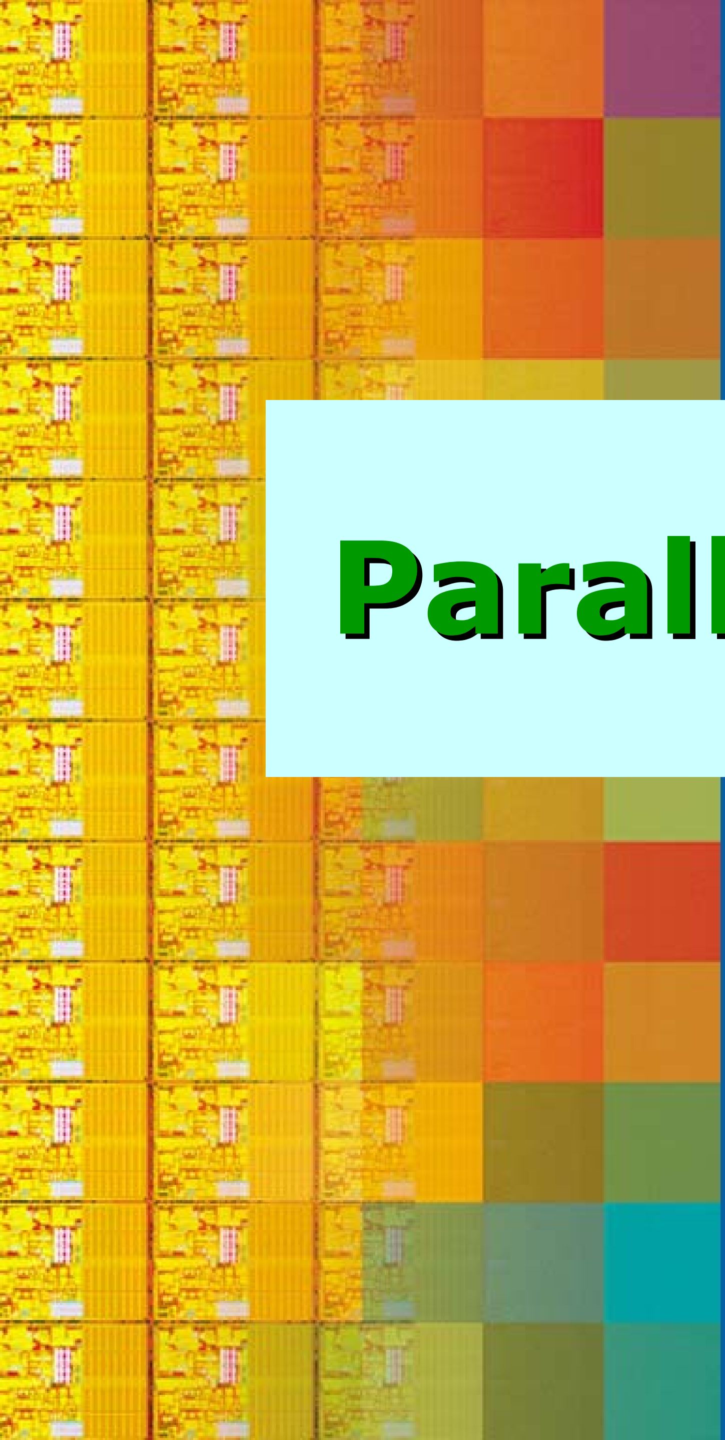


Parallel Software 2.0

Wei Li

Senior Principal Engineer
Intel Corporation

CS243 Stanford 3/16/06

Agenda

Major Technological Change

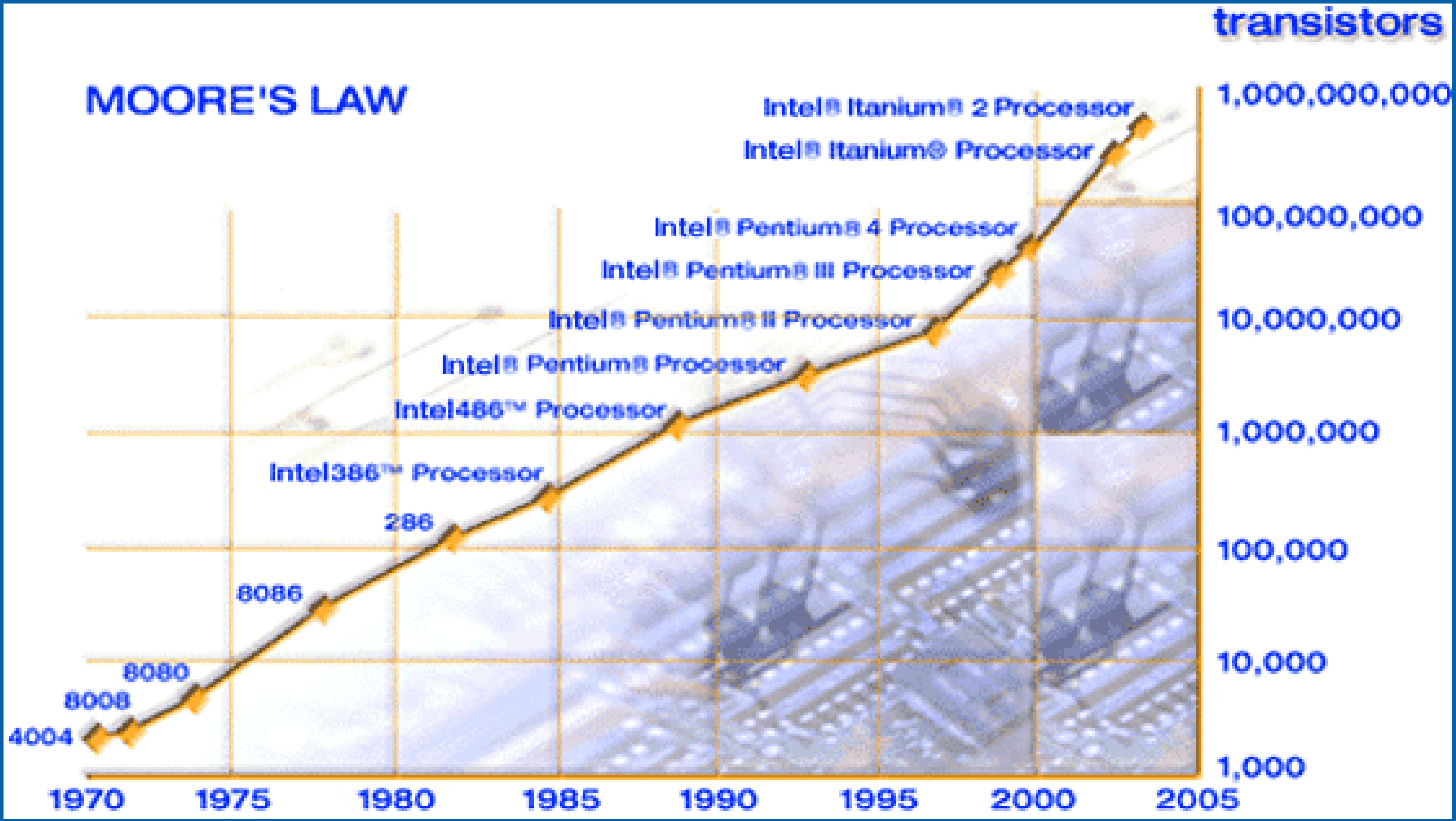
Software Response

Parallel Software 2.0

Quiz

Moore’s law states which of the following roughly doubles every 2 years?

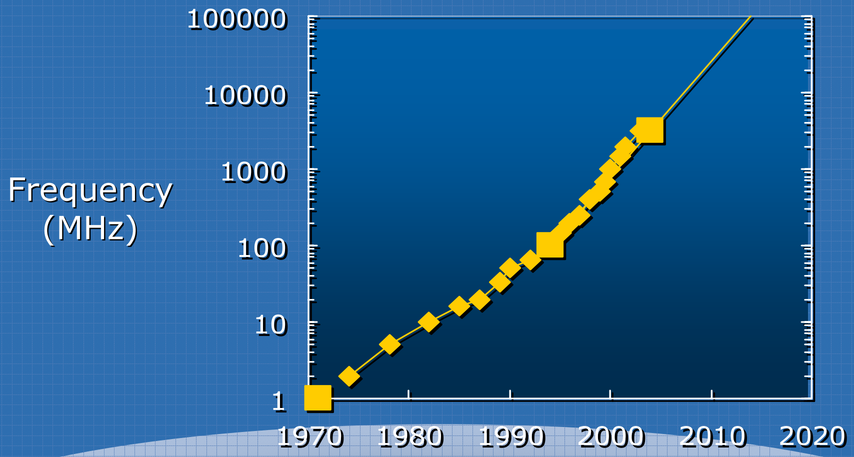
1. Frequency
2. Performance
3. Transistors
4. Transistor Density



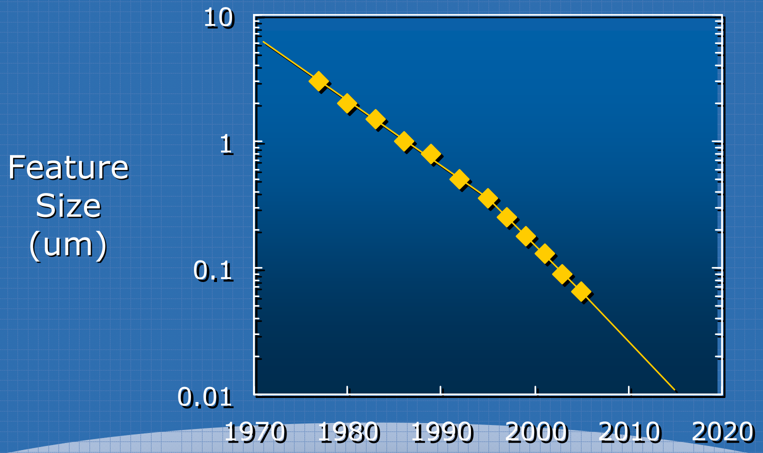
from www.intel.com

Historical Driving Forces

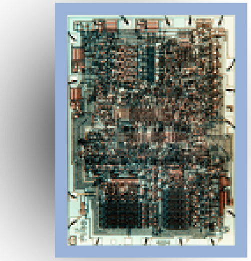
Increased Performance via Increased Frequency



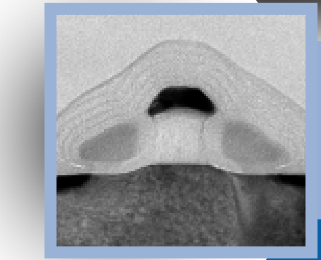
Shrinking Geometry



1946
20 Numbers
in Main Memory



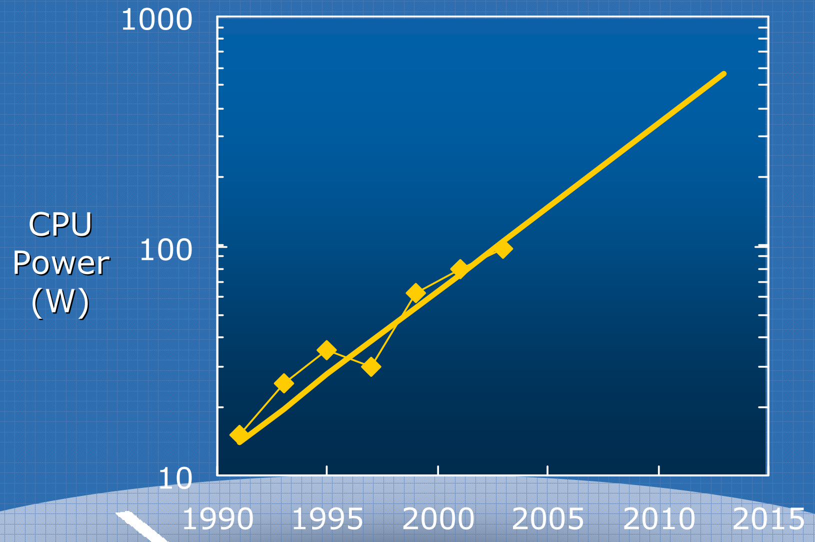
1971
I4004 Processor
2300 Transistors



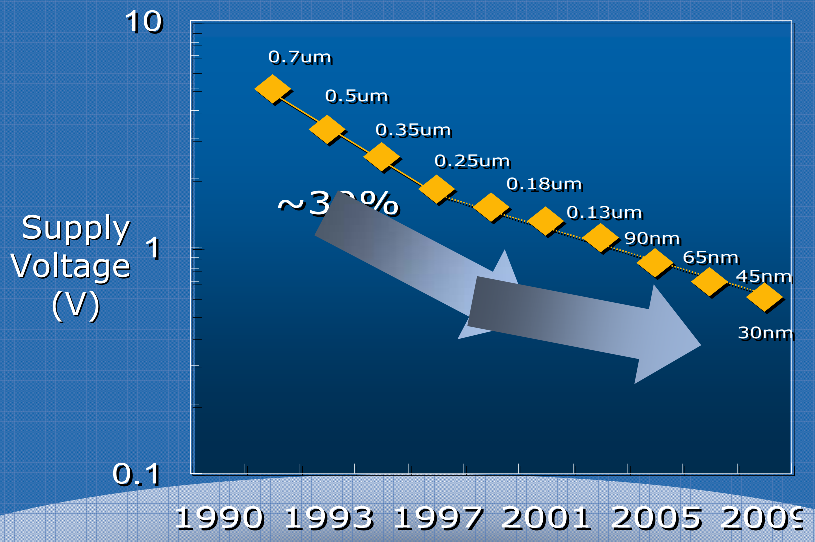
2005
65nm
1B+ Transistors

The Challenges

Power Limitations



Diminishing Voltage Scaling

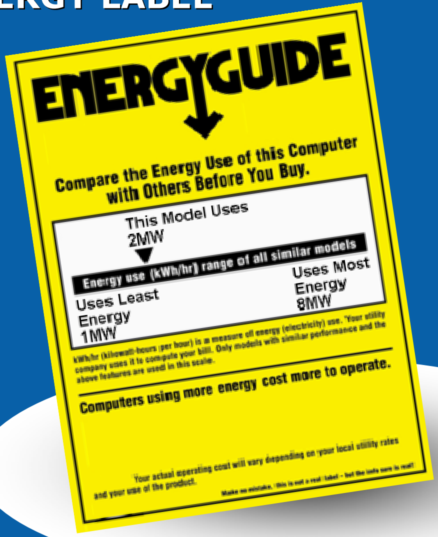


Power = Capacitance x Voltage^2 x Frequency
also
Power ~ Voltage^3

Energy:
The Next Frontier

Energy Efficient Performance – High End

DATACENTER
"ENERGY LABEL"



ENERGYGUIDE

Compare the Energy Use of this Computer with Others Before You Buy.

This Model Uses
2MW

Energy use (kWh/hr) range of all similar models

Uses Least Energy
1MW

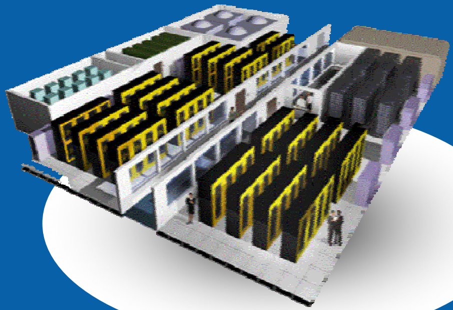
Uses Most Energy
8MW

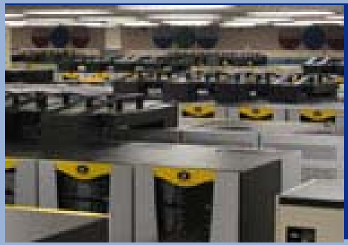
kWh/hr (kilowatt-hours per hour) is a measure of energy (electricity) use. Your utility company uses it to compute your bill. Only models with similar performance and the above features are used in this scale.

Computers using more energy cost more to operate.

Your actual operating cost will vary depending on your local utility rates and your use of the product.

Make no mistake, this is not a real label – but the info sure is real!





NASA Columbia
2 MWatt
60 TFlops goal
10,240 cpus – Itanium II
$50M

Source: NASA

30,720 Flops/Watt
1,288 Flops/Dollar

Computational Efficiency

17,066 Flops/Watt
467 Flops/Dollar

ASC Purple
6 MWatt
100 TFlops goal
12K+ cpus – Power5
$230M

Source: LLNL

The Classic Tradeoff

Higher
Top Speed and Acceleration

Increased
Range and Economy

The Real Challenge

Capabilities

Performance

Energy-Efficiency

Reducing Power with Voltage Scaling

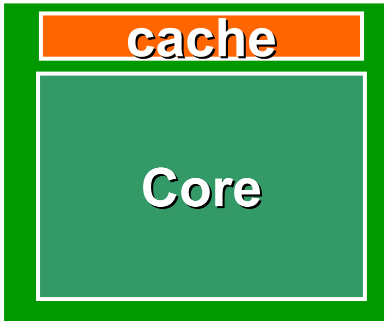
- Power = Capacitance * Voltage^2 * Frequency
- Frequency ~ Voltage in region of interest
- Power ~ Voltage^3
- 10% reduction of voltage yields
  - 10% reduction in frequency
  - 30% reduction in power
  - Less than 10% reduction in performance

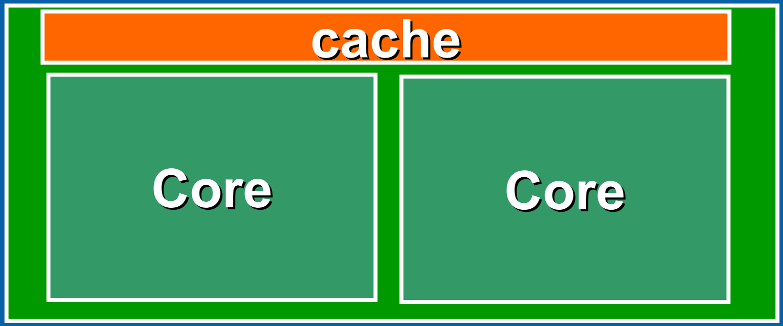
Rule of Thumb

Voltage	Frequency	Power	Performance
1%	1%	3%	0.66%

Dual Core example of Voltage Scaling

Voltage	Frequency	Power	Performance
1%	1%	3%	0.66%





Voltage = 1

Freq = 1

Area = 1

Power = 1

Perf = 1

Voltage = - 15%

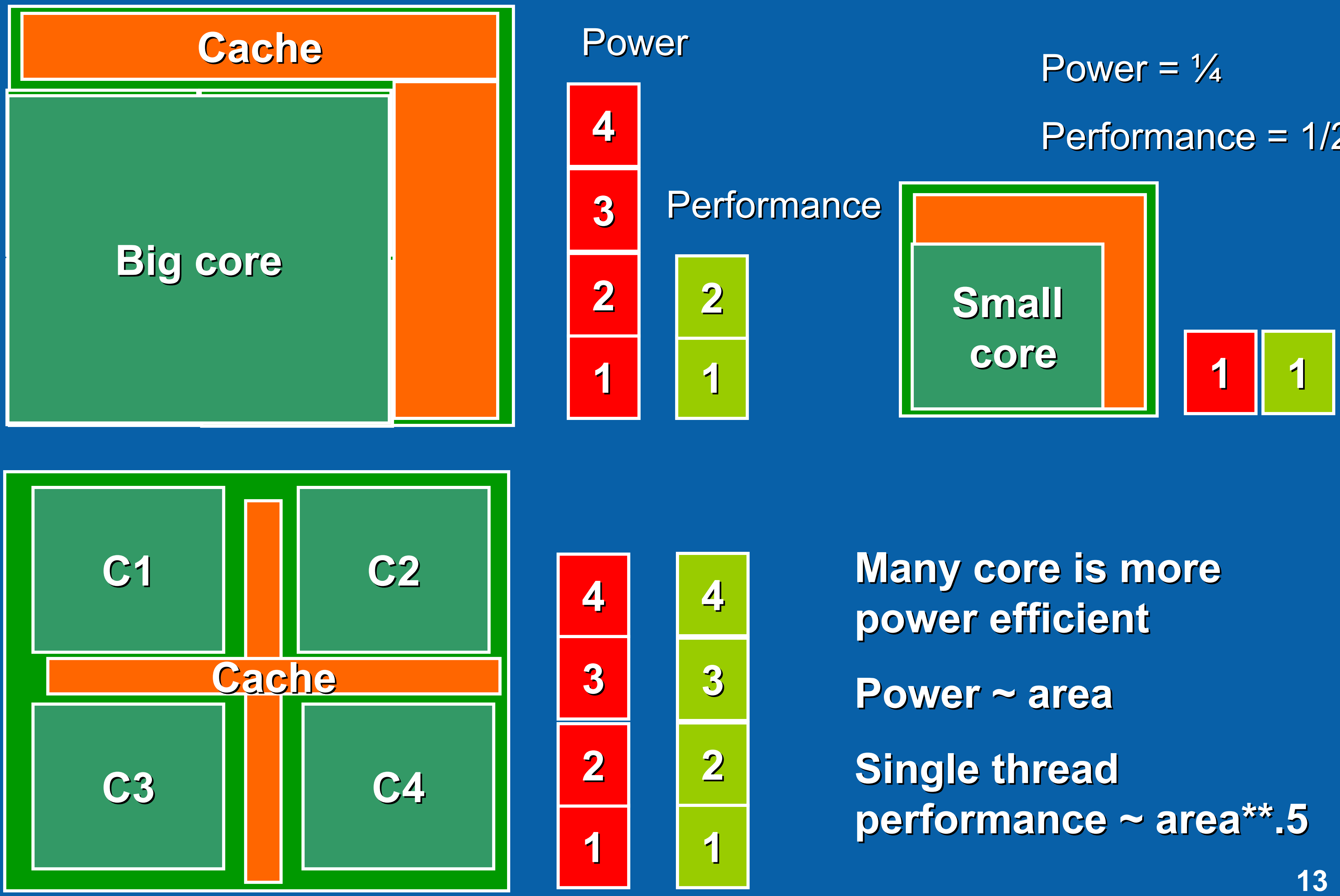
Freq = - 15%

Area = 2

Power = 1

Perf = ~1.8

Multiple cores deliver more performance per watt



Many core is more power efficient

Power ~ area

Single thread performance ~ area.5**

Moore’s Law will provide transistors

Intel process technology capabilities

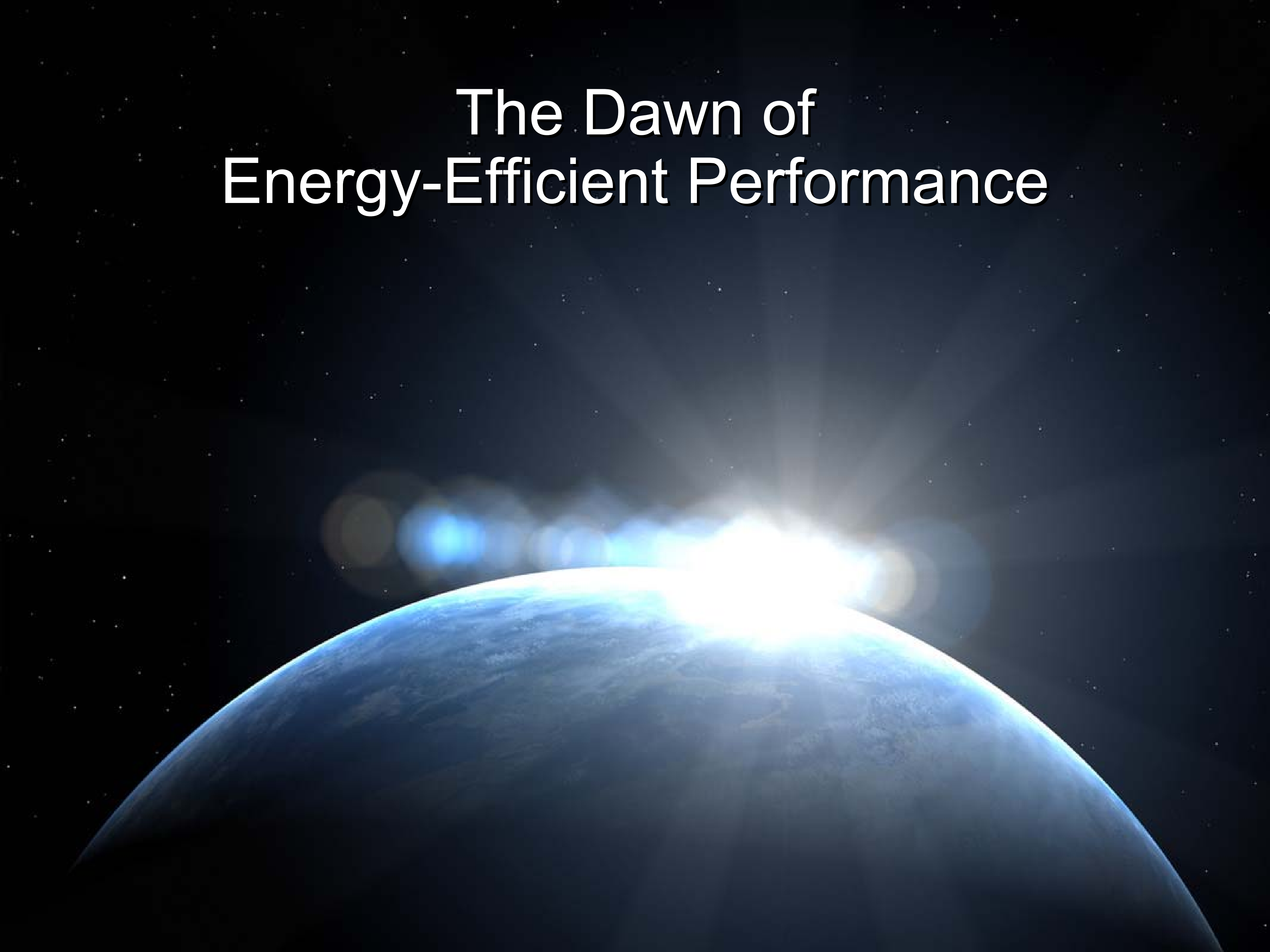
High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Feature Size	90nm	65nm	45nm	32nm	22nm	16nm	11nm	8nm
Integration Capacity (Billions of Transistors)	2	4	8	16	32	64	128	256

Use transistors for

- **Multiple cores**
- **On-core memory (caches)**
- **New features (*Ts)**

Multiple cores and caches address power and memory latency issues

The Dawn of Energy-Efficient Performance



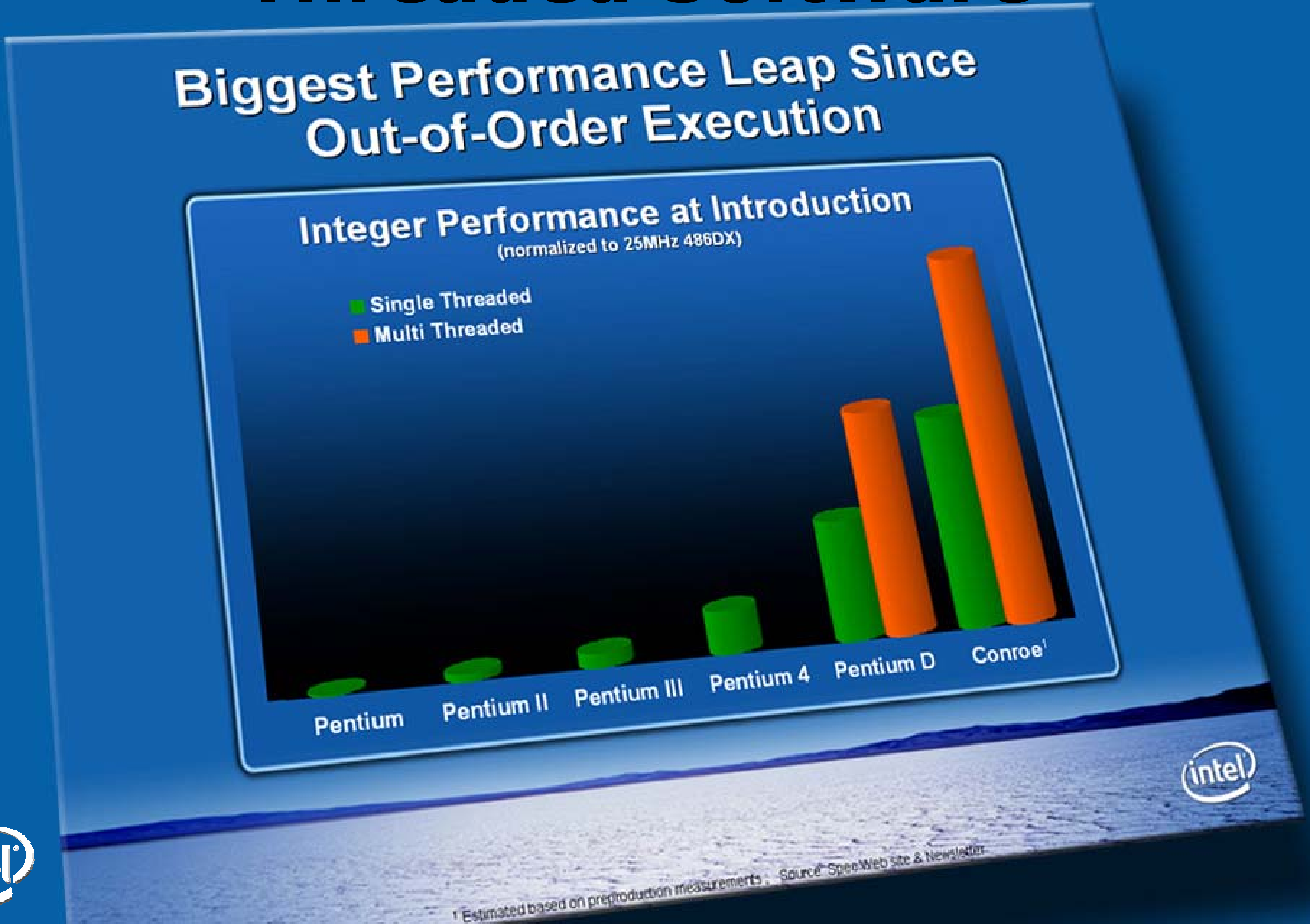
Agenda

Major Technological Change

Software Response

Parallel Software 2.0

Multi-Core Platforms Demand Threaded Software



Biggest Performance Leap Since Out-of-Order Execution

Integer Performance at Introduction

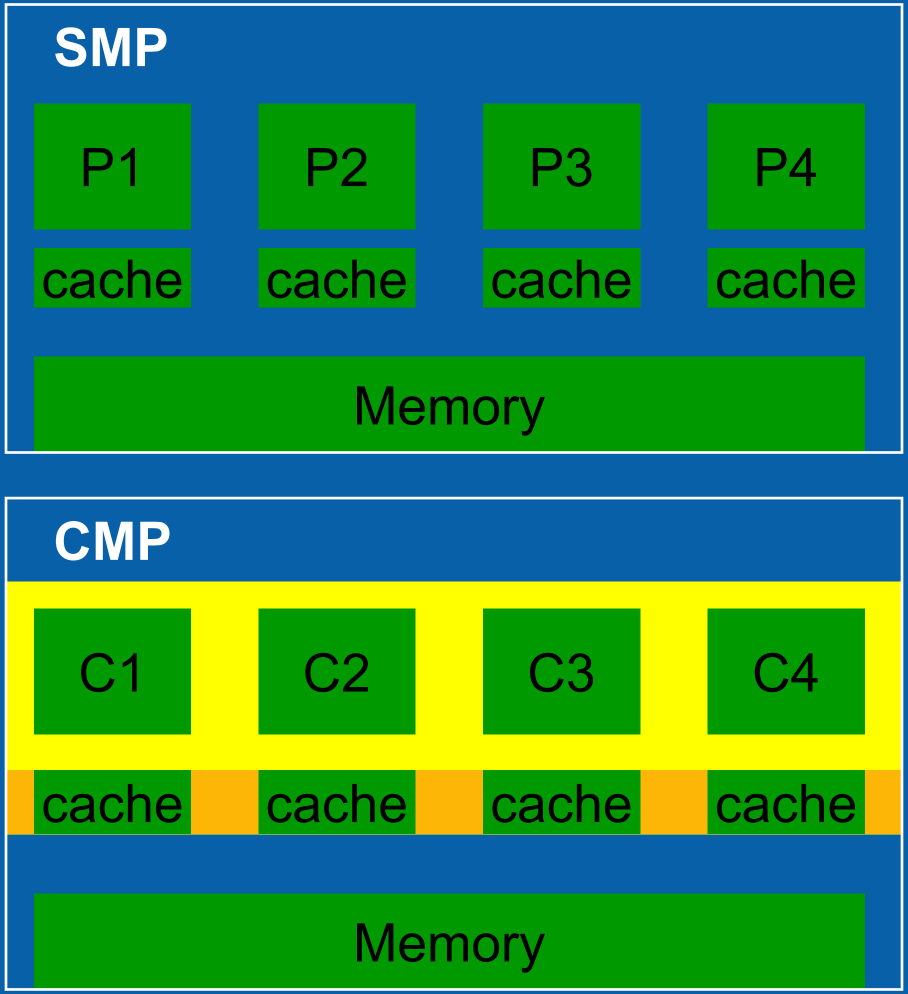
(normalized to 25MHz 486DX)

[1] Estimated based on preproduction measurements , Source: Spec Web Site & Newsletter

The Importance of Threading

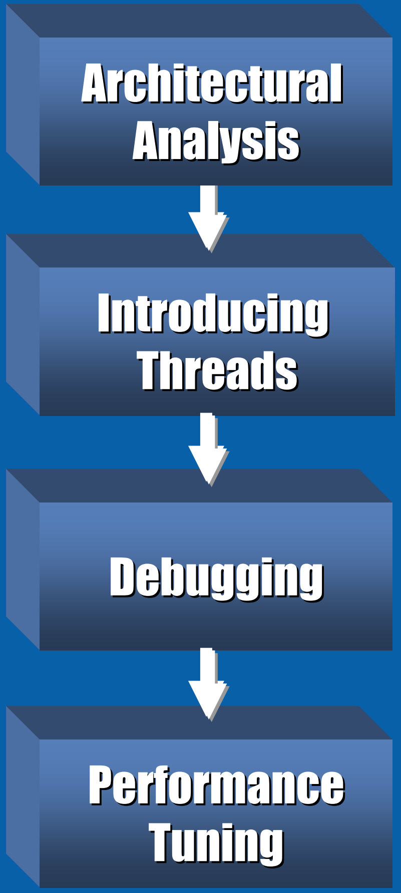
- Do Nothing: Benefits Still Visible
  - Operating systems ready for multi-processing
  - Background tasks benefit from more compute resources
- Parallelize: Unlock the Potential
  - Native threads
  - Threaded libraries
  - Compiler generated threads

Multiple cores and Parallel Programming



- No change in fundamental programming model
- Synchronization and communication costs greatly reduced
  - Optimization choices may be different
  - Makes it practical to parallelize more programs

Threading for Multi-Core



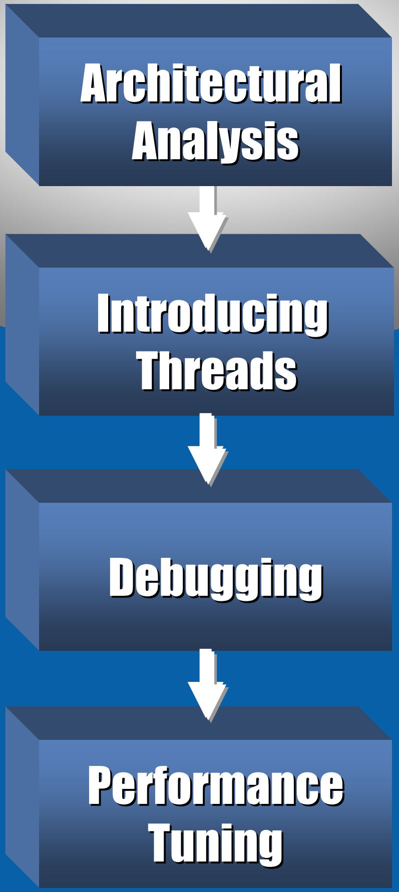
Architectural Analysis

Introducing Threads

Debugging

Performance Tuning

Threading for Multi-Core


Architectural Analysis

Introducing Threads

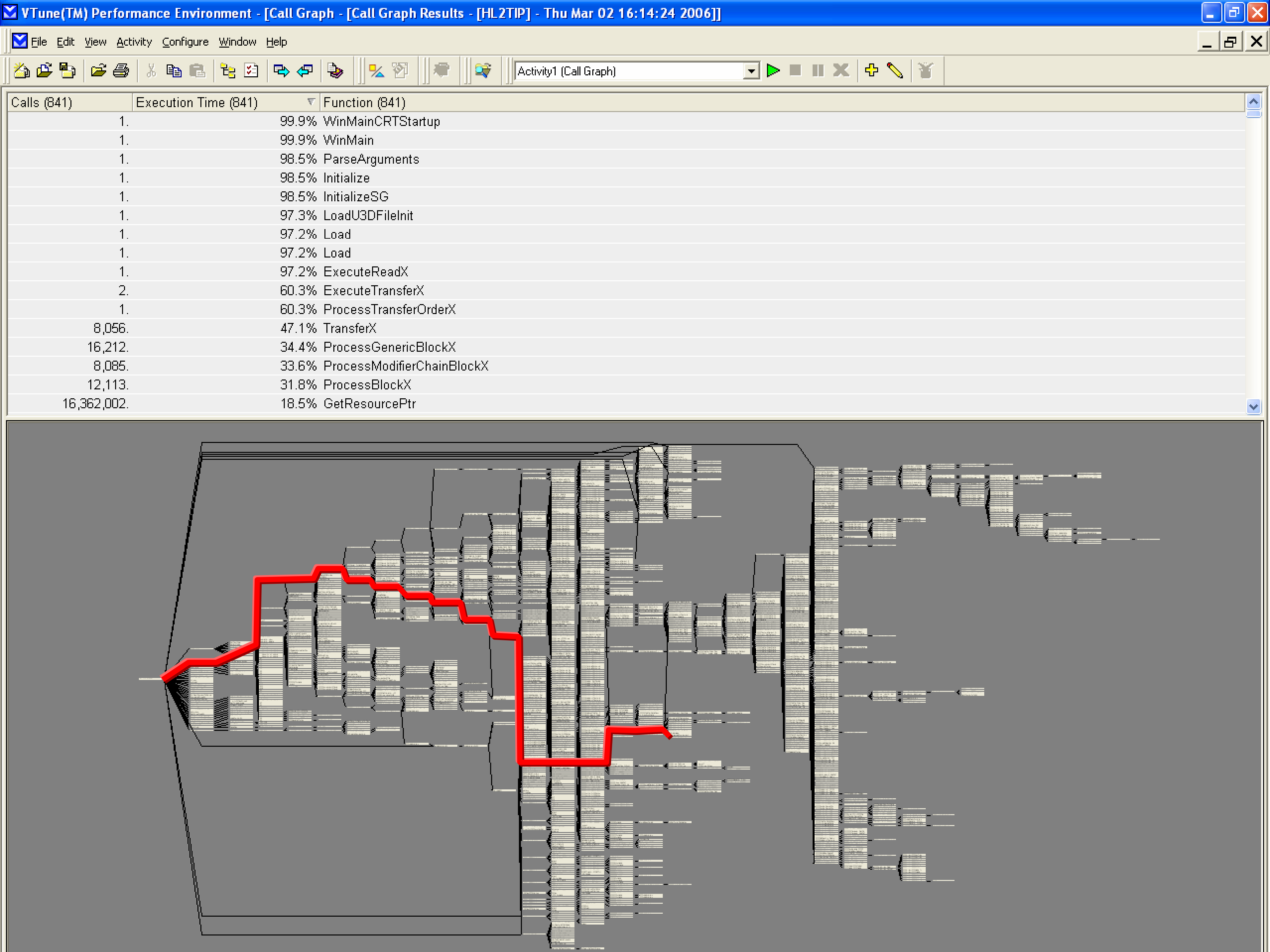
Debugging

Performance Tuning

Intel® VTune™ Performance Analyzer

Call Graph

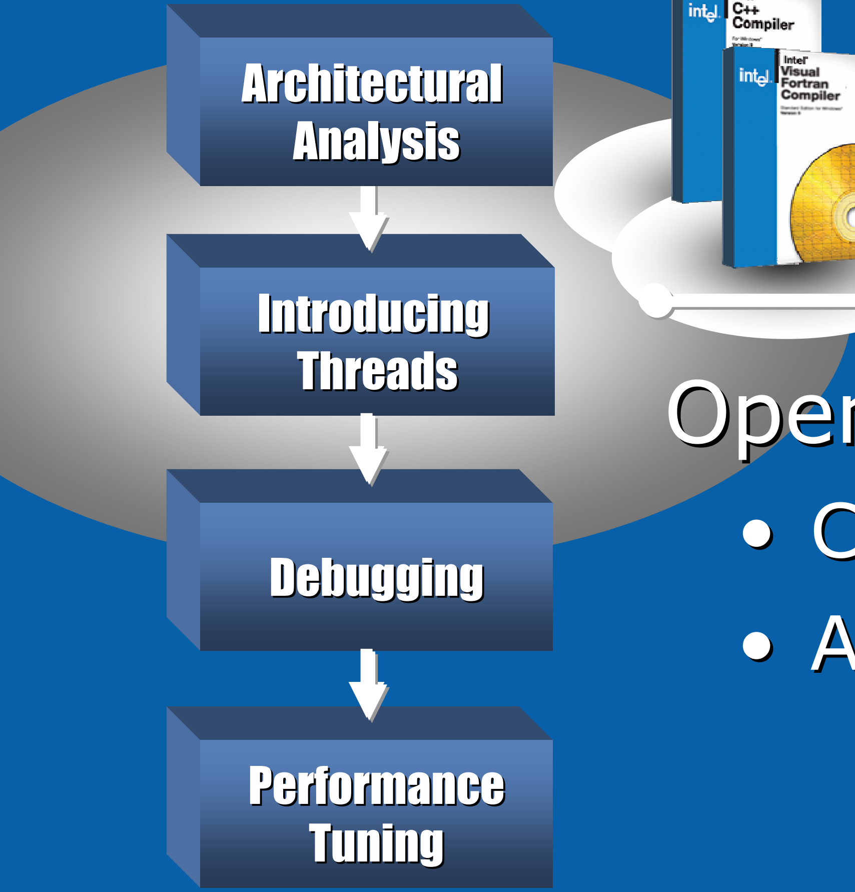
- Functional Structure
- Execution Times
- Counts



VTune(TM) Performance Environment - [Call Graph - [Call Graph Results - [HL2TIP] - Thu Mar 02 16:14:24 2006]]
File
Edit
View
Activity
Configure
Window
Help
Activity1 (Call Graph)

Calls (841)	Execution Time (841)	Function (841)
1.	99.9%	WinMainCRTStartup
1.	99.9%	WinMain
1.	98.5%	ParseArguments
1.	98.5%	Initialize
1.	98.5%	InitializeSG
1.	97.3%	LoadU3DFileInit
1.	97.2%	Load
1.	97.2%	Load
1.	97.2%	ExecuteReadX
2.	60.3%	ExecuteTransferX
1.	60.3%	ProcessTransferOrderX
8,056.	47.1%	TransferX
16,212.	34.4%	ProcessGenericBlockX
8,085.	33.6%	ProcessModifierChainBlockX
12,113.	31.8%	ProcessBlockX
16,362,002.	18.5%	GetResourcePtr

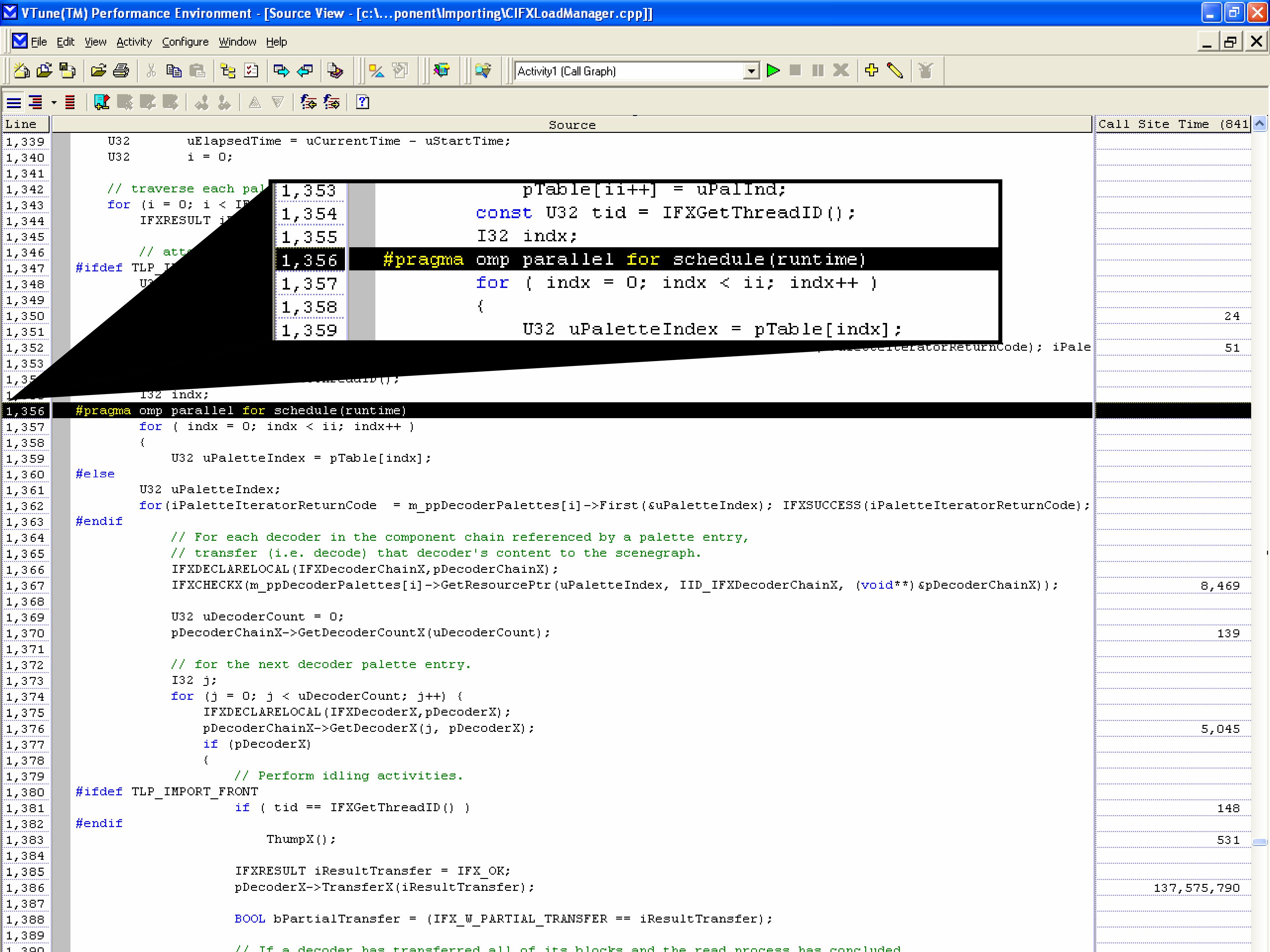
Threading for Multi-Core



Intel® Compilers

OpenMP Loop Construct

- Creates one thread per core
- Assigns iterations to threads

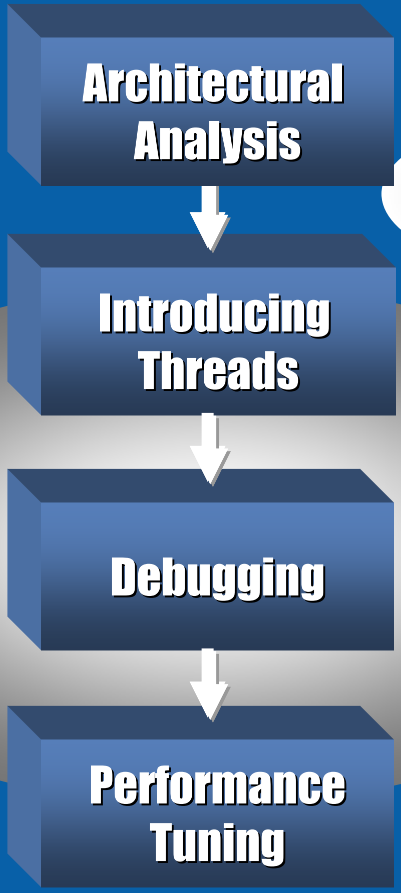


VTune(TM) Performance Environment - [Source View - [c:\...ponent\Importing\CIFXLoadManager.cpp]]
File Edit View Activity Configure Window Help
Activity1 (Call Graph)

```
1,353              pTable[ii++] = uPalInd;
1,354          const U32 tid = IFXGetThreadID();
1,355          I32 indx;
1,356    #pragma omp parallel for schedule(runtime)
1,357          for ( indx = 0; indx < ii; indx++ )
1,358          {
1,359              U32 uPaletteIndex = pTable[indx];
```

```
Line   Source                                                                                                          Call Site Time (841
1,339      U32         uElapsedTime = uCurrentTime - uStartTime;
1,340      U32         i = 0;
1,341
1,342      // traverse each pal
1,343      for (i = 0; i < IF
1,344          IFXRESULT i
1,345
1,346          // att
1,347  #ifdef TLP_I
1,348          U3
1,349
1,350                                                                                                                            24
1,351
1,352                                                                                   atorReturnCode); iPale                  51
1,353
1,35
1,
1,            I32 indx;
1,356  #pragma omp parallel for schedule(runtime)
1,357          for ( indx = 0; indx < ii; indx++ )
1,358          {
1,359              U32 uPaletteIndex = pTable[indx];
1,360  #else
1,361          U32 uPaletteIndex;
1,362          for(iPaletteIteratorReturnCode  = m_ppDecoderPalettes[i]->First(&uPaletteIndex); IFXSUCCESS(iPaletteIteratorReturnCode);
1,363  #endif
1,364              // For each decoder in the component chain referenced by a palette entry,
1,365              // transfer (i.e. decode) that decoder's content to the scenegraph.
1,366              IFXDECLARELOCAL(IFXDecoderChainX,pDecoderChainX);
1,367              IFXCHECKX(m_ppDecoderPalettes[i]->GetResourcePtr(uPaletteIndex, IID_IFXDecoderChainX, (void**)&pDecoderChainX));      8,469
1,368
1,369              U32 uDecoderCount = 0;
1,370              pDecoderChainX->GetDecoderCountX(uDecoderCount);                                                                   139
1,371
1,372              // for the next decoder palette entry.
1,373              I32 j;
1,374              for (j = 0; j < uDecoderCount; j++) {
1,375                  IFXDECLARELOCAL(IFXDecoderX,pDecoderX);
1,376                  pDecoderChainX->GetDecoderX(j, pDecoderX);                                                                   5,045
1,377                  if (pDecoderX)
1,378                  {
1,379                      // Perform idling activities.
1,380  #ifdef TLP_IMPORT_FRONT
1,381                      if ( tid == IFXGetThreadID() )                                                                           148
1,382  #endif
1,383                          ThumpX();                                                                                            531
1,384
1,385                      IFXRESULT iResultTransfer = IFX_OK;
1,386                      pDecoderX->TransferX(iResultTransfer);                                                           137,575,790
1,387
1,388                      BOOL bPartialTransfer = (IFX_W_PARTIAL_TRANSFER == iResultTransfer);
1,389
1,390                      // If a decoder has transferred all of its blocks and the read process has concluded
```

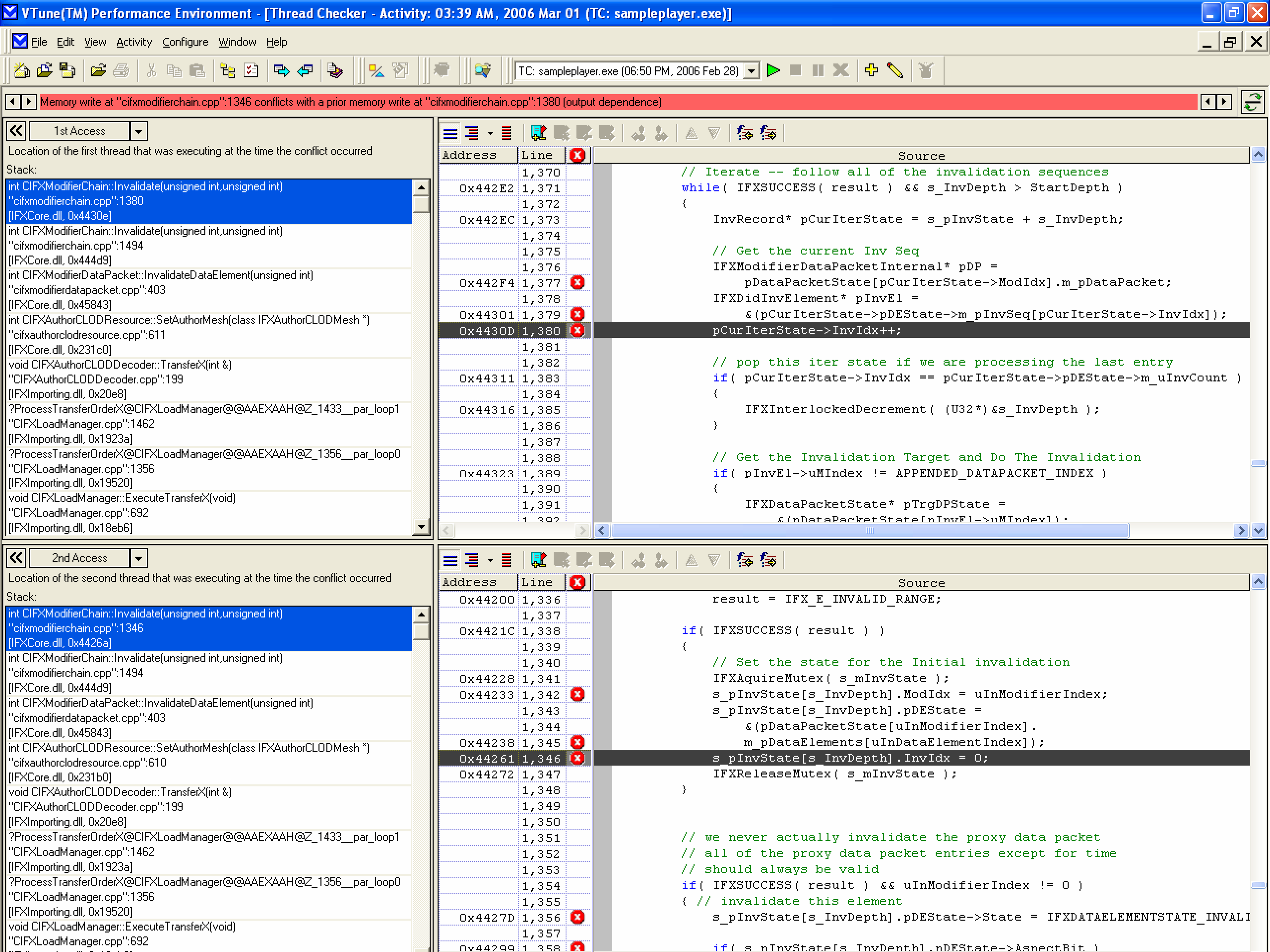
Threading for Multi-Core



Intel® Thread Checker

Thread Safety Issues

- Data Races
- Deadlocks



VTune(TM) Performance Environment - [Thread Checker - Activity: 03:39 AM, 2006 Mar 01 (TC: sampleplayer.exe)]

TC: sampleplayer.exe (06:50 PM, 2006 Feb 28)

Memory write at "cifxmodifierchain.cpp":1346 conflicts with a prior memory write at "cifxmodifierchain.cpp":1380 (output dependence)

1st Access

Location of the first thread that was executing at the time the conflict occurred

Stack:

```
int CIFXModifierChain::Invalidate(unsigned int,unsigned int)
"cifxmodifierchain.cpp":1380
[IFXCore.dll, 0x4430e]
int CIFXModifierChain::Invalidate(unsigned int,unsigned int)
"cifxmodifierchain.cpp":1494
[IFXCore.dll, 0x444d9]
int CIFXModifierDataPacket::InvalidateDataElement(unsigned int)
"cifxmodifierdatapacket.cpp":403
[IFXCore.dll, 0x45843]
int CIFXAuthorCLODResource::SetAuthorMesh(class IFXAuthorCLODMesh *)
"cifxauthorclodresource.cpp":611
[IFXCore.dll, 0x231c0]
void CIFXAuthorCLODDecoder::TransferX(int &)
"CIFXAuthorCLODDecoder.cpp":199
[IFXImporting.dll, 0x20e8]
?ProcessTransferOrderX@CIFXLoadManager@@AAEXAAH@Z_1433__par_loop1
"CIFXLoadManager.cpp":1462
[IFXImporting.dll, 0x1923a]
?ProcessTransferOrderX@CIFXLoadManager@@AAEXAAH@Z_1356__par_loop0
"CIFXLoadManager.cpp":1356
[IFXImporting.dll, 0x19520]
void CIFXLoadManager::ExecuteTransferX(void)
"CIFXLoadManager.cpp":692
[IFXImporting.dll, 0x18eb6]
```

```
Address  Line   ☒  Source
         1,370        // Iterate -- follow all of the invalidation sequences
0x442E2  1,371        while( IFXSUCCESS( result ) && s_InvDepth > StartDepth )
         1,372        {
0x442EC  1,373            InvRecord* pCurIterState = s_pInvState + s_InvDepth;
         1,374
         1,375            // Get the current Inv Seq
         1,376            IFXModifierDataPacketInternal* pDP =
0x442F4  1,377  ☒             pDataPacketState[pCurIterState->ModIdx].m_pDataPacket;
         1,378            IFXDidInvElement* pInvEl =
0x44301  1,379  ☒             &(pCurIterState->pDEState->m_pInvSeq[pCurIterState->InvIdx]);
0x4430D  1,380  ☒         pCurIterState->InvIdx++;
         1,381
         1,382            // pop this iter state if we are processing the last entry
0x44311  1,383            if( pCurIterState->InvIdx == pCurIterState->pDEState->m_uInvCount )
         1,384            {
0x44316  1,385                IFXInterlockedDecrement( (U32*)&s_InvDepth );
         1,386            }
         1,387
         1,388            // Get the Invalidation Target and Do The Invalidation
0x44323  1,389            if( pInvEl->uMIndex != APPENDED_DATAPACKET_INDEX )
         1,390            {
         1,391                IFXDataPacketState* pTrgDPState =
         1,392                    &(pDataPacketState[pInvEl->uMIndex]).
```

2nd Access

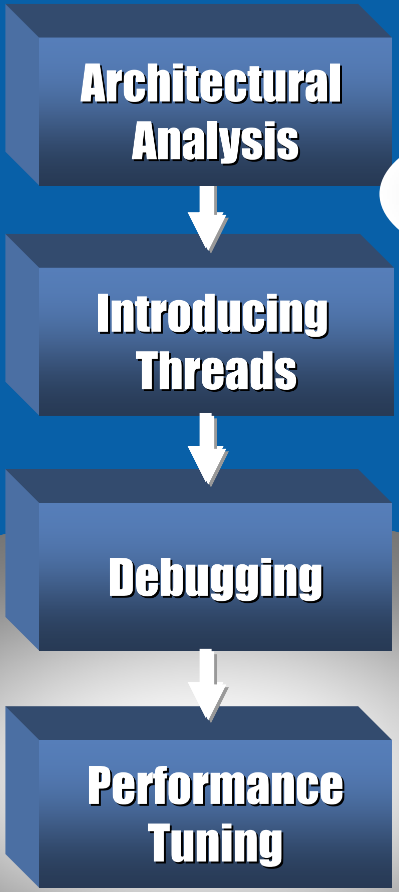
Location of the second thread that was executing at the time the conflict occurred

Stack:

```
int CIFXModifierChain::Invalidate(unsigned int,unsigned int)
"cifxmodifierchain.cpp":1346
[IFXCore.dll, 0x4426a]
int CIFXModifierChain::Invalidate(unsigned int,unsigned int)
"cifxmodifierchain.cpp":1494
[IFXCore.dll, 0x444d9]
int CIFXModifierDataPacket::InvalidateDataElement(unsigned int)
"cifxmodifierdatapacket.cpp":403
[IFXCore.dll, 0x45843]
int CIFXAuthorCLODResource::SetAuthorMesh(class IFXAuthorCLODMesh *)
"cifxauthorclodresource.cpp":610
[IFXCore.dll, 0x231b0]
void CIFXAuthorCLODDecoder::TransferX(int &)
"CIFXAuthorCLODDecoder.cpp":199
[IFXImporting.dll, 0x20e8]
?ProcessTransferOrderX@CIFXLoadManager@@AAEXAAH@Z_1433__par_loop1
"CIFXLoadManager.cpp":1462
[IFXImporting.dll, 0x1923a]
?ProcessTransferOrderX@CIFXLoadManager@@AAEXAAH@Z_1356__par_loop0
"CIFXLoadManager.cpp":1356
[IFXImporting.dll, 0x19520]
void CIFXLoadManager::ExecuteTransferX(void)
"CIFXLoadManager.cpp":692
```

```
Address  Line   ☒  Source
0x44200  1,336            result = IFX_E_INVALID_RANGE;
         1,337
0x4421C  1,338        if( IFXSUCCESS( result ) )
         1,339        {
         1,340            // Set the state for the Initial invalidation
0x44228  1,341            IFXAquireMutex( s_mInvState );
0x44233  1,342  ☒         s_pInvState[s_InvDepth].ModIdx = uInModifierIndex;
         1,343            s_pInvState[s_InvDepth].pDEState =
         1,344                &(pDataPacketState[uInModifierIndex].
0x44238  1,345  ☒             m_pDataElements[uInDataElementIndex]);
0x44261  1,346  ☒         s_pInvState[s_InvDepth].InvIdx = 0;
0x44272  1,347            IFXReleaseMutex( s_mInvState );
         1,348        }
         1,349
         1,350
         1,351        // we never actually invalidate the proxy data packet
         1,352        // all of the proxy data packet entries except for time
         1,353        // should always be valid
         1,354        if( IFXSUCCESS( result ) && uInModifierIndex != 0 )
         1,355        { // invalidate this element
0x4427D  1,356  ☒         s_pInvState[s_InvDepth].pDEState->State = IFXDATAELEMENTSTATE_INVALI
         1,357
0x44299  1,358  ☒         if( s_pInvState[s_InvDepth].pDEState->AspectBit )
```

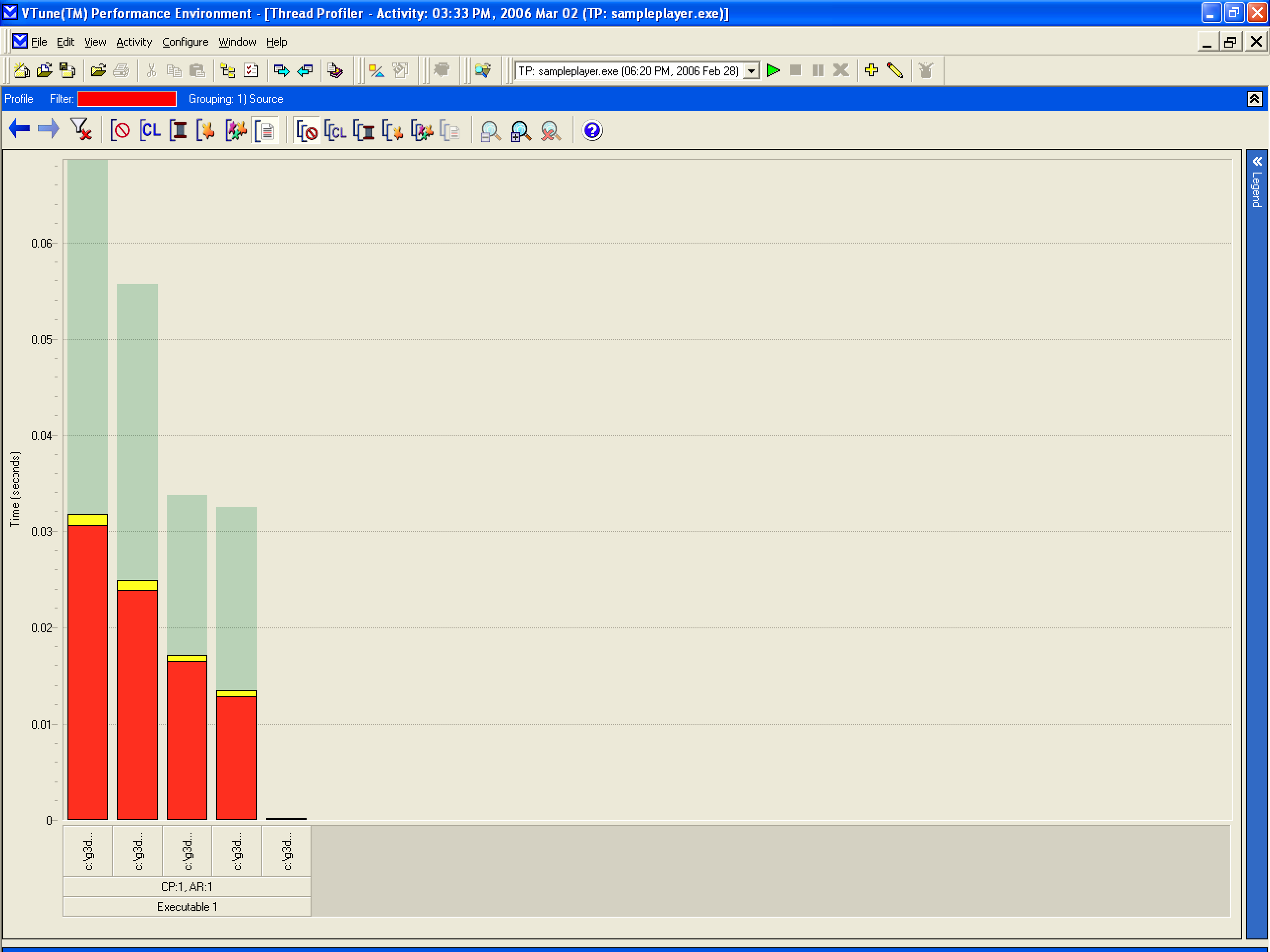
Threading for Multi-Core



Intel® Thread Profiler

Find Contended Locks

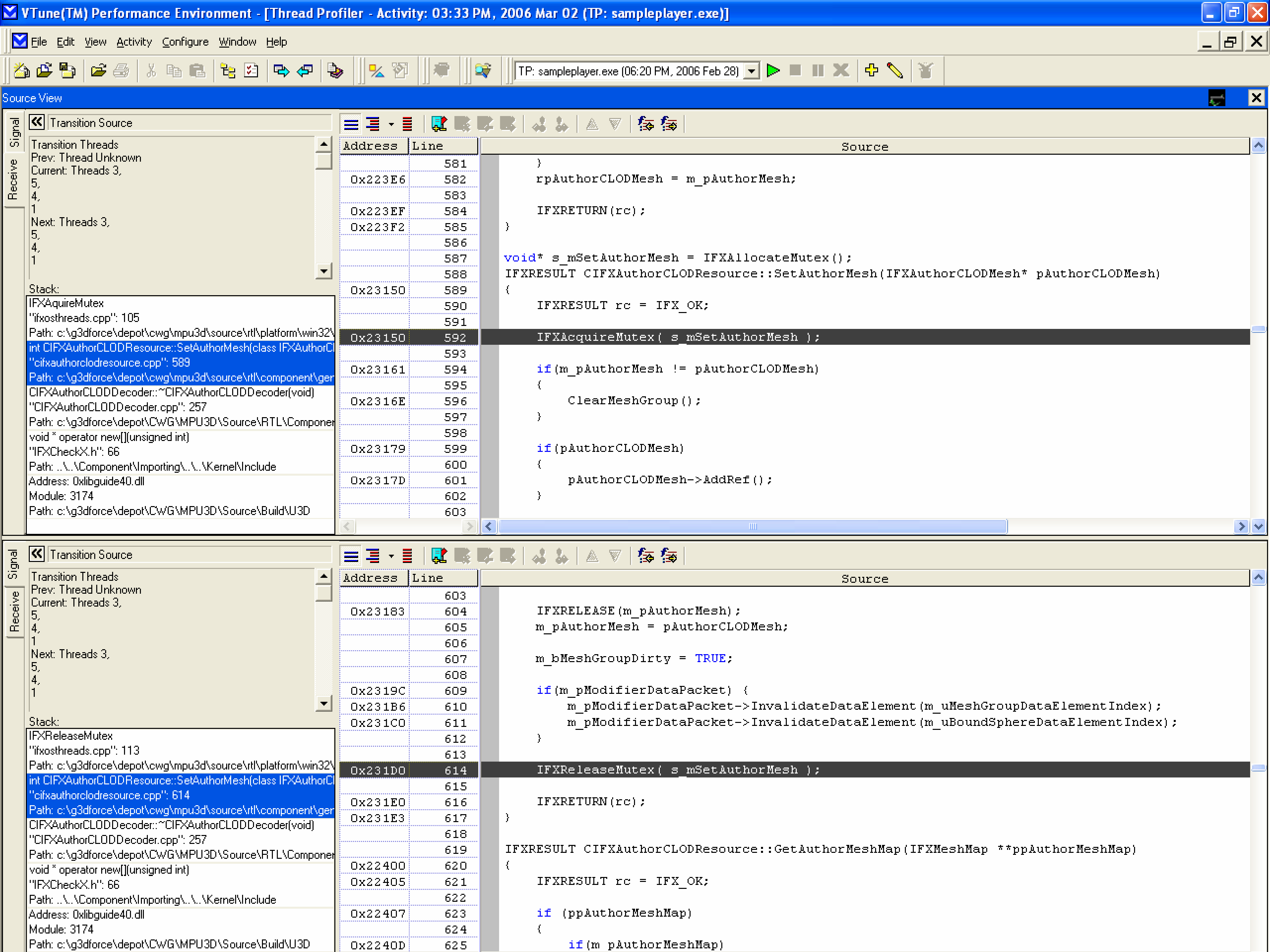
- Most Overhead
- Largest Reduction in Parallelism



VTune(TM) Performance Environment - [Thread Profiler - Activity: 03:33 PM, 2006 Mar 02 (TP: sampleplayer.exe)]

TP: sampleplayer.exe (06:20 PM, 2006 Feb 28)

Profile Filter: Grouping: 1) Source



VTune(TM) Performance Environment - [Thread Profiler - Activity: 03:33 PM, 2006 Mar 02 (TP: sampleplayer.exe)]
File Edit View Activity Configure Window Help
TP: sampleplayer.exe (06:20 PM, 2006 Feb 28)
Source View
Signal
Receive
Transition Source

```
Transition Threads
Prev: Thread Unknown
Current: Threads 3,
5,
4,
1
Next: Threads 3,
5,
4,
1

Stack:
IFXAquireMutex
"ifxosthreads.cpp": 105
Path: c:\g3dforce\depot\cwg\mpu3d\source\rtl\platform\win32\
int CIFXAuthorCLODResource::SetAuthorMesh(class IFXAuthorCl
"cifxauthorclodresource.cpp": 589
Path: c:\g3dforce\depot\cwg\mpu3d\source\rtl\component\ger
CIFXAuthorCLODDecoder::~CIFXAuthorCLODDecoder(void)
"CIFXAuthorCLODDecoder.cpp": 257
Path: c:\g3dforce\depot\CWG\MPU3D\Source\RTL\Componer
void * operator new[](unsigned int)
"IFXCheckX.h": 66
Path: ..\..\Component\Importing\..\..\Kernel\Include
Address: 0xlibguide40.dll
Module: 3174
Path: c:\g3dforce\depot\CWG\MPU3D\Source\Build\U3D
```

```
Address  Line   Source
         581        }
0x223E6  582        rpAuthorCLODMesh = m_pAuthorMesh;
         583
0x223EF  584        IFXRETURN(rc);
0x223F2  585    }
         586
         587    void* s_mSetAuthorMesh = IFXAllocateMutex();
         588    IFXRESULT CIFXAuthorCLODResource::SetAuthorMesh(IFXAuthorCLODMesh* pAuthorCLODMesh)
0x23150  589    {
         590        IFXRESULT rc = IFX_OK;
         591
0x23150  592        IFXAcquireMutex( s_mSetAuthorMesh );
         593
0x23161  594        if(m_pAuthorMesh != pAuthorCLODMesh)
         595        {
0x2316E  596            ClearMeshGroup();
         597        }
         598
0x23179  599        if(pAuthorCLODMesh)
         600        {
0x2317D  601            pAuthorCLODMesh->AddRef();
         602        }
         603
```

```
Transition Threads
Prev: Thread Unknown
Current: Threads 3,
5,
4,
1
Next: Threads 3,
5,
4,
1

Stack:
IFXReleaseMutex
"ifxosthreads.cpp": 113
Path: c:\g3dforce\depot\cwg\mpu3d\source\rtl\platform\win32\
int CIFXAuthorCLODResource::SetAuthorMesh(class IFXAuthorCl
"cifxauthorclodresource.cpp": 614
Path: c:\g3dforce\depot\cwg\mpu3d\source\rtl\component\ger
CIFXAuthorCLODDecoder::~CIFXAuthorCLODDecoder(void)
"CIFXAuthorCLODDecoder.cpp": 257
Path: c:\g3dforce\depot\CWG\MPU3D\Source\RTL\Componer
void * operator new[](unsigned int)
"IFXCheckX.h": 66
Path: ..\..\Component\Importing\..\..\Kernel\Include
Address: 0xlibguide40.dll
Module: 3174
Path: c:\g3dforce\depot\CWG\MPU3D\Source\Build\U3D
```

```
Address  Line   Source
         603
0x23183  604        IFXRELEASE(m_pAuthorMesh);
         605        m_pAuthorMesh = pAuthorCLODMesh;
         606
         607        m_bMeshGroupDirty = TRUE;
         608
0x2319C  609        if(m_pModifierDataPacket) {
0x231B6  610            m_pModifierDataPacket->InvalidateDataElement(m_uMeshGroupDataElementIndex);
0x231C0  611            m_pModifierDataPacket->InvalidateDataElement(m_uBoundSphereDataElementIndex);
         612        }
         613
0x231D0  614        IFXReleaseMutex( s_mSetAuthorMesh );
         615
0x231E0  616        IFXRETURN(rc);
0x231E3  617    }
         618
         619    IFXRESULT CIFXAuthorCLODResource::GetAuthorMeshMap(IFXMeshMap **ppAuthorMeshMap)
0x22400  620    {
0x22405  621        IFXRESULT rc = IFX_OK;
         622
0x22407  623        if (ppAuthorMeshMap)
         624        {
0x2240D  625            if(m_pAuthorMeshMap)
```

Growing Momentum For Software Parallelization

Activision (Ravensoft)
Adobe
Algorithmics
Alias
Autodesk
Business Objects
Cakewalk
CodecPeople
Computer Associates
Corel (WordPerfect)
Cyberlink
Discreet
IBM
id Software
Landmark
Macromedia
Mainconcept
Maxon
mental images
Microsoft (Office Suite)
Midway
MSC
Novell SUSE
Oracle
Pegasus
Pinnacle
Pixar (Renderman)
Paradigm
PTC
Red Hat
SAP
SAS
Siebel CRM
Signet
Skype
SLB
SnapStream
Sonic (Roxio)
Sony
Steinberg
SunGard
Sybase
Symantec
Thomson
THQ
Ubisoft
UGS
Valve
Yahoo (Musicmatch)

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

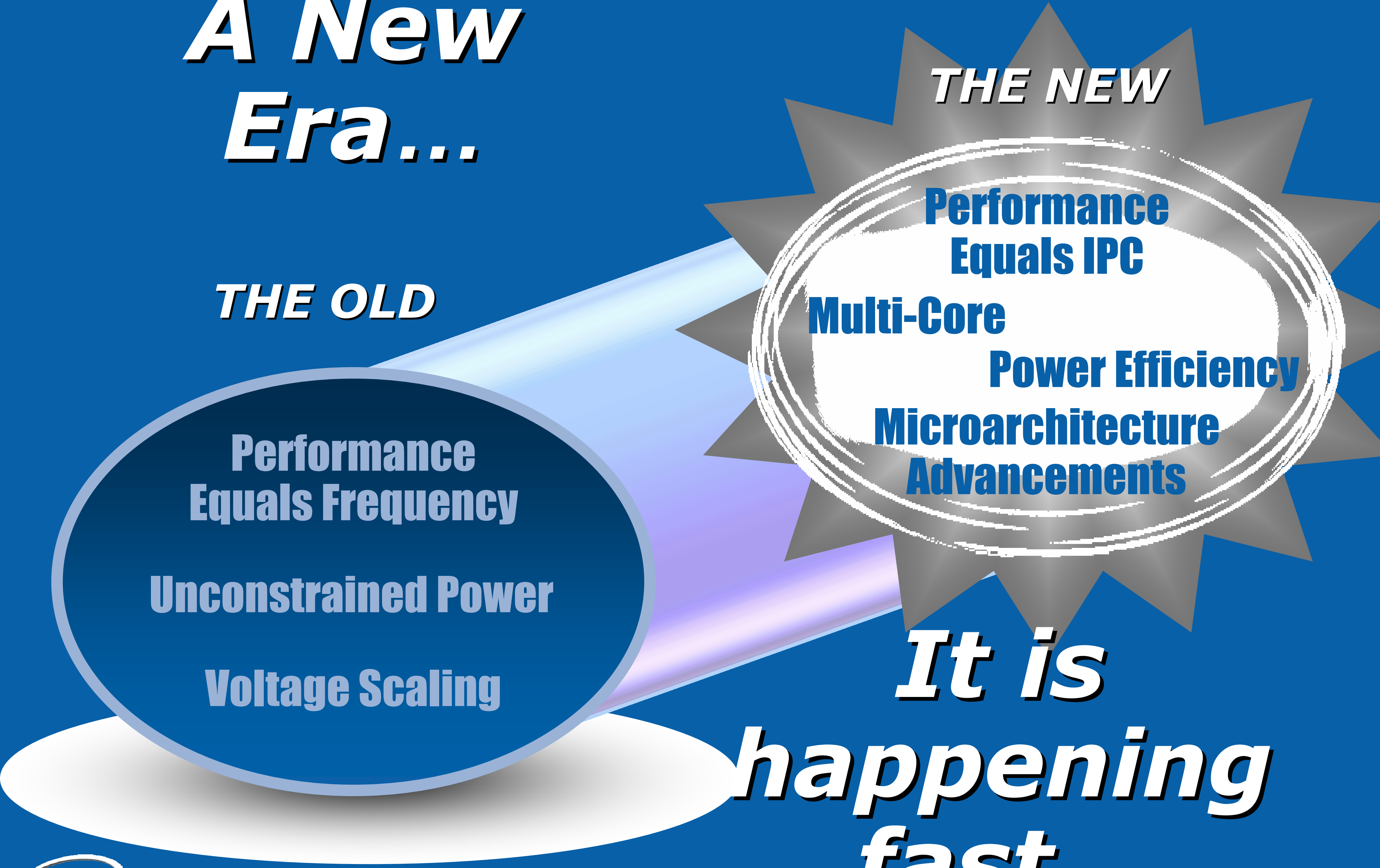
Agenda

Major Technological Change

Software Response

Parallel Software 2.0

A New Era…



THE OLD

Performance Equals Frequency

Unconstrained Power

Voltage Scaling

THE NEW

Performance Equals IPC

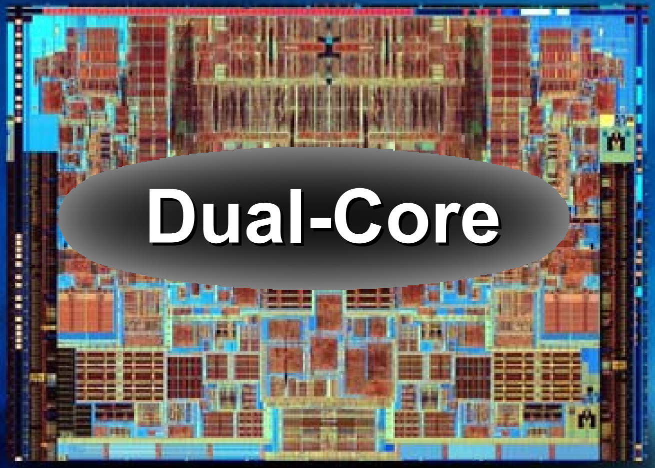
Multi-Core

Power Efficiency

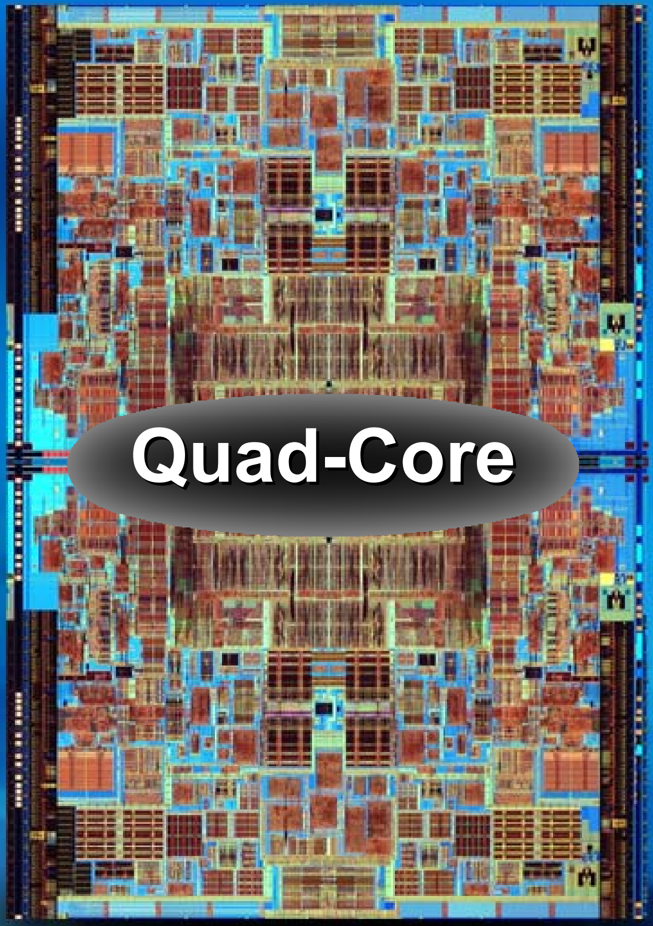
Microarchitecture Advancements

It is happening fast…

Multi-Core Trajectory



2006



2007

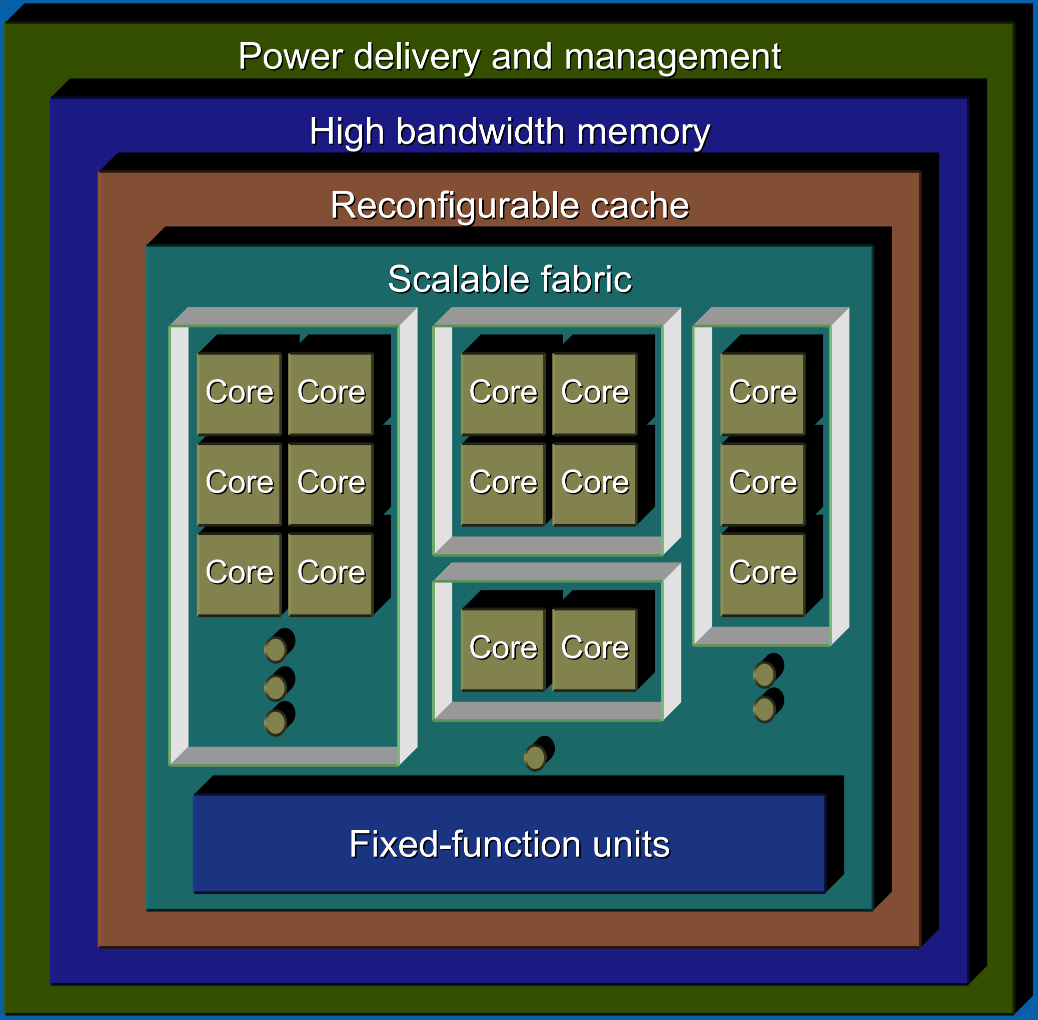
Future Architecture
Many More Cores

Parallel extension of IA

- Homogeneous array of cores
- Fixed-function units
- Coarse- and fine-grained data- and thread-level parallelism
- Global coherency hardware

Partitioned array

- Application domains
- Isolated communication traffic
- Fault tolerance



Parallel Software 2.0

- Ease of programming
  - Programming language, compiler, tools
- Ubiquitous
  - Consumer/wireless vs HPC/database
  - Home vs nuclear labs
  - More legacy applications
- Explosion of cores
  - 2X cores every 18 months
  - Scalable software
- Reliability
- User experience
  - vs. just raw performance
- Education
  - Mass vs elite

Parallel Software 2.0

The Beginning of a New Era