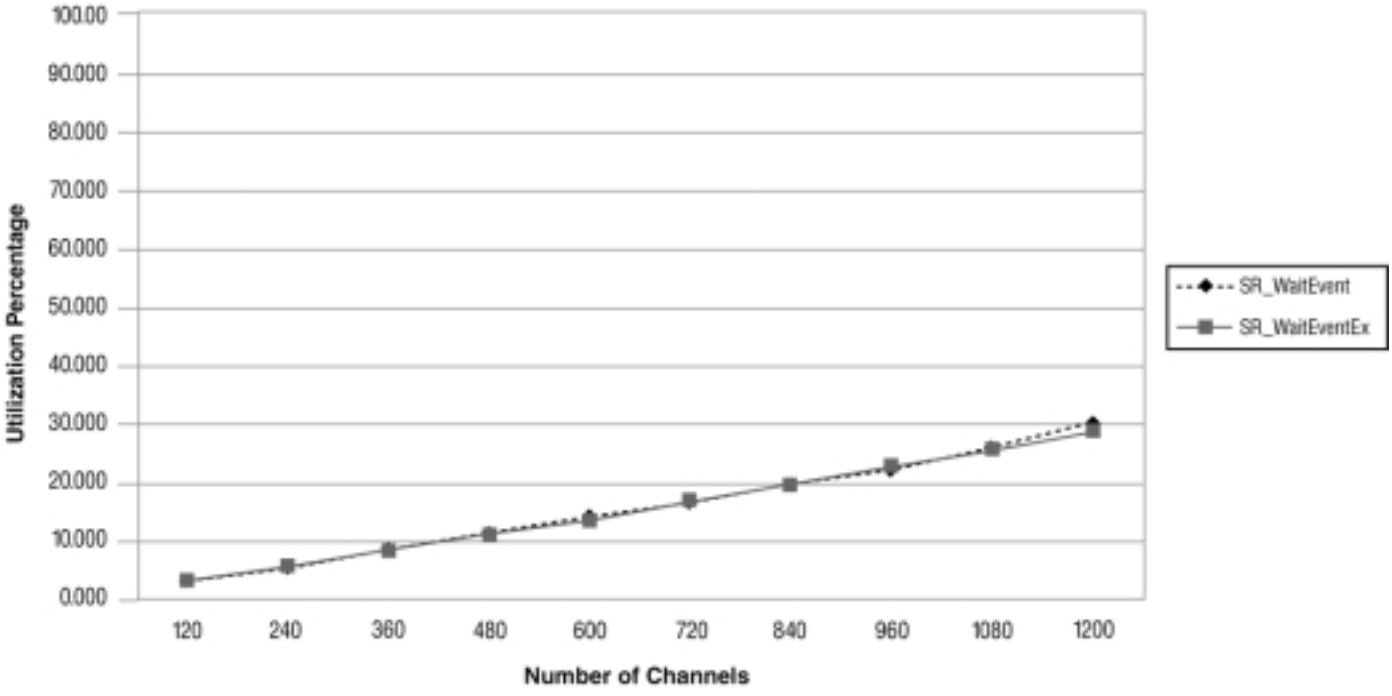


Figure 2: Comparison of RECORD CPU Rate and Scalability for Different Programming Models

1200-Channel Play CPU Rate with the extended asynchronous model

Figure 3 shows the different CPU rates versus the total threads running. When there is more than one thread running, each thread handles an equally distributed load. For example, if there are two threads running, each will handle 600 simultaneous plays.

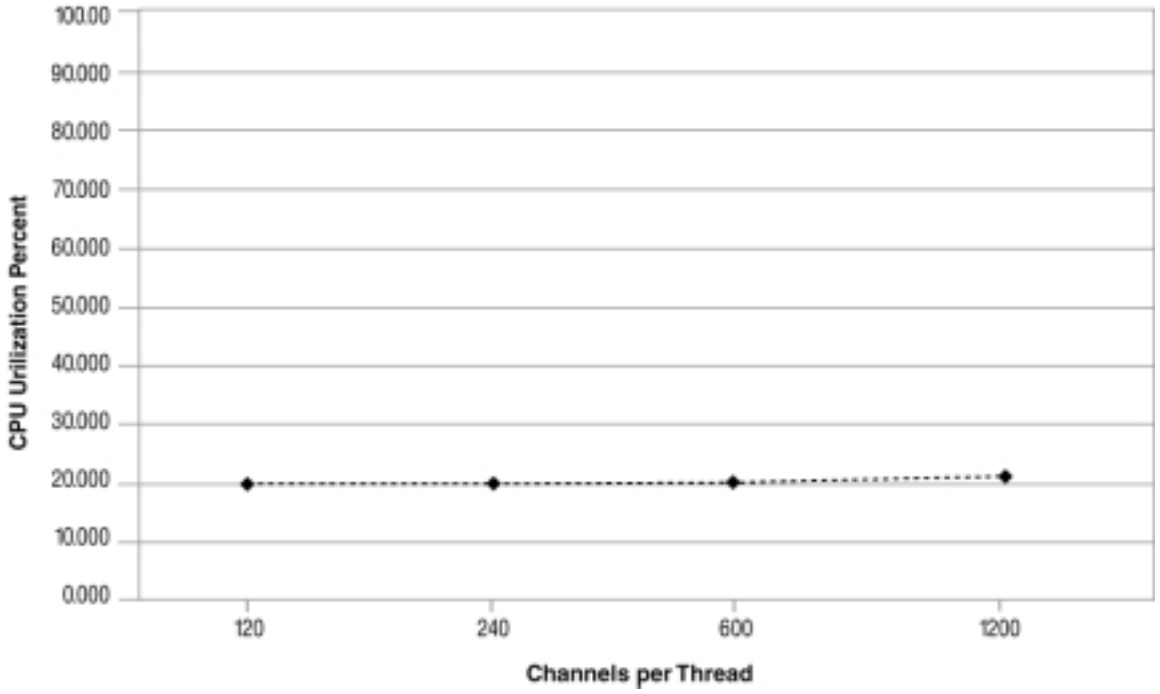


Figure 3: 1200-Channel Play CPU Rate

1200-Channel Record CPU Rate with the extended asynchronous model

Figure 4 shows the different CPU rates versus the total threads running. When there is more than one thread running, each thread handles an equally distributed load. For example, if there are two threads running, each will handle 600 simultaneous records.

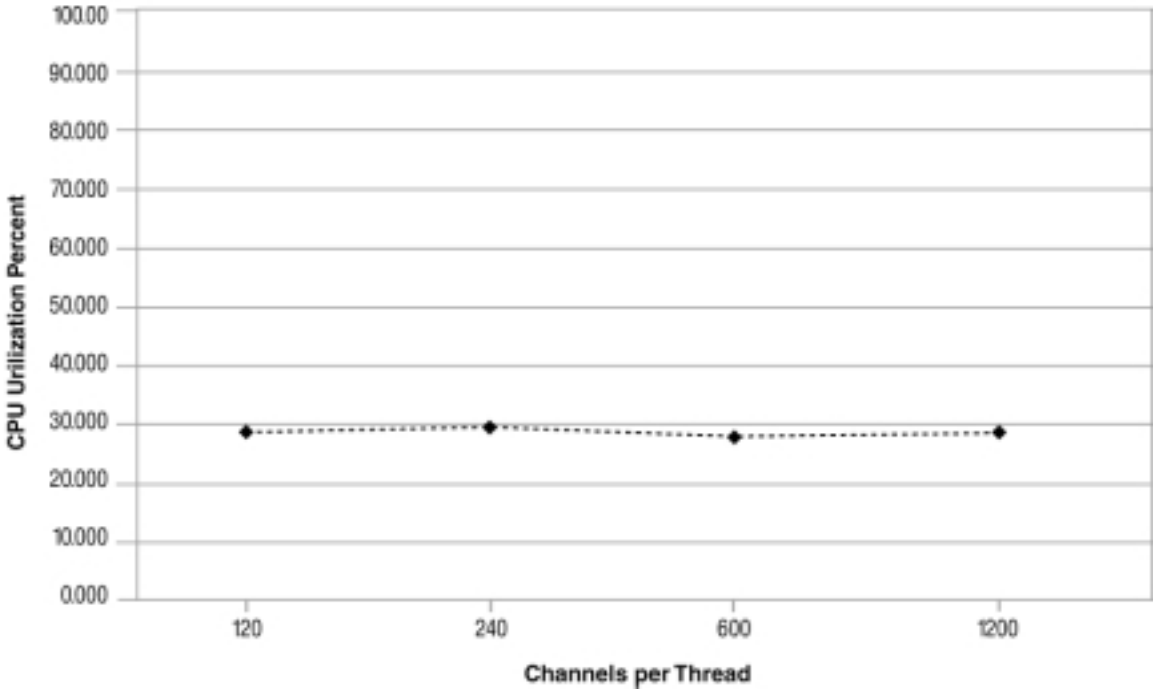


Figure 4: 1200-Channel Record CPU Rate

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/procs/perf/limits.htm, or call (U.S.) 1-800-628-8686 or 1-916-356-3104.

[1] Achieved in the testing environment described in Section VI. The newer DM/V-A Multifunction Series boards exhibit similar performance characteristics.

[2] See the *Voice Software Reference: Programmer's Guide for Windows* for more information about the **DV_TPT** and **DX_CST** data structures and recording.

[3] See the *Voice Software Reference: Programmer's Guide for Windows* for more information about the **DV_TPT** and **DX_IOTT** data structures and recording.

[4] See the *Voice Software Reference: Standard Runtime Library for Windows* for more information about the asynchronous programming model.

[5] Refer to the description of **dx_rec()** in the *Voice Software Reference: Programmer's Guide for Windows*

[6] For details, please refer to the *Voice Software Reference: Programmer's Guide for Windows*.

[7] See the *Voice Software Reference: Programmer's Guide for Windows* for more information about the **dx_stopch( )** function.

Intel Corporation
1515 Route Ten
Parsippany, NJ 07054
Phone: 1-973-993-3000
Fax: 1-973-993-3093

For more information
To learn more, visit our site on the World Wide Web at *www.intel.com*

Printed in the USA

00-8333-001 08/02