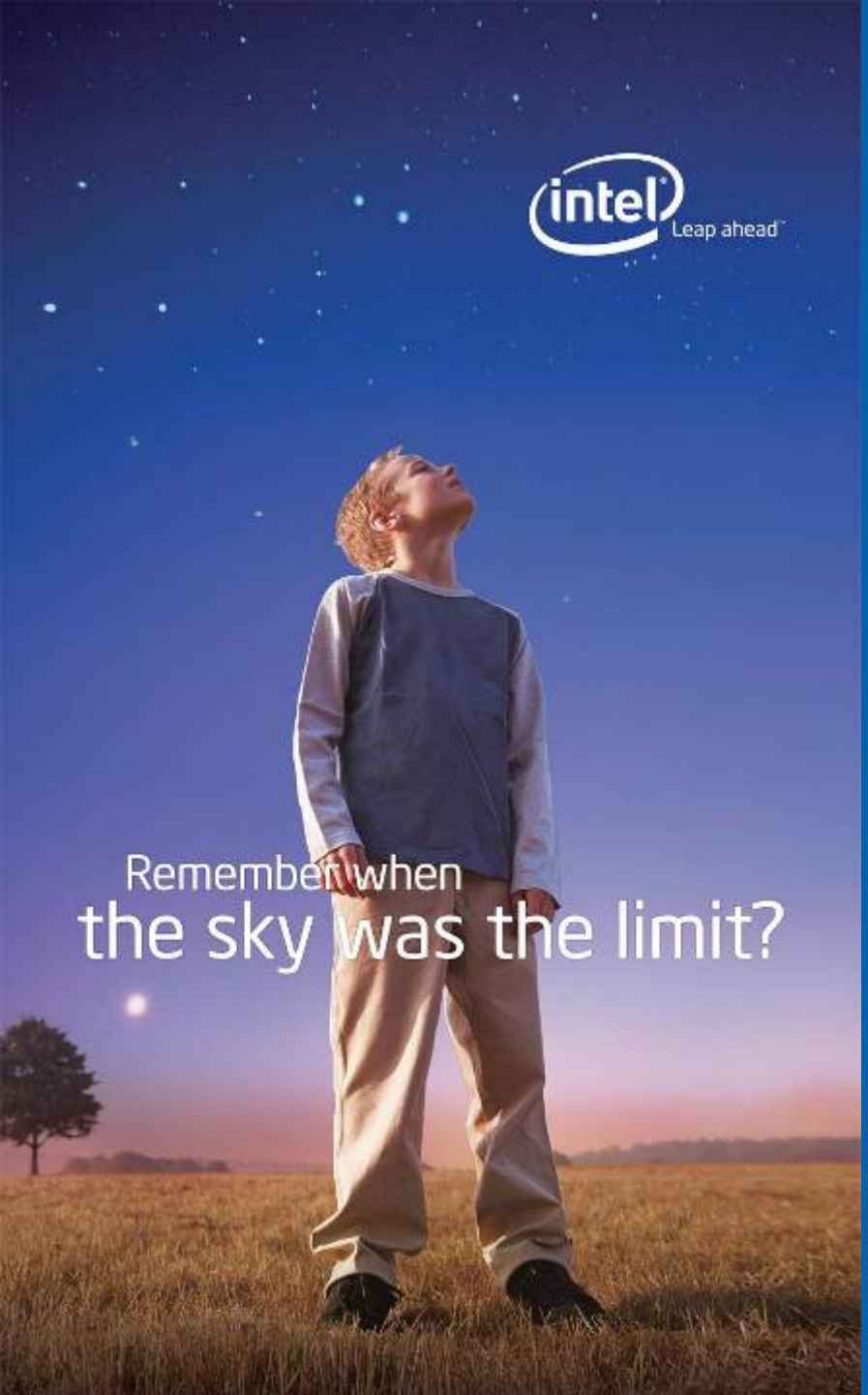


Jeremy Siadal

Technical Program Manager

Cluster Software and Technologies

Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.
- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- 64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.
  Intel Virtualization Technology requires a computer system with a processor, chipset, BIOS, virtual machine monitor (VMM) and applications enabled for virtualization technology. Functionality, performance or other virtualization technology benefits will vary depending on hardware and software configurations. Virtualization technology-enabled BIOS and VMM applications are currently in development.
- Intel, Intel Xeon, Intel Core microarchitecture, Intel Pentium-D, Intel. Leap ahead. logo, Xeon Inside logo and the Itanium 2 Inside logo and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
- Copyright © 2007 Intel Corporation.

2007 Predictions

From *Worldwide Technical Computing 2007 Top 10 Predictions*, published by IDC; by Earl Joseph Ph.D., Matthew Eastwood, Jean S. Bozman, Christopher G. Willard, Ph.D., Addison Snell

5. Clusters Will Become the "Industry Standard" HPC Server, and the Perception of Cluster Technology Will Move Quickly Out of the New Technology Category and Into Early and Late Majority Categories

6. Capacity-Oriented Markets Will Place Greater Emphasis on Non-Price/Performance Factors in Purchasing Clustered Systems

Intel® Cluster Tools Validation Lab

- Intel® Xeon® and Itanium® 2 processor platforms
- Five 9-36 node clusters (~100 systems)
- InfiniBand*, Myrinet*, and QsNet* II fabrics
- Rapid OS image changes
- Full IPMI managed infrastructure

Lab Provisioning Solution

- Requirements
  - Support Red Hat Enterprise Linux on all major Intel® Architectures
  - Rapid deployment and redeployment of clusters
  - Easy replacement of failed nodes
  - Support modified drivers and kernels
  - Easy installation of beta software and fabric stacks
- Decision
  - Implement OSCAR
  - Backup and restore head nodes from central server
  - Compute node images easily altered in chroot environment

4-6 months to fully implement

Endeavor Cluster

Rank	System	Procs	Rmax	Rpeak	Vendor
76	*Endeavor* Intel Cluster, Xeon 3.0 GHz dual core, Infiniband	1024	8564	12288	Intel

Building the Endeavor Cluster

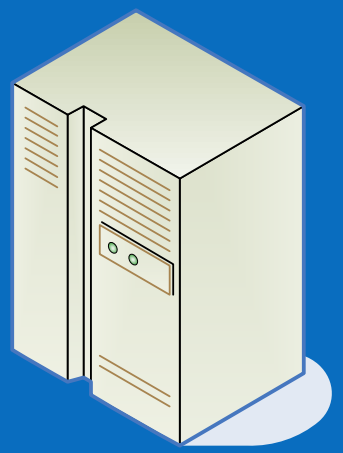
- Completed in May, 2006
- Configuration
  - 260 Nodes
  - Intel® Xeon® processors, 5100 series at 3.0 GHz
  - 8 GB memory per node
  - SDR InfiniBand
  - IPMI management
  - 120 TB permanent storage
- Provisioned using OSCAR 4.2.1
  - Required new driver integration and flexible provisioning
  - Head node required ~4 days to customize
  - Complete compute node provisioning < 4 hours

What have we learned?

Price/performance vs. Symmetric Multiprocessors (SMP)

SMP System

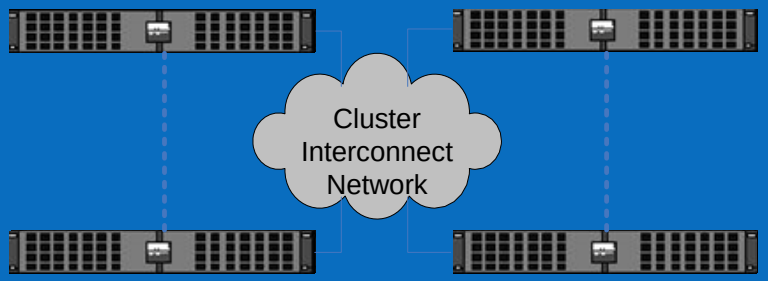
- Easy to build, closed environment
- Low latency shared memory
- Simple Programming model
- High cost of memory/cache design



SMP Price

~ $300k-500K

Volume server
Price < $3000



Cluster price

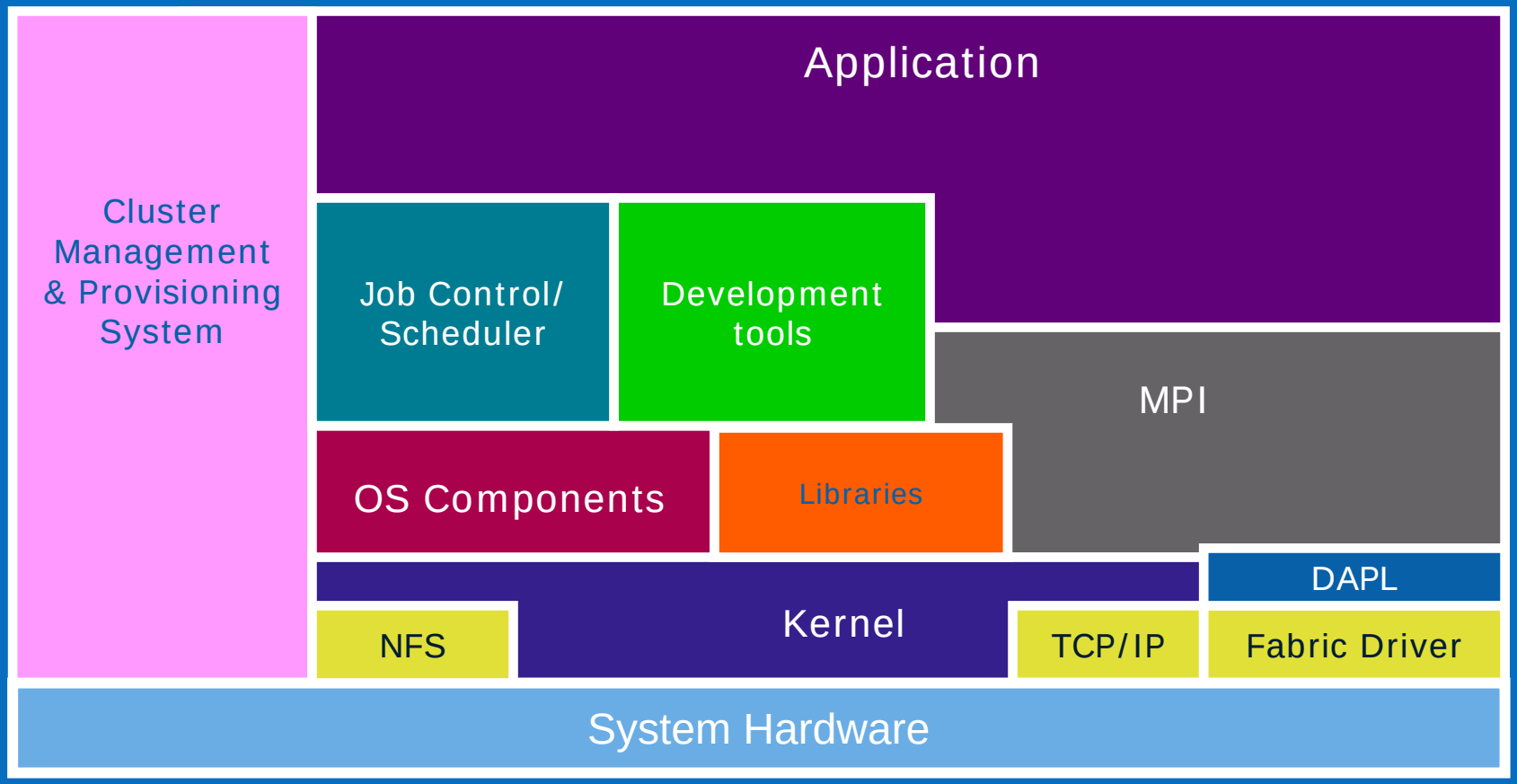
24×($3000+$1000/port) = $96,000

3-5× price/performance advantage

Cluster System

- Lots of cooperating components
- Higher latency for sharing
- Increased software complexity
- Lower cost

Cluster Components Schematic



Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

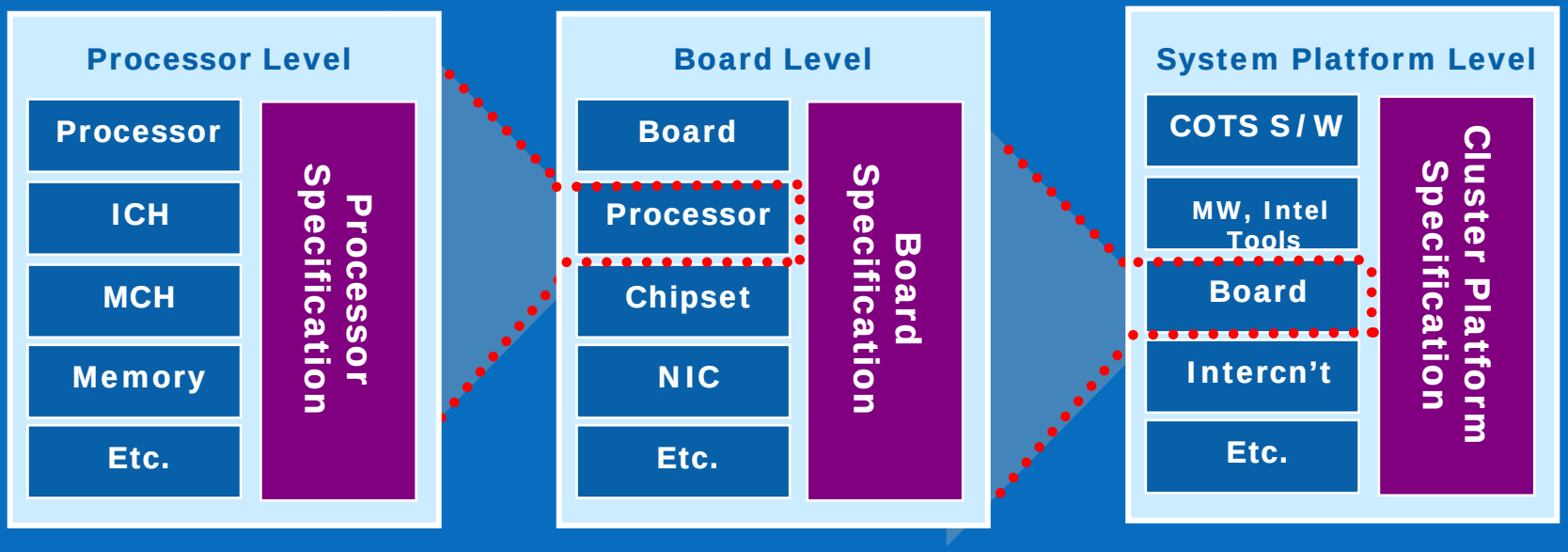
The Cluster Problem Statement

- Challenge
  - Simplify purchase, deployment, usage and management of clusters
  - Run the applications needed to do the job
- Status
  - System vendors invest effort and time to define, integrate, and validate clusters
  - Software vendors must support many design choices or limit supported systems
- Result
  - Higher cost to provide outstanding “out-of-box” experience

What’s needed?

A cluster with characteristics of an enterprise server

Solutions Platform for Clusters



- Create Cluster Platform standards
- Analogous to LSB standards for Linux
- Volume sweetspot at 32–64 nodes
- Designed for commercial ISV applications

Value Proposition of Standardized Cluster Platform

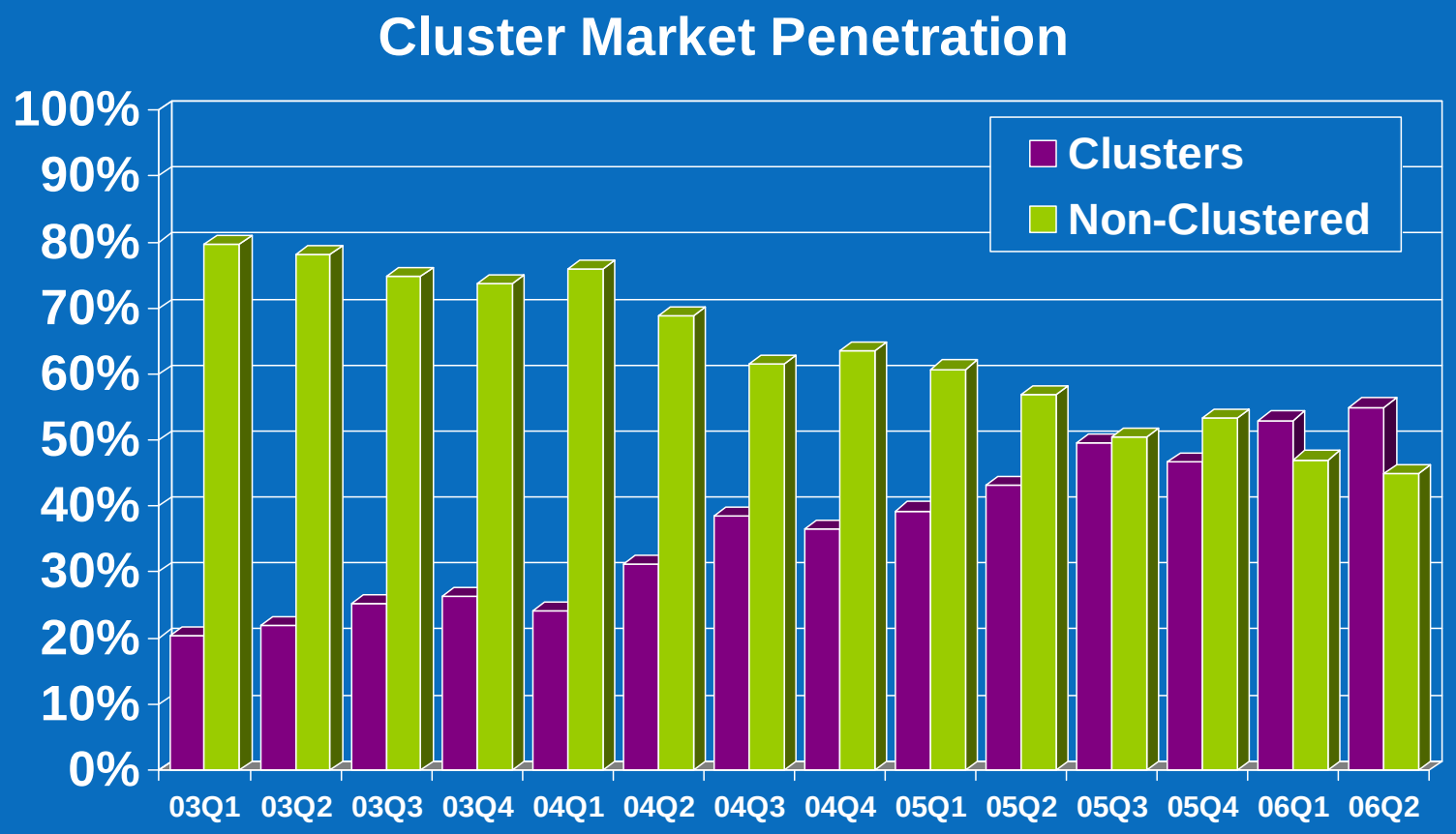
- For the Integrator
  - Platform will run many applications
  - ISVs will recommend standardized platforms
  - Users will look for standard solutions
- For the Software Vendor
  - Validate once, execute on other standard platforms
  - Less time and effort to validate software
  - Broader market reach with fewer application binaries
  - Less support cost

Broader Solution Portfolio, Increased Demand

HPC Growth In Clusters

Clusters @ 2010

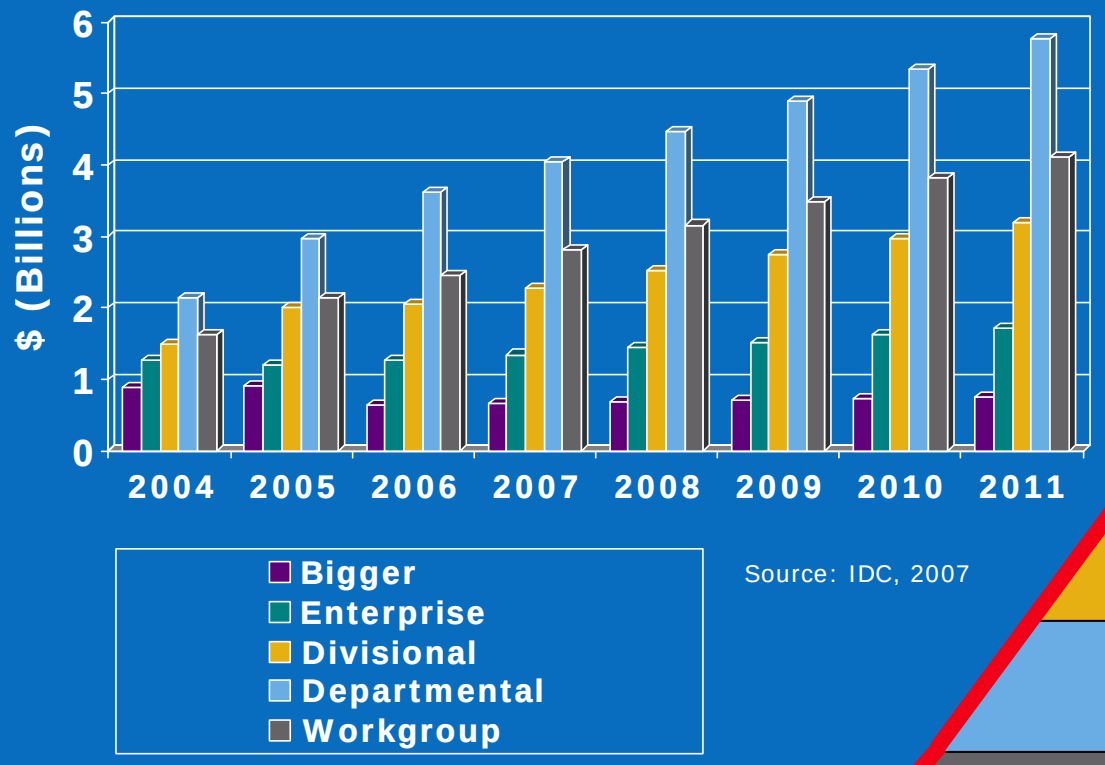
- ***$11b HPC revenue***
- ***75-85% of total HPC revenue***



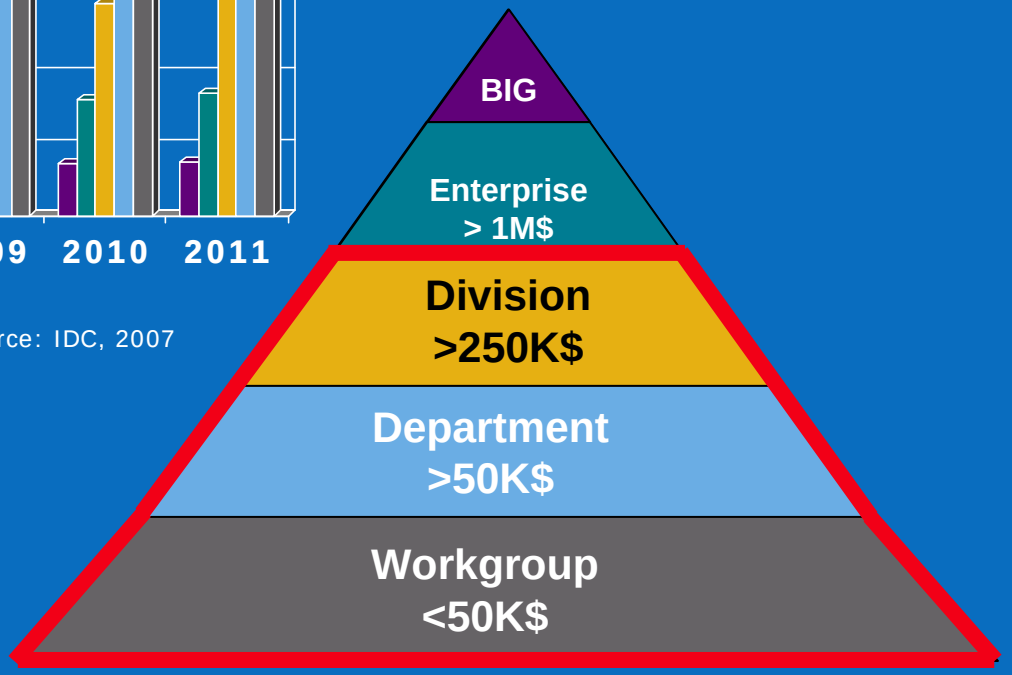
* All data from IDC Worldwide Technical Computing Systems Revenue 2006-2010 #201733 May 2006.

Volume Market Segment Growth

Worldwide HPC Server Revenue by Segment

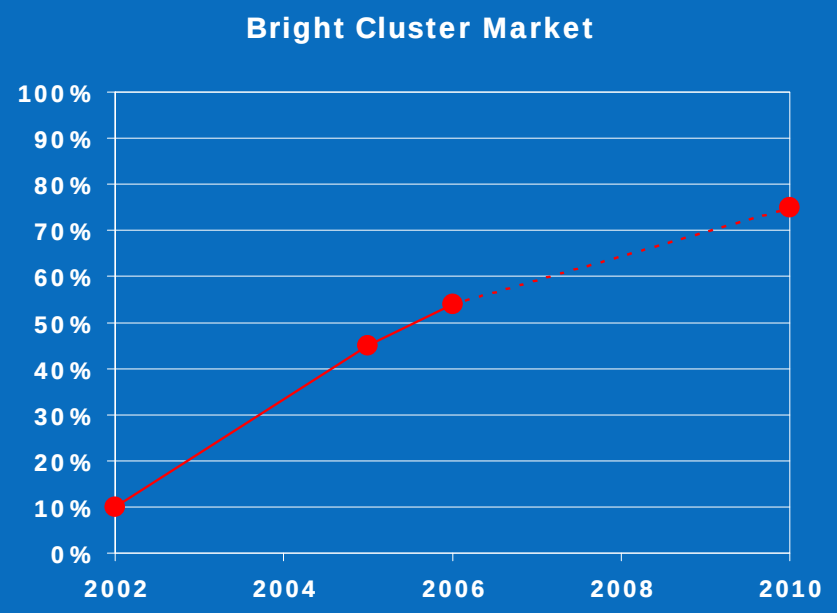


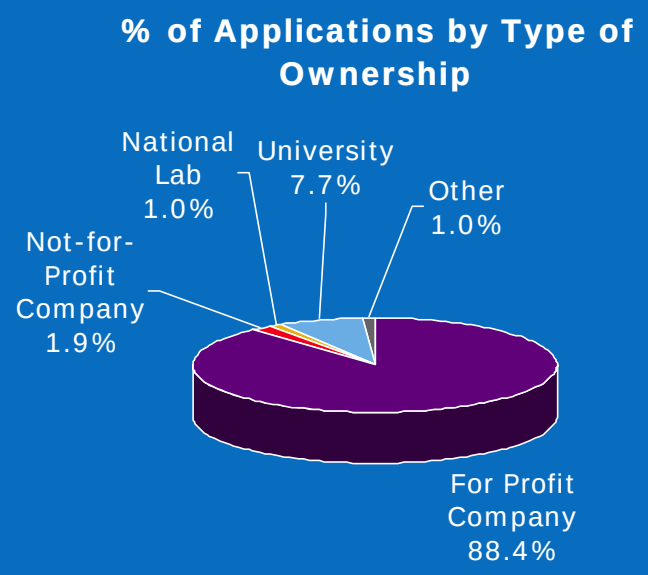
Source: IDC, 2007



HPC Trend to Commercial Clusters

Today, ~3% of HPC applications are open source

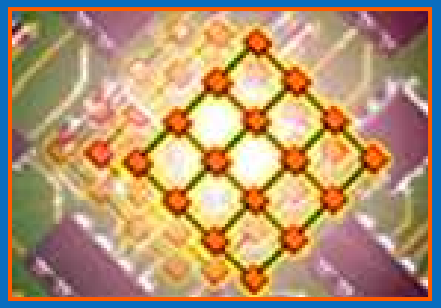


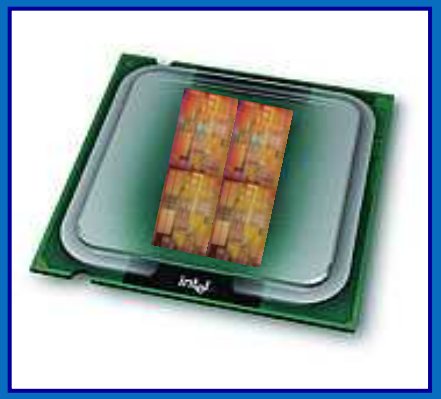


IDC White Paper: Study o f ISVs Serving the High Performance Computing Market: The Need For Better Application Software
July, 2005

What is Intel doing to enable the volume HPC cluster market?

Computing Parallelism Top-to-Bottom





Grid

Cluster

SMP (Nodes)

Multi-core (Sockets)

Multi-Threading (Cores)

SIMD

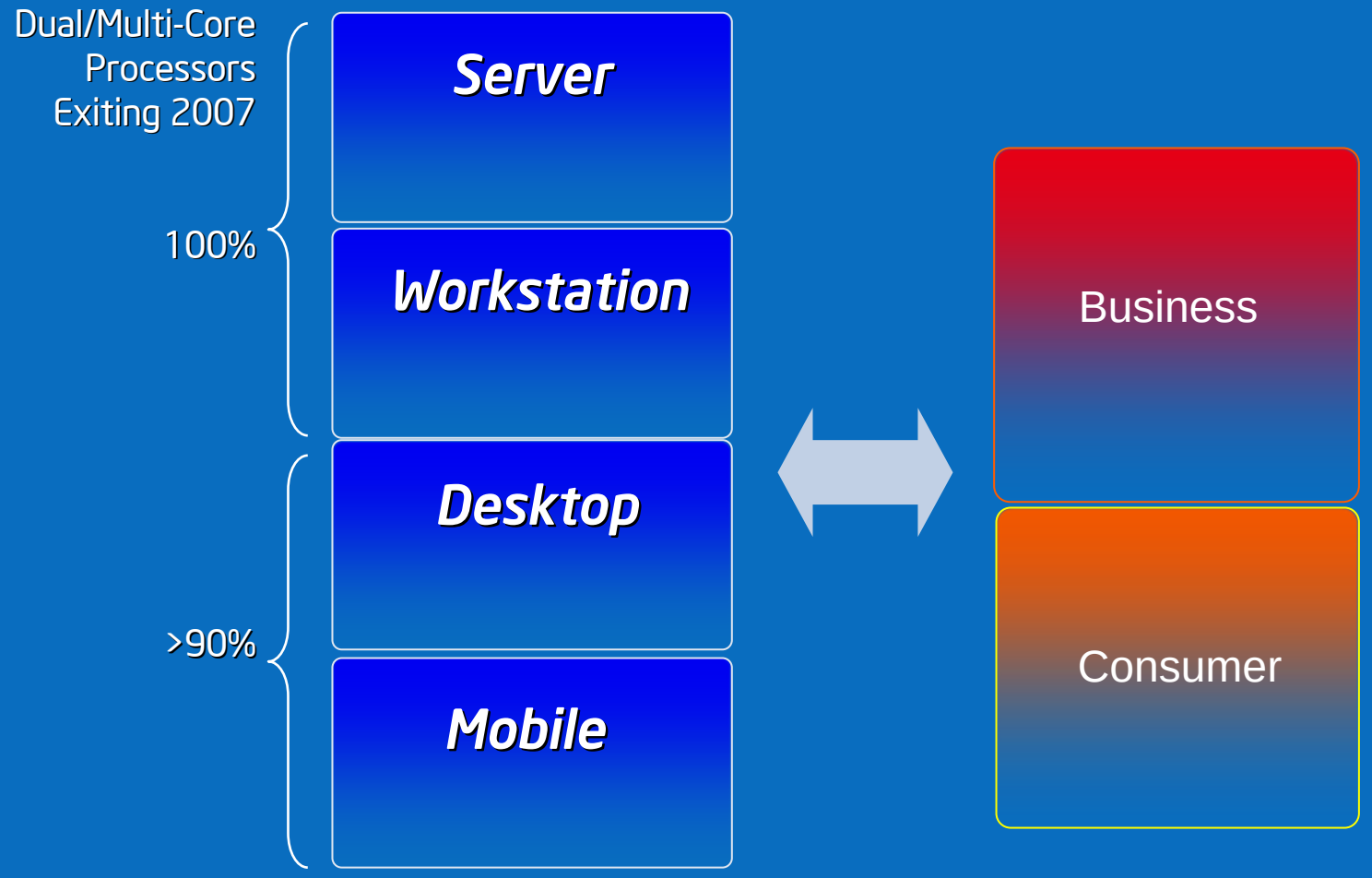
ILP

Ecosystem

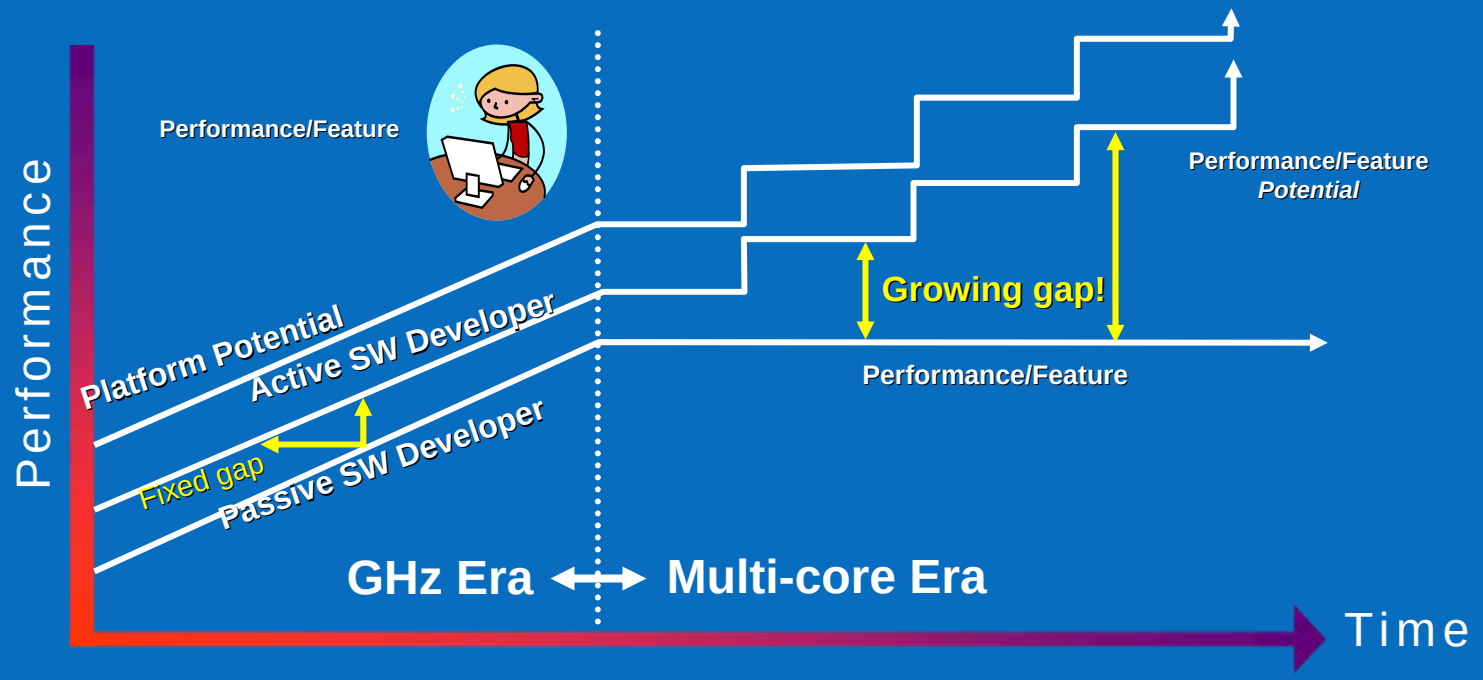
- Processors
- Software Tools
- Platforms
- Technologies
- Solutions

By innovation and technology leadership

Parallelism Goes Mainstream Across all Market-Segments



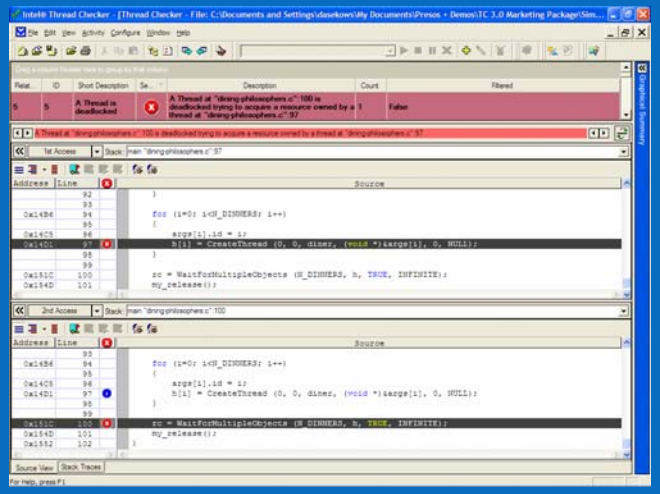
A New Era … A New Reality

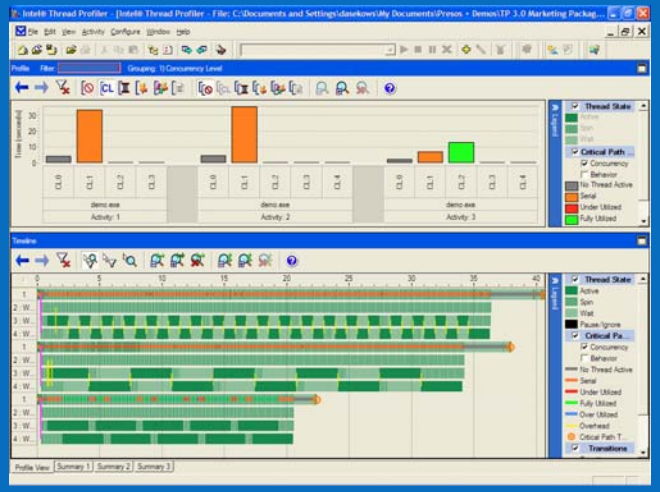


- End of the free lunch for passive software and application developers
- Implications for active developers also changes dramatically
- The multi-core “stairs” demand scalable, parallel applications

Intel® Threading Tools

- Intel® Thread Checker
  - Pinpoints elusive threading bugs in source code
- Intel® Thread Profiler
  - Pinpoints threading inefficiencies
- Intel® Threading Building Blocks
  - Ready to use parallel algorithms that easily plug into applications
  - Task based parallelism to abstract platform details





Less code to achieve parallelism

Example: 2D Ray Tracing Application

Windows* Threads

Thread Setup and Initialization

```
CRITICAL_SECTION MyMutex, MyMutex2, MyMutex3;
int get_num_cpus (void) {
    SYSTEM_INFO si;
    GetSystemInfo(&si);
    return (int)si.dwNumberOfProcessors;}
int nthreads = get_num_cpus ();
HANDLE *threads = (HANDLE *) alloca (nthreads * sizeof (HANDLE));
InitializeCriticalSection (&MyMutex);
InitializeCriticalSection (&MyMutex2);
InitializeCriticalSection (&MyMutex3);
for (int i = 0; i < nthreads; i++) {
    DWORD id;
    &threads[i] = CreateThread (NULL, 0, parallel_thread, i, 0, &id);;}
for (int i = 0; i < nthreads; i++) {
    WaitForSingleObject (&threads[i], INFINITE);
}
```

Parallel Task Scheduling and Execution

```
const int MINPATCH = 150;
const int DIVFACTOR = 2;
typedef struct work_queue_entry_s {
    patch pch;
    struct work_queue_entry_s *next;
} work_queue_entry_t;
work_queue_entry_t *work_queue_head = NULL;
work_queue_entry_t *work_queue_tail = NULL;
void generate_work (patch* pchin)
{  int startx, stopx, starty, stopy;
    int xs,ys;
    startx=pchin->startx; stopx= pchin->stopx;
    starty=pchin->starty; stopy= pchin->stopy;
    if(((stopx-startx) >= MINPATCH) || ((stopy-starty) >= MINPATCH)) {
      int xpatchsize = (stopx-startx)/DIVFACTOR + 1;
      int ypatchsize = (stopy-starty)/DIVFACTOR + 1;
      for (ys=starty; ys<=stopy; ys+=ypatchsize)
      for (xs=startx; xs<=stopx; xs+=xpatchsize) {
        patch pch;
        pch.startx = xs;
        pch.starty = ys;
        pch.stopx  = MIN(xs+xpatchsize-1,stopx);
        pch.stopy  = MIN(ys+ypatchsize-1,stopy);
        generate_work (&pch);}
} else {
      /* just trace this patch */
      work_queue_entry_t *q = (work_queue_entry_t *) malloc (sizeof
(work_queue_entry_t));
      q->pch.starty = starty; q->pch.stopy = stopy;
      q->pch.startx = startx; q->pch.stopx = stopx;
      q->next = NULL;
      if (work_queue_head == NULL) {
        work_queue_head = q;
      } else {
        work_queue_tail->next = q;
      }
      work_queue_tail = q;
    }
}
void generate_worklist (void)
{
    patch pch;
    pch.startx = startx;
    pch.stopx = stopx;
    pch.starty = starty;
    pch.stopy = stopy;
    generate_work (&pch);
}
bool schedule_thread_work (patch &pch)
{
    EnterCriticalSection (&MyMutex3);
    work_queue_entry_t *q = work_queue_head;
    if (q != NULL) {
      pch = q->pch;
      work_queue_head = work_queue_head->next;
    }
    LeaveCriticalSection (&MyMutex3);
    return (q != NULL);
}
generate_worklist ();


void parallel_thread (void *arg)
{
    patch pch;
    while (schedule_thread_work (pch)) {
      for (int y = pch.starty; y <= pch.stopy; y++) {
        for (int x=pch.startx; x<=pch.stopx; x++) {
          render_one_pixel (x, y); } }
      if (scene.displaymode == RT_DISPLAY_ENABLED) {
        EnterCriticalSection (&MyMutex3);
        for (int y = pch.starty; y <= pch.stopy; y++) {
          GraphicsDrawRow(pch.startx-1, y-1, pch.stopx-pch.startx+1,
(unsigned char *) &global_buffer[((y-starty)*totalx+(pch.startx-startx))*3]);
        }
        LeaveCriticalSection (&MyMutex3);
      }
    }
}
```

This example includes software developed by John E. Stone.

Intel® Threading Building Blocks

Thread Setup and Initialization

```
#include "tbb/task_scheduler_init.h"
#include "tbb/spin_mutex.h"
tbb::task_scheduler_init init;
tbb::spin_mutex MyMutex, MyMutex2;
```

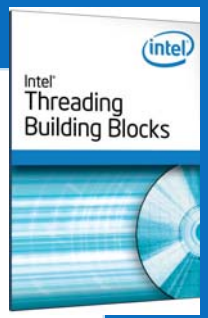
Parallel Task Scheduling and Execution

```
#include "tbb/parallel_for.h"
#include "tbb/blocked_range2d.h"
class parallel_task {
public:
    void operator() (const tbb::blocked_range2d<int> &r) const {
      for (int y = r.rows().begin(); y != r.rows().end(); ++y) {
        for (int x = r.cols().begin(); x != r.cols().end(); x++) {
          render_one_pixel (x, y);
        }
      }
      if (scene.displaymode == RT_DISPLAY_ENABLED) {
        tbb::spin_mutex::scoped_lock lock (MyMutex2);
        for (int y = r.rows().begin(); y != r.rows().end(); ++y) {
          GraphicsDrawRow(startx-1, y-1, totalx, (unsigned char
*) &global_buffer[(y-starty)*totalx*3]);
        }
      }
    }
    parallel_task () {}
};
parallel_for (tbb::blocked_range2d<int> (starty, stopy + 1,
grain_size, startx, stopx + 1, grain_size), parallel_task ());
```

Focus on work to do, not "how" (thread control) to manage threads

Intel® Threading Building Blocks offers platform portability on Windows*, Linux*, and Mac OS* through its cross-platform API. This code comparison shows the additional code needed to make a 2D ray tracing program, Tacheon, correctly threaded. This allows the application to take advantage of current and future multi-core hardware.

Thread for scalable performance vs. POSIX* Threads Benchmark: 2D Ray Tracing Application



Intel® Threading Building Blocks (Intel® TBB) for Linux* vs. POSIX* Threads

2D Ray Tracing Application

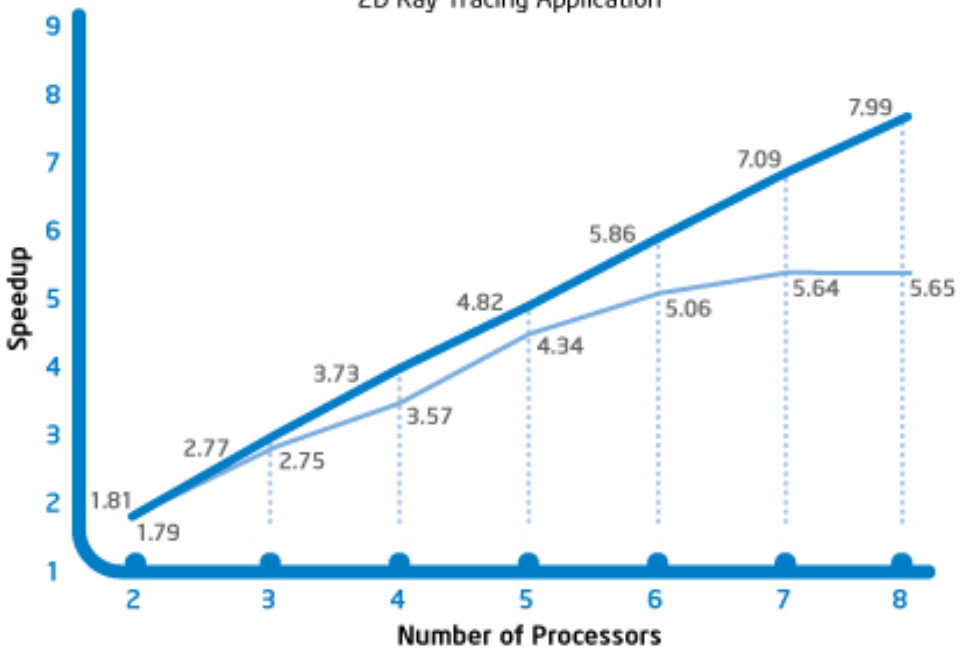


Figure 1: Speedup vs. Serial Implementation (Linux* 64-bit)

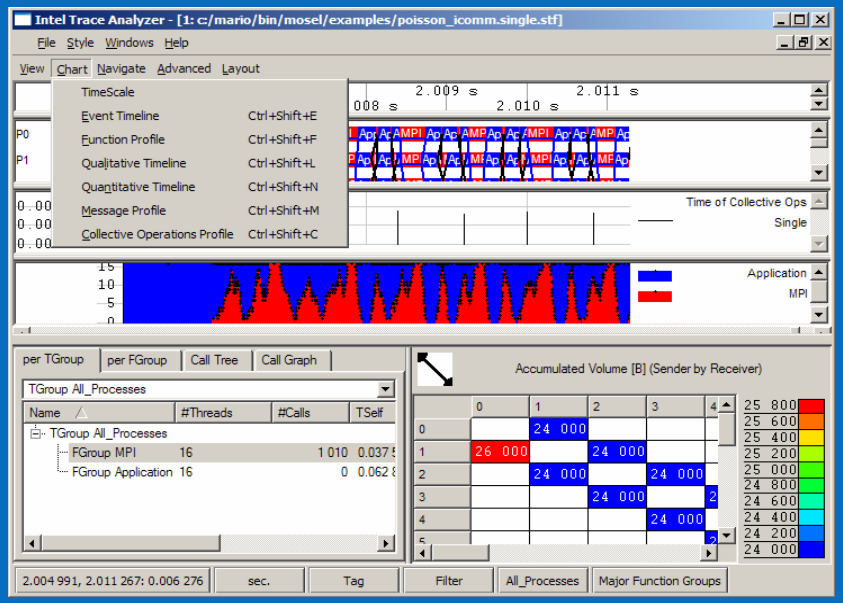
Test environment (Linux* 64-bit):
CPU Configuration: 8-processor system with 3.0 GHz 64-bit Intel® Xeon® processors
System RAM: 4GB main memory
OS: Linux Red Hat EL4
Intel® Compilers: Version 9.1
Intel® TBB: Version 1.0 Pre-release

Intel® TBB
POSIX* Threads

Figure 1 and **Figure 2** show performance on 2-8 cores versus Windows* Threads and POSIX* Threads. Since Intel® Threading Building Blocks is also scalable, no additional coding work is required to benefit from future increases in the number of cores per platform.

Intel® Cluster Toolkit
A complete MPI tools environment

- Intel® MPI Library
  - Network-independent MPI library
- Intel® Trace Analyzer and Collector
  - Analyze MPI processes and threads
- Intel® Math Kernel Library, Cluster Edition
- Intel® MPI Benchmarks



Intel® MPI Library:
A Simple Solution to Network Dependence

Apps See & Support Single Interconnect Fabric

Choose an interconnect fabric with no application code changes and no recompile. ISVs can distribute single binaries.

Intel® Trace Analyzer and Collector: Message Checker Tool

- Detects errors with data types, buffers, communicators, point-to-point & collective ops, deadlocks, message corruption, and hangs.
- Collects and analyzes MPI event data as the application runs
- Can trigger debugger breakpoints for in-place analysis

At LSTC we know how difficult MPI programming can be and invest considerable effort into making LS-Dyna robust. Message Checking with Intel Trace Analyzer and Collector identified a very subtle issue before it became a problem, saving us a significant amount of potential future debugging. No other tool of which I am aware has this capability or could have detected this problem."

*Brian Wainscott, Developer, LSTC/LS-Dyna**

LSTC Livermore Software Technology Corp.

* Other names and brands may be claimed as the property of others

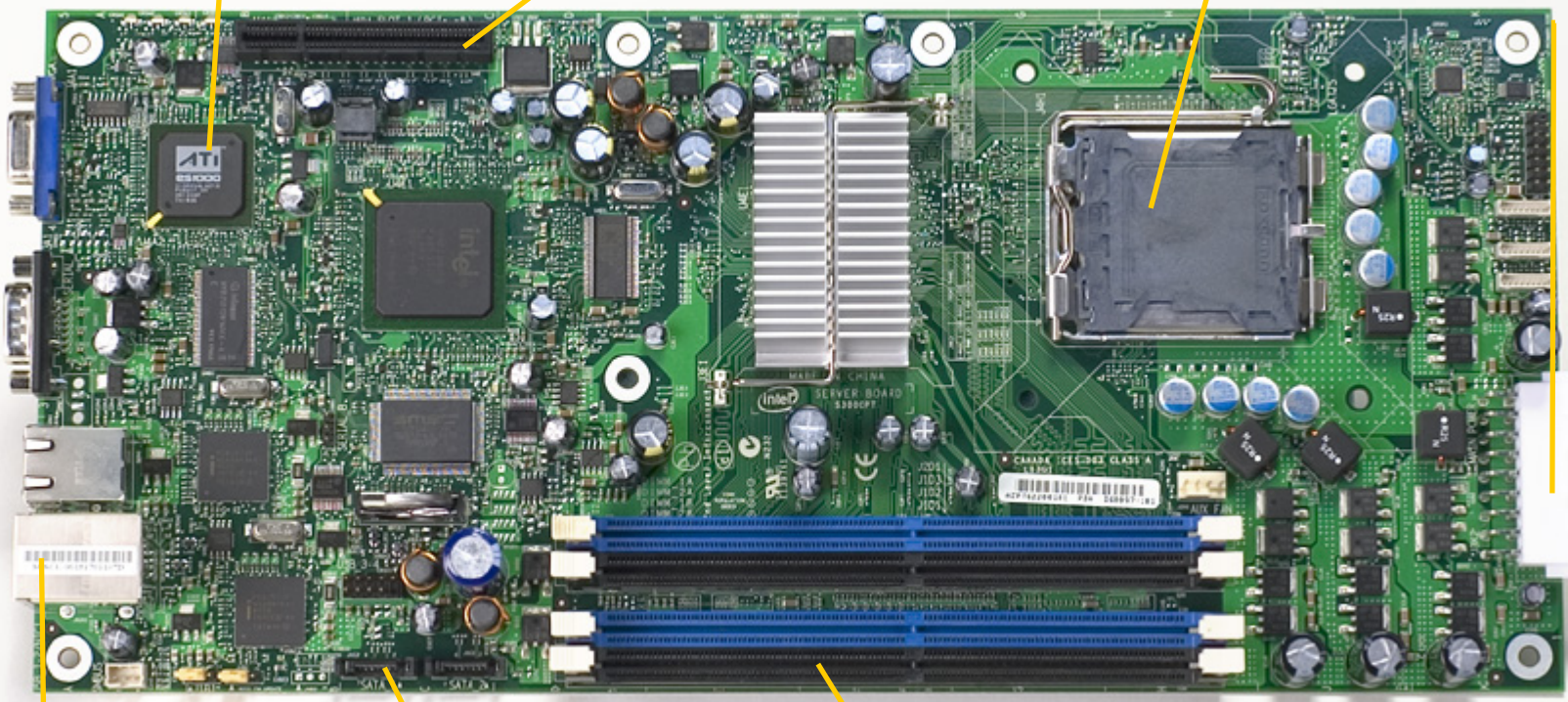
Cluster OpenMP*

- Runs (slightly modified) OpenMP codes on a commodity cluster
  - No need to explode your code and rewrite it in MPI
  - Exploit existing OpenMP codes which run on SMP machines on cheaper clusters
  - Available today as add-on for Intel 9.1 compilers
- Suitable Programs
  - Programs that scale successfully with OpenMP on SMP
  - Programs that have good data locality
  - Programs that use synchronization sparingly

Intel® Software Development Products

	COMPILER LIBRARIES	ALGORITHMS	DEBUG & QA	ANALYZE & TUNE
Sequential	Fortran	Math Kernel Library	Intel® Debugger	VTune™ Performance Analyzer
	C++	Integrated Performance Primitiives	Array Visualizer	
Multi-Core parallel	Compiler OpenMP Autoparallel	MKL/IPP Threaded	Thread Checker	Thread Profiler
Cluster parallel	MPI Library	Cluster Math Kernel Library	MPI Checker	Trace Analyzer & Collector
	Cluster OpenMP			

Intel® S3000PT Server Board



16MB graphics

PCIe x8 slot

Support for Intel® Xeon® and Core 2 (Duo & Quad) processors

Vertical or horizontal power connector option depending on SKU

Dual GbE

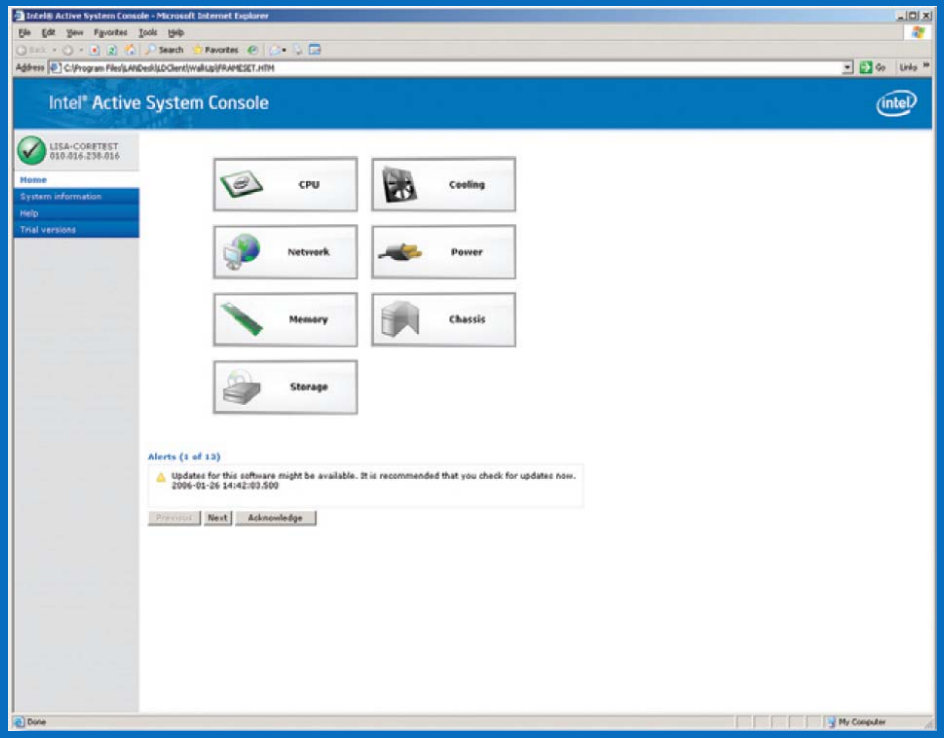
Dual SATA 3.0 Gbps ports

8GB Max ECC DDR2 533/667MHz

6x13” form factor (Half-ATX)

Server Management Features

- Provided by IPMI 2.0 and AMT
- Serial over LAN
- Sensor monitoring
- System Event Log
- Alerts
- Power Management Instrumentation
- Watchdog Timer
- Advanced Features
  - KVM/IP
  - USB media redirect
  - Dedicated LAN

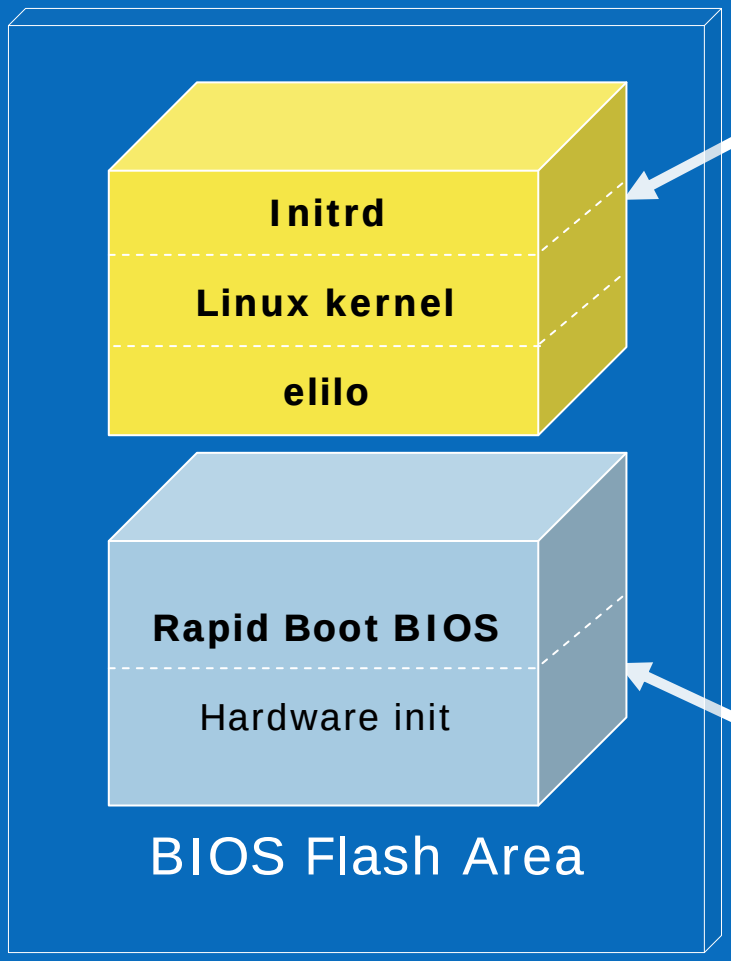


Intel Rapid Boot Toolkit Release 1 Overview

- Designed for dense computing infrastructures
- Eliminates BIOS features inappropriate to dense, lights-out computing
- Full automation of BIOS and firmware upgrades and configurations.
  - Minimal initialization and boots a user-defined payload
  - Rapid Boot Toolkit includes Fast Boot BIOS, example payloads, tools to build payloads and utilities
  - Currently supports Intel® Server Boards S5000PAL (Alcolu), S5000PSL (Star Lake), S3000PT (Port Townsend) and S3000AH (Aspen Hill)

http://developer.intel.com/design/servers/ism/rbt/

Intel Rapid Boot Toolkit



- User defined EFI payload
- OS agnostic (Linux shown), can use any interconnect fabric, any storage

Toolkit

- Payload Build Environment
- Payload Tools
- Example Payloads (Generic payload, Launcher Payload)

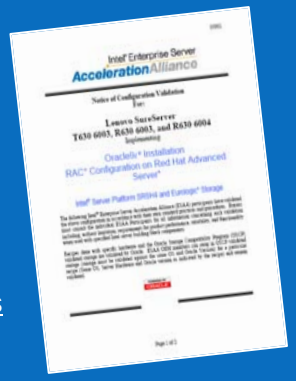
- Limited HW initialization
- Fast boot
- Handoff control to Payload

Intel® Enabled Server Acceleration Alliance (ESAA)

- An alliance of OEM's/resellers and application vendors providing validated server solutions to the end-user market.
- To Join or Obtain More Info:
  - Resellers go to: www.intel.com/go/esaa
  - Click on "New Reseller Sign Up".
  - ISV/IHVs, send an email to: esaamail@intel.com.

Solution Recipes

Sales Guides



Certificates

ESAA Recipes for Intel® Server Board S3000PT	
24INTC210000000013-01	BIOS Provisioning for Intel© Server Board S3000PT
24PLTF240000000012-01	Platform Open Cluster Stack (OCS)* 4.1.1- 1.1 on Red Hat Enterprise Linux* 4.0 - Intel® Server Board S3000PT
24PLTF240000000100-01	How to Build a Cluster - Intel® Server Board S3000PT
24SLVS030000000121-01	High-Performance Computing (HPC) Cluster Installation using SilverStorm* InfiniBand* Interconnect on Linux* - Intel® Server Board S3000PT

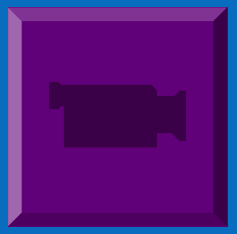
Conclusions

- There is need for standardized cluster platforms
- Market trend is increased growth in volume commercial clusters
- Intel provides a broad ecosystem to maximize volume cluster performance and simplify management
- But what about Petascale…

Industry Trend to Multi / Many-Core

Intel Tera-Scale Computing Research Program:
www.intel.com/go/terascale

Many-Core



Multi-Core

Dual-Core

Hyper-Threading

Multi Processor

Energy Efficient Petascale with Multi-threaded Cores

intel®
Leap ahead™