



# Parallel Software 2.0

**Wei Li**  
Senior Principal Engineer  
Intel Corporation

# Agenda

Major Technological Change

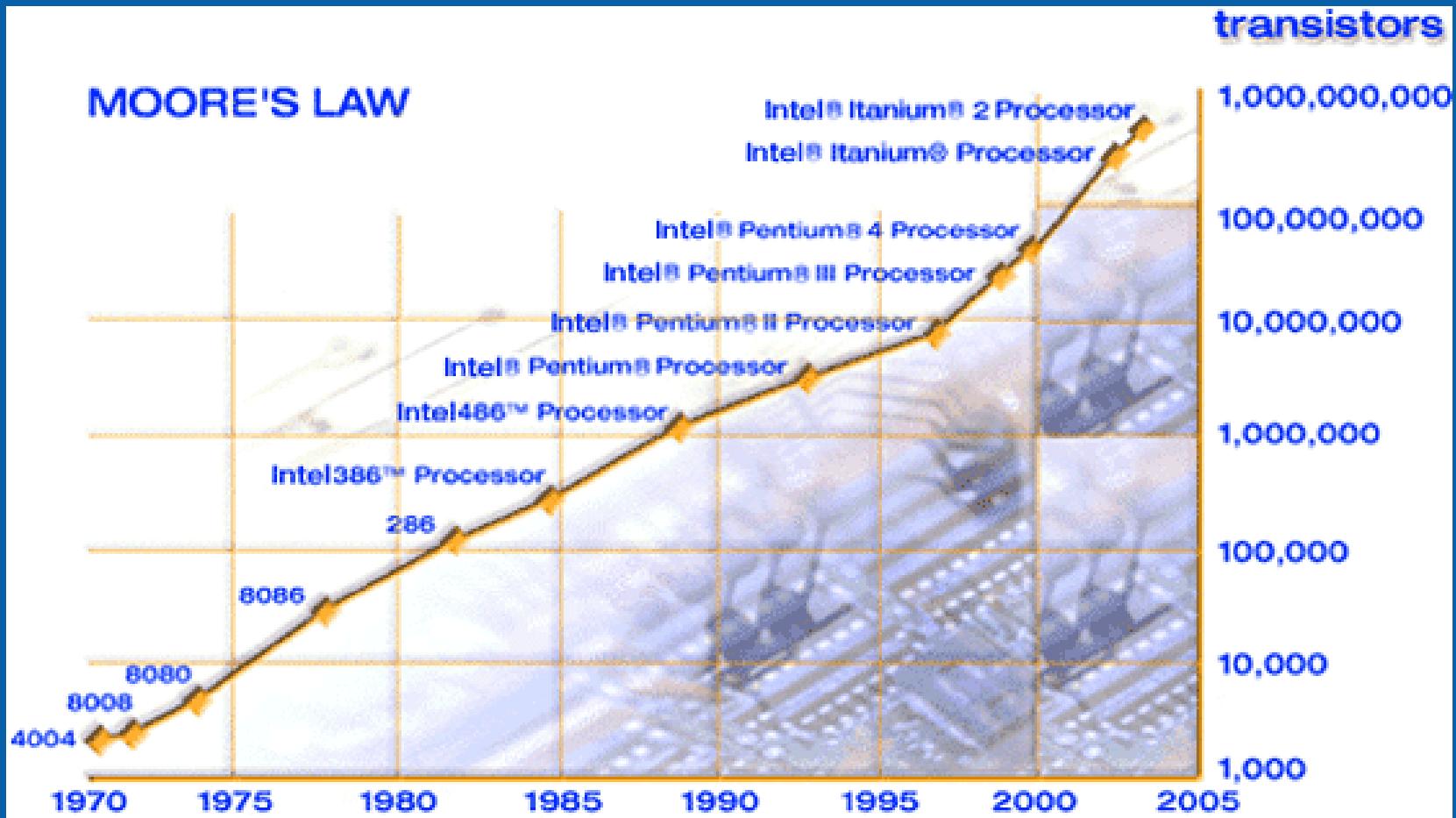
Software Response

Parallel Software 2.0

# Quiz

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

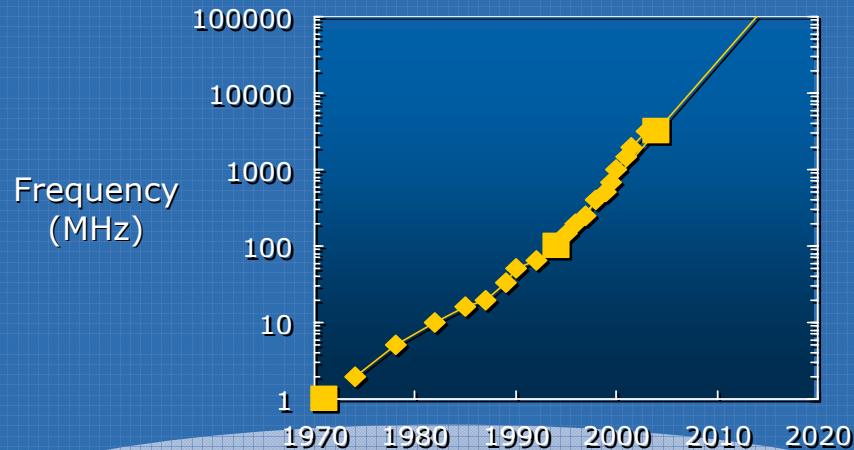
1. Frequency
2. Performance
3. Transistors
4. Transistor Density



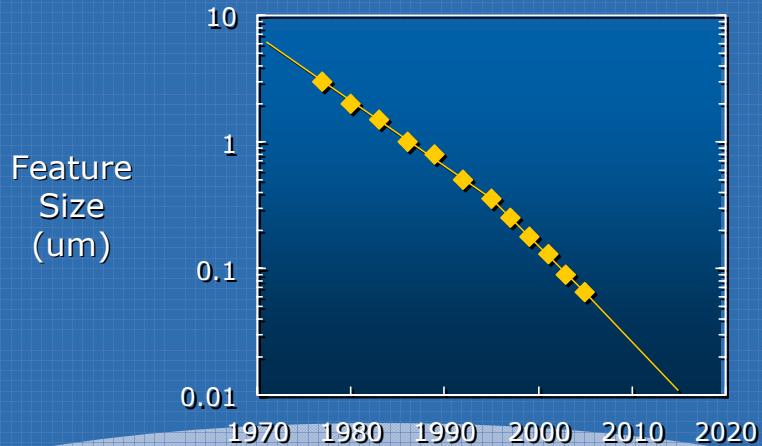
*from [www.intel.com](http://www.intel.com)*

# Historical Driving Forces

## Increased Performance via Increased Frequency



## Shrinking Geometry



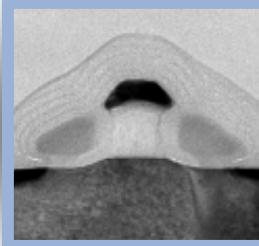
1946

20 Numbers  
in Main Memory



1971

I4004 Processor  
2300 Transistors

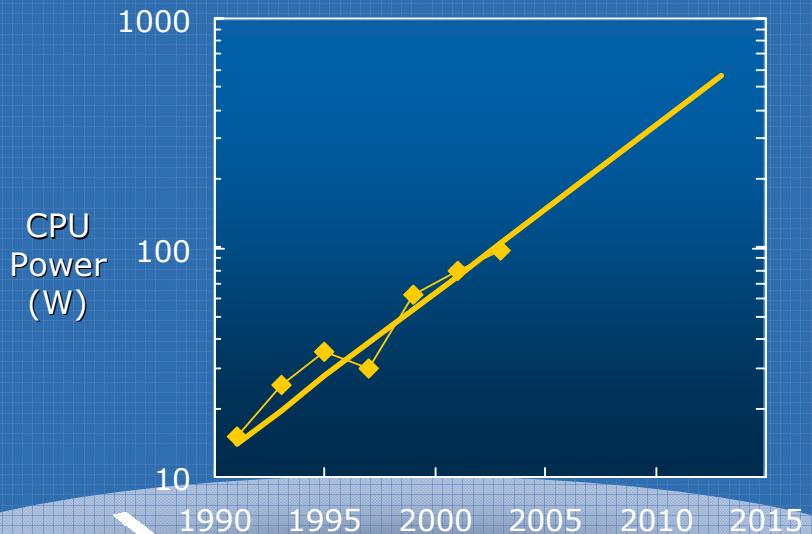


2005

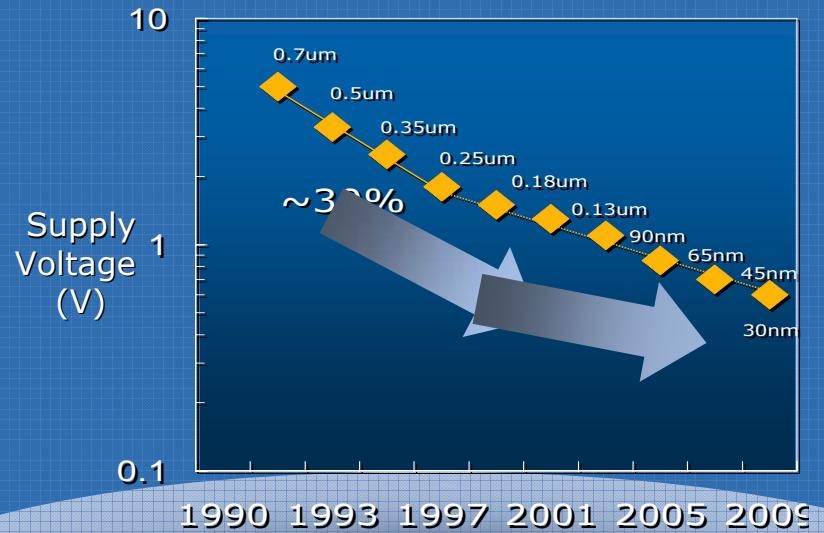
65nm  
1B+ Transistors

# The Challenges

## Power Limitations



## Diminishing Voltage Scaling



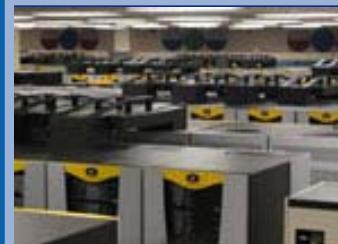
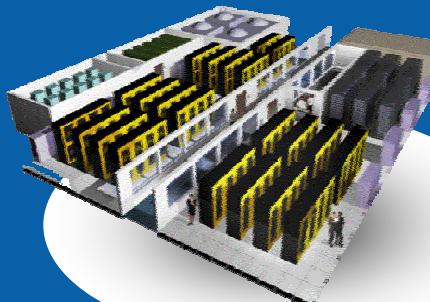
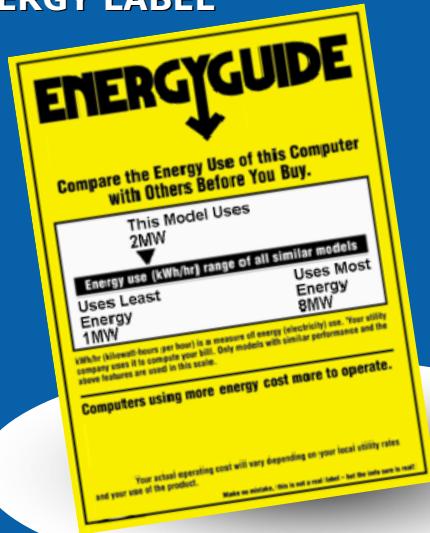
**Power = Capacitance x Voltage<sup>2</sup> x Frequency**  
also  
**Power ~ Voltage<sup>3</sup>**

# Energy: The Next Frontier



# Energy Efficient Performance – High End

DATACENTER  
“ENERGY LABEL”



## 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



**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

OR



# The Real Challenge

Capabilities



Performance



Energy-Efficiency

# Reducing Power with Voltage Scaling

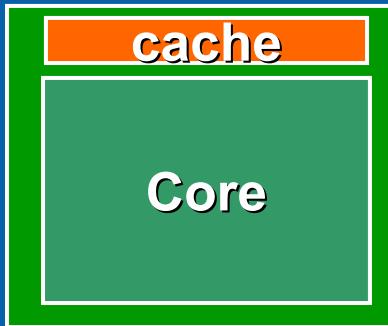
- **Power = Capacitance \* Voltage<sup>2</sup> \* Frequency**
- **Frequency ~ Voltage in region of interest**
- **Power ~ Voltage<sup>3</sup>**
- **10% reduction of voltage yields**
  - 10% reduction in frequency
  - 30% reduction in power
  - Less than 10% reduction in performance

## 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%



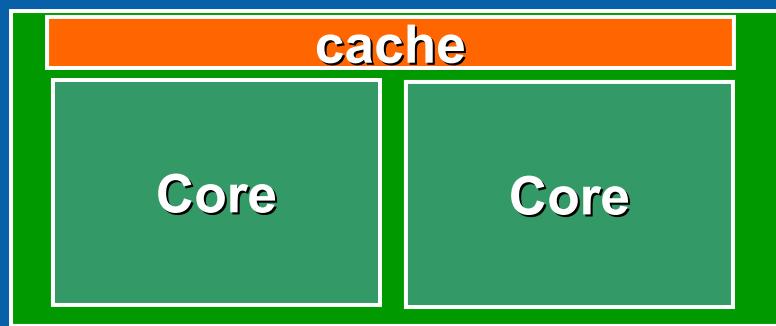
**Voltage = 1**

**Freq = 1**

**Area = 1**

**Power = 1**

**Perf = 1**



**Voltage = -15%**

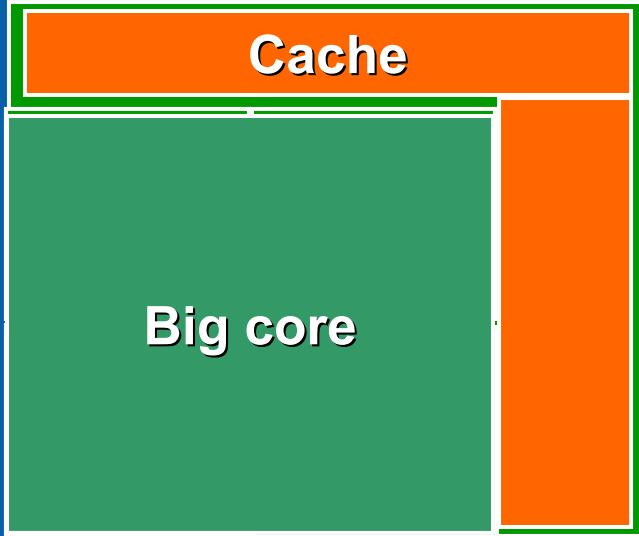
**Freq = -15%**

**Area = 2**

**Power = 1**

**Perf = ~1.8**

# Multiple cores deliver more performance per watt



Power



Power =  $\frac{1}{4}$

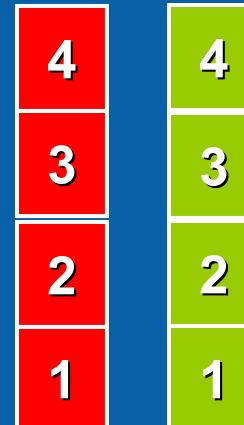
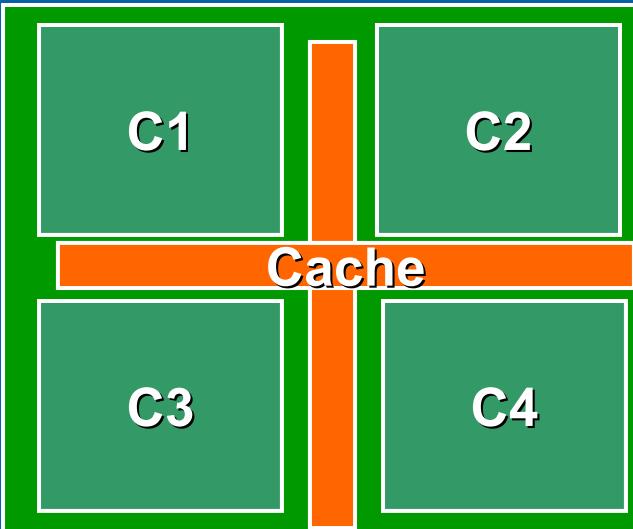
Performance



Performance = 1/2



Small  
core



Many core is more power efficient

Power ~ area

Single thread performance ~ area<sup>\*\*.5</sup>

# 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)

- Single Threaded
- Multi Threaded

Pentium    Pentium II    Pentium III    Pentium 4    Pentium D    Conroe<sup>†</sup>



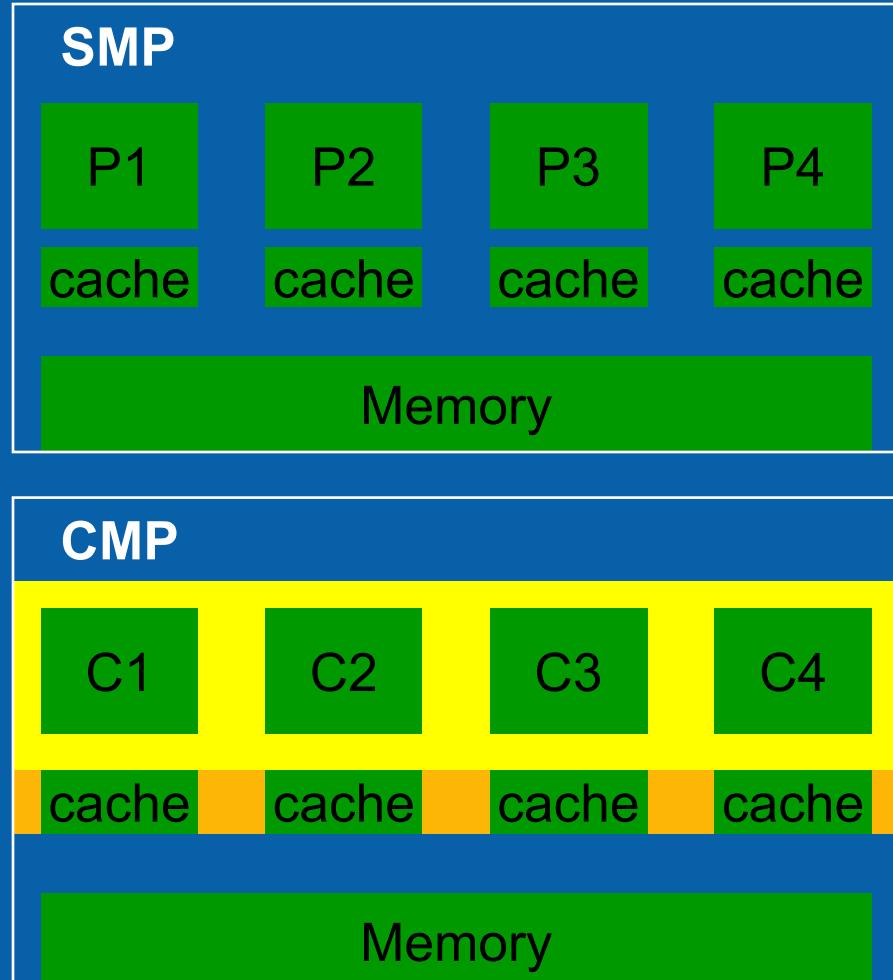
† Estimated based on preproduction measurements. Source: SpecWeb site & Newsletter



# The Importance of Threading

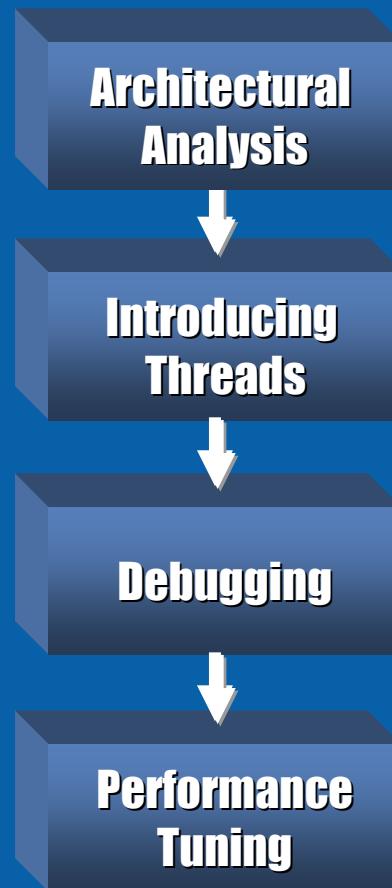
- Do Nothing: Benefits Still Visible
  - Operating systems ready for multi-processing
  - Background tasks benefit from more compute resources
- Parallelize: Unlock the Potential
  - Native threads
  - Threaded libraries
  - Compiler generated threads

# Multiple cores and Parallel Programming

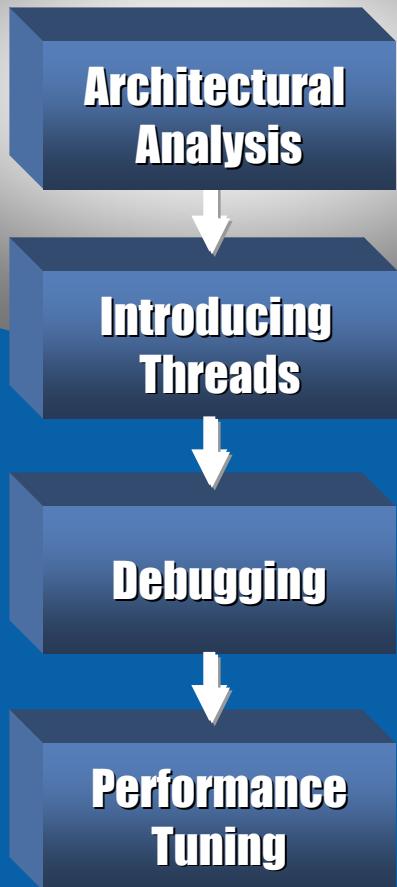


- No change in fundamental programming model
- Synchronization and communication costs greatly reduced
  - Optimization choices may be different
  - Makes it practical to parallelize more programs

# Threading for Multi-Core



# Threading for Multi-Core



Intel® VTune™  
Performance Analyzer

## Call Graph

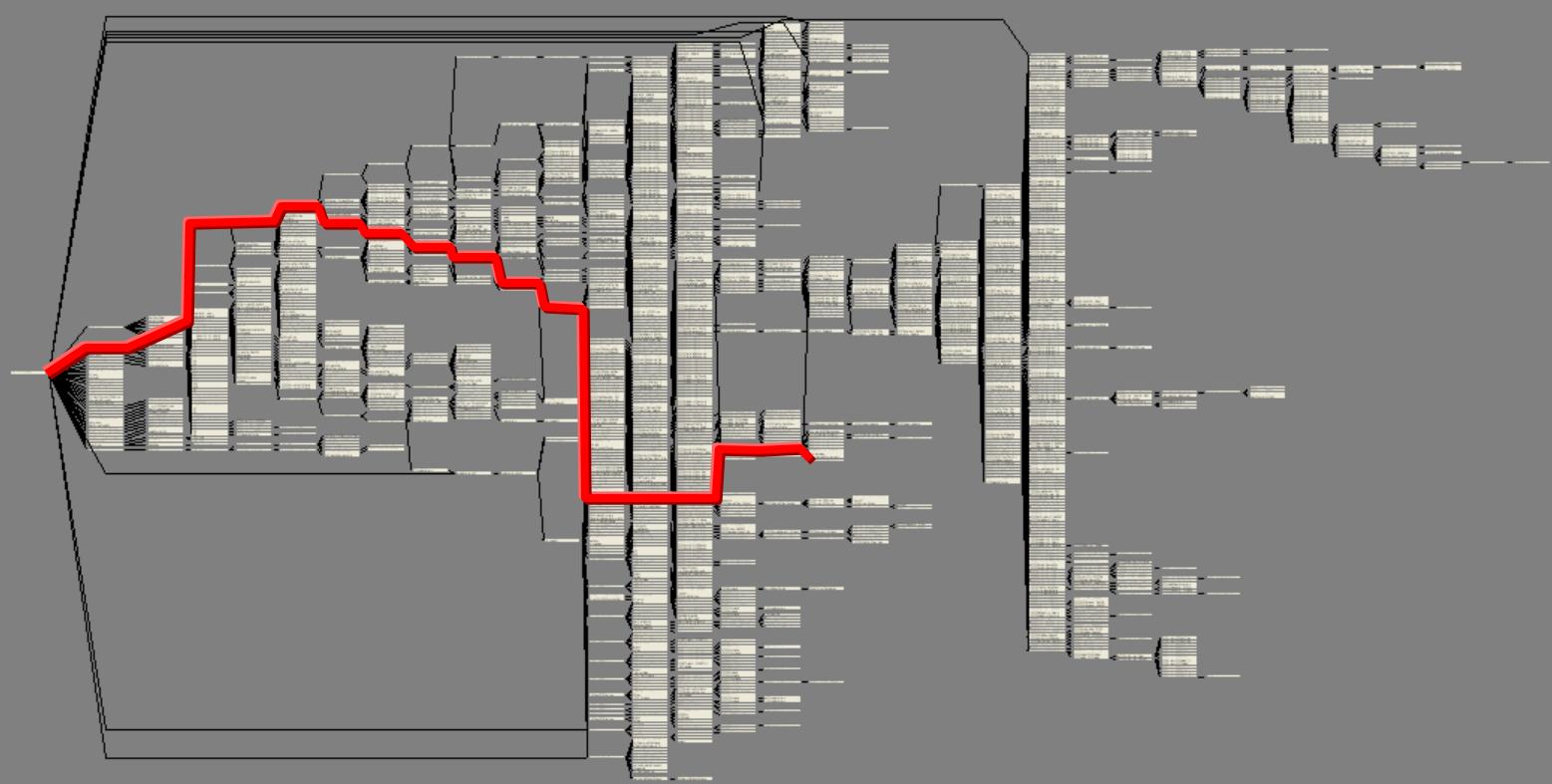
- Functional Structure
- Execution Times
- Counts



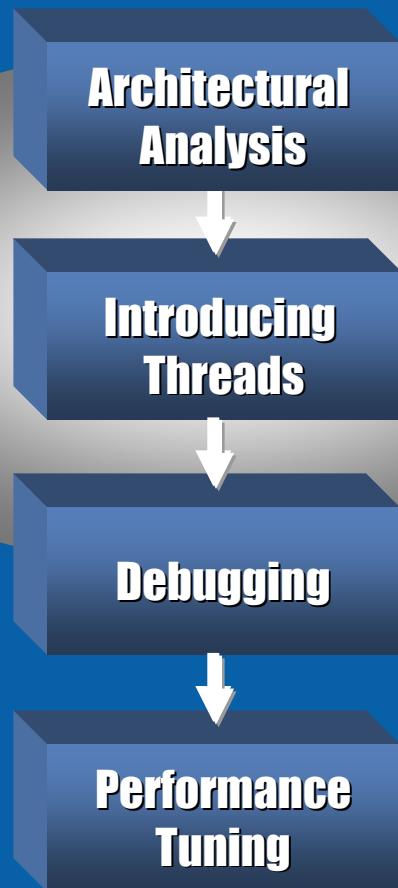
Activity1 (Call Graph)



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\IFXLoadManager.cpp]]

File Edit View Activity Configure Window Help

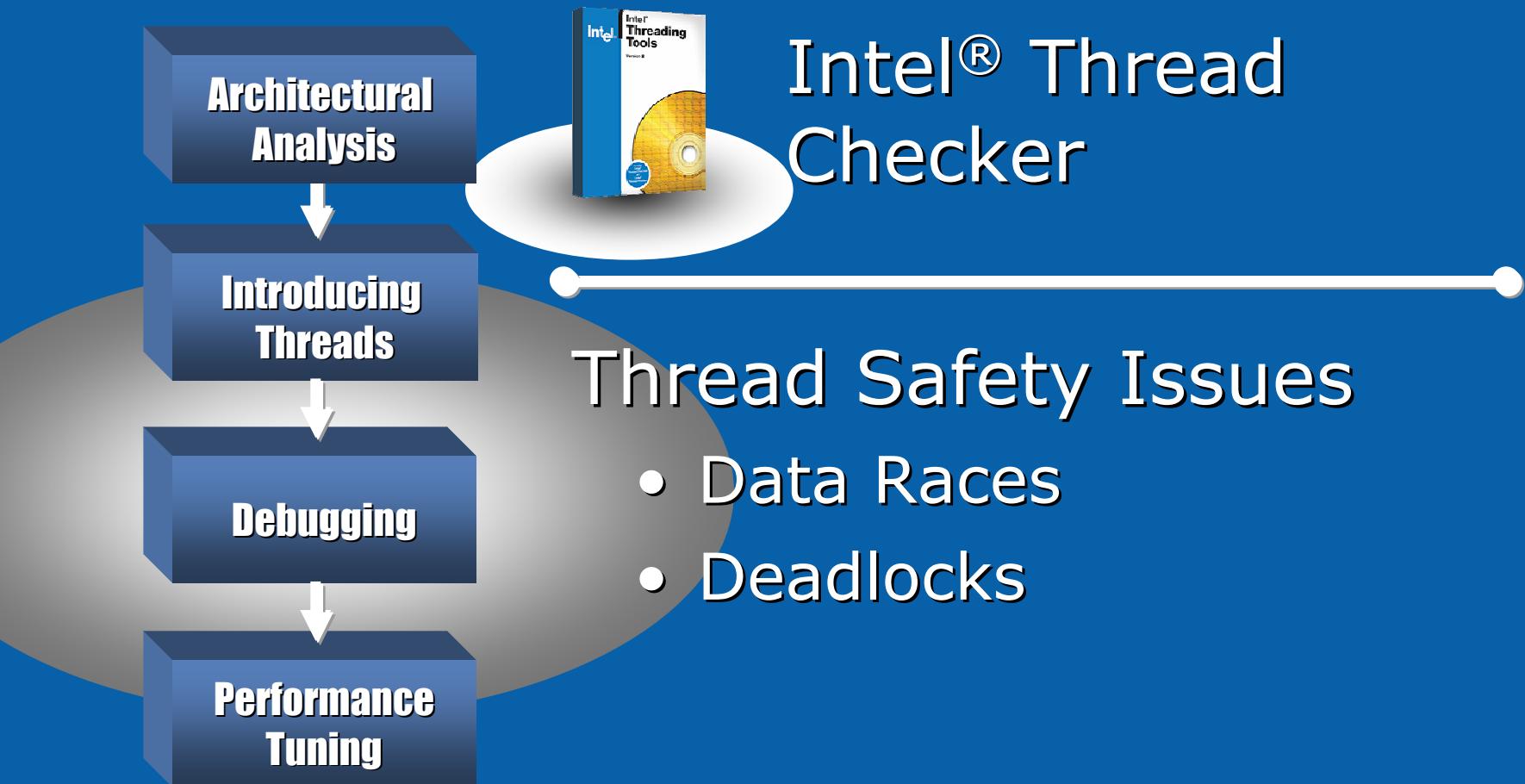
Activity1 (Call Graph)

Source

Line

```
1,339     U32      uElapsedTime = ucurrentTime - uStartTime;
1,340     U32      i = 0;
1,341
1,342     // traverse each palette entry.
1,343     for (i = 0; i < IFXDecoderChainX::kDecoderCount; i++)
1,344         IFXRRESULT iResultTransfer = pDecoderChainX->TransferX(i);
1,345
1,346     // attempt to get the palette index from the palette entry.
1,347     #ifdef TLP_IMPORT_FRONT
1,348         U32 uPaletteIndex = pTable[ui];
1,349
1,350     #else
1,351         U32 uPaletteIndex = pTable[i];
1,352
1,353     #endif
1,354
1,355         if (iResultTransfer == IFX_OK)
1,356             #pragma omp parallel for schedule(runtime)
1,357             for ( intx = 0; intx < ii; intx++ )
1,358             {
1,359                 U32 uPaletteIndex = pTable[intx];
1,360
1,361                 if (iResultTransfer == IFX_OK)
1,362                     m_ppDecoderPalettes[i]->First(&uPaletteIndex); IFXSUCCESS(iPaletteIteratorReturnCode);
1,363
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));
1,368
1,369                 U32 uDecoderCount = 0;
1,370                 pDecoderChainX->GetDecoderCountX(uDecoderCount);
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);
1,377                     if (pDecoderX)
1,378                     {
1,379                         // Perform idling activities.
1,380                     #ifdef TLP_IMPORT_FRONT
1,381                         if ( tid == IFXGetThreadID() )
1,382                     #endif
1,383                         ThumpX();
1,384
1,385                         iResultTransfer = IFX_OK;
1,386                         pDecoderX->TransferX(iResultTransfer);
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
1,391                         // then we can move on to the next decoder in the palette entry.
1,392                     #endif
1,393                 }
1,394             }
1,395         }
1,396     }
1,397
1,398     // If a palette entry has been processed then we can move on to the next palette entry.
1,399
1,400     if (i == ii)
1,401         break;
1,402
1,403     if (i == ii)
1,404         break;
1,405
1,406     if (i == ii)
1,407         break;
1,408
1,409     if (i == ii)
1,410         break;
1,411
1,412     if (i == ii)
1,413         break;
1,414
1,415     if (i == ii)
1,416         break;
1,417
1,418     if (i == ii)
1,419         break;
1,420
1,421     if (i == ii)
1,422         break;
1,423
1,424     if (i == ii)
1,425         break;
1,426
1,427     if (i == ii)
1,428         break;
1,429
1,430     if (i == ii)
1,431         break;
1,432
1,433     if (i == ii)
1,434         break;
1,435
1,436     if (i == ii)
1,437         break;
1,438
1,439     if (i == ii)
1,440         break;
1,441
1,442     if (i == ii)
1,443         break;
1,444
1,445     if (i == ii)
1,446         break;
1,447
1,448     if (i == ii)
1,449         break;
1,450
1,451     if (i == ii)
1,452         break;
1,453
1,454     if (i == ii)
1,455         break;
1,456
1,457     if (i == ii)
1,458         break;
1,459
1,460     if (i == ii)
1,461         break;
1,462
1,463     if (i == ii)
1,464         break;
1,465
1,466     if (i == ii)
1,467         break;
1,468
1,469     if (i == ii)
1,470         break;
1,471
1,472     if (i == ii)
1,473         break;
1,474
1,475     if (i == ii)
1,476         break;
1,477
1,478     if (i == ii)
1,479         break;
1,480
1,481     if (i == ii)
1,482         break;
1,483
1,484     if (i == ii)
1,485         break;
1,486
1,487     if (i == ii)
1,488         break;
1,489
1,490     if (i == ii)
1,491         break;
1,492
1,493     if (i == ii)
1,494         break;
1,495
1,496     if (i == ii)
1,497         break;
1,498
1,499     if (i == ii)
1,500         break;
1,501
1,502     if (i == ii)
1,503         break;
1,504
1,505     if (i == ii)
1,506         break;
1,507
1,508     if (i == ii)
1,509         break;
1,510
1,511     if (i == ii)
1,512         break;
1,513
1,514     if (i == ii)
1,515         break;
1,516
1,517     if (i == ii)
1,518         break;
1,519
1,520     if (i == ii)
1,521         break;
1,522
1,523     if (i == ii)
1,524         break;
1,525
1,526     if (i == ii)
1,527         break;
1,528
1,529     if (i == ii)
1,530         break;
1,531
1,532     if (i == ii)
1,533         break;
1,534
1,535     if (i == ii)
1,536         break;
1,537
1,538     if (i == ii)
1,539         break;
1,540
1,541     if (i == ii)
1,542         break;
1,543
1,544     if (i == ii)
1,545         break;
1,546
1,547     if (i == ii)
1,548         break;
1,549
1,550     if (i == ii)
1,551         break;
1,552
1,553     if (i == ii)
1,554         break;
1,555
1,556     if (i == ii)
1,557         break;
1,558
1,559     if (i == ii)
1,560         break;
1,561
1,562     if (i == ii)
1,563         break;
1,564
1,565     if (i == ii)
1,566         break;
1,567
1,568     if (i == ii)
1,569         break;
1,570
1,571     if (i == ii)
1,572         break;
1,573
1,574     if (i == ii)
1,575         break;
1,576
1,577     if (i == ii)
1,578         break;
1,579
1,580     if (i == ii)
1,581         break;
1,582
1,583     if (i == ii)
1,584         break;
1,585
1,586     if (i == ii)
1,587         break;
1,588
1,589     if (i == ii)
1,590         break;
1,591
1,592     if (i == ii)
1,593         break;
1,594
1,595     if (i == ii)
1,596         break;
1,597
1,598     if (i == ii)
1,599         break;
1,600
1,601     if (i == ii)
1,602         break;
1,603
1,604     if (i == ii)
1,605         break;
1,606
1,607     if (i == ii)
1,608         break;
1,609
1,610     if (i == ii)
1,611         break;
1,612
1,613     if (i == ii)
1,614         break;
1,615
1,616     if (i == ii)
1,617         break;
1,618
1,619     if (i == ii)
1,620         break;
1,621
1,622     if (i == ii)
1,623         break;
1,624
1,625     if (i == ii)
1,626         break;
1,627
1,628     if (i == ii)
1,629         break;
1,630
1,631     if (i == ii)
1,632         break;
1,633
1,634     if (i == ii)
1,635         break;
1,636
1,637     if (i == ii)
1,638         break;
1,639
1,640     if (i == ii)
1,641         break;
1,642
1,643     if (i == ii)
1,644         break;
1,645
1,646     if (i == ii)
1,647         break;
1,648
1,649     if (i == ii)
1,650         break;
1,651
1,652     if (i == ii)
1,653         break;
1,654
1,655     if (i == ii)
1,656         break;
1,657
1,658     if (i == ii)
1,659         break;
1,660
1,661     if (i == ii)
1,662         break;
1,663
1,664     if (i == ii)
1,665         break;
1,666
1,667     if (i == ii)
1,668         break;
1,669
1,670     if (i == ii)
1,671         break;
1,672
1,673     if (i == ii)
1,674         break;
1,675
1,676     if (i == ii)
1,677         break;
1,678
1,679     if (i == ii)
1,680         break;
1,681
1,682     if (i == ii)
1,683         break;
1,684
1,685     if (i == ii)
1,686         break;
1,687
1,688     if (i == ii)
1,689         break;
1,690
1,691     if (i == ii)
1,692         break;
1,693
1,694     if (i == ii)
1,695         break;
1,696
1,697     if (i == ii)
1,698         break;
1,699
1,700     if (i == ii)
1,701         break;
1,702
1,703     if (i == ii)
1,704         break;
1,705
1,706     if (i == ii)
1,707         break;
1,708
1,709     if (i == ii)
1,710         break;
1,711
1,712     if (i == ii)
1,713         break;
1,714
1,715     if (i == ii)
1,716         break;
1,717
1,718     if (i == ii)
1,719         break;
1,720
1,721     if (i == ii)
1,722         break;
1,723
1,724     if (i == ii)
1,725         break;
1,726
1,727     if (i == ii)
1,728         break;
1,729
1,730     if (i == ii)
1,731         break;
1,732
1,733     if (i == ii)
1,734         break;
1,735
1,736     if (i == ii)
1,737         break;
1,738
1,739     if (i == ii)
1,740         break;
1,741
1,742     if (i == ii)
1,743         break;
1,744
1,745     if (i == ii)
1,746         break;
1,747
1,748     if (i == ii)
1,749         break;
1,750
1,751     if (i == ii)
1,752         break;
1,753
1,754     if (i == ii)
1,755         break;
1,756
1,757     if (i == ii)
1,758         break;
1,759
1,760     if (i == ii)
1,761         break;
1,762
1,763     if (i == ii)
1,764         break;
1,765
1,766     if (i == ii)
1,767         break;
1,768
1,769     if (i == ii)
1,770         break;
1,771
1,772     if (i == ii)
1,773         break;
1,774
1,775     if (i == ii)
1,776         break;
1,777
1,778     if (i == ii)
1,779         break;
1,780
1,781     if (i == ii)
1,782         break;
1,783
1,784     if (i == ii)
1,785         break;
1,786
1,787     if (i == ii)
1,788         break;
1,789
1,790     if (i == ii)
1,791         break;
1,792
1,793     if (i == ii)
1,794         break;
1,795
1,796     if (i == ii)
1,797         break;
1,798
1,799     if (i == ii)
1,800         break;
1,801
1,802     if (i == ii)
1,803         break;
1,804
1,805     if (i == ii)
1,806         break;
1,807
1,808     if (i == ii)
1,809         break;
1,810
1,811     if (i == ii)
1,812         break;
1,813
1,814     if (i == ii)
1,815         break;
1,816
1,817     if (i == ii)
1,818         break;
1,819
1,820     if (i == ii)
1,821         break;
1,822
1,823     if (i == ii)
1,824         break;
1,825
1,826     if (i == ii)
1,827         break;
1,828
1,829     if (i == ii)
1,830         break;
1,831
1,832     if (i == ii)
1,833         break;
1,834
1,835     if (i == ii)
1,836         break;
1,837
1,838     if (i == ii)
1,839         break;
1,840
1,841     if (i == ii)
1,842         break;
1,843
1,844     if (i == ii)
1,845         break;
1,846
1,847     if (i == ii)
1,848         break;
1,849
1,850     if (i == ii)
1,851         break;
1,852
1,853     if (i == ii)
1,854         break;
1,855
1,856     if (i == ii)
1,857         break;
1,858
1,859     if (i == ii)
1,860         break;
1,861
1,862     if (i == ii)
1,863         break;
1,864
1,865     if (i == ii)
1,866         break;
1,867
1,868     if (i == ii)
1,869         break;
1,870
1,871     if (i == ii)
1,872         break;
1,873
1,874     if (i == ii)
1,875         break;
1,876
1,877     if (i == ii)
1,878         break;
1,879
1,880     if (i == ii)
1,881         break;
1,882
1,883     if (i == ii)
1,884         break;
1,885
1,886     if (i == ii)
1,887         break;
1,888
1,889     if (i == ii)
1,890         break;
1,891
1,892     if (i == ii)
1,893         break;
1,894
1,895     if (i == ii)
1,896         break;
1,897
1,898     if (i == ii)
1,899         break;
1,900
1,901     if (i == ii)
1,902         break;
1,903
1,904     if (i == ii)
1,905         break;
1,906
1,907     if (i == ii)
1,908         break;
1,909
1,910     if (i == ii)
1,911         break;
1,912
1,913     if (i == ii)
1,914         break;
1,915
1,916     if (i == ii)
1,917         break;
1,918
1,919     if (i == ii)
1,920         break;
1,921
1,922     if (i == ii)
1,923         break;
1,924
1,925     if (i == ii)
1,926         break;
1,927
1,928     if (i == ii)
1,929         break;
1,930
1,931     if (i == ii)
1,932         break;
1,933
1,934     if (i == ii)
1,935         break;
1,936
1,937     if (i == ii)
1,938         break;
1,939
1,940     if (i == ii)
1,941         break;
1,942
1,943     if (i == ii)
1,944         break;
1,945
1,946     if (i == ii)
1,947         break;
1,948
1,949     if (i == ii)
1,950         break;
1,951
1,952     if (i == ii)
1,953         break;
1,954
1,955     if (i == ii)
1,956         break;
1,957
1,958     if (i == ii)
1,959         break;
1,960
1,961     if (i == ii)
1,962         break;
1,963
1,964     if (i == ii)
1,965         break;
1,966
1,967     if (i == ii)
1,968         break;
1,969
1,970     if (i == ii)
1,971         break;
1,972
1,973     if (i == ii)
1,974         break;
1,975
1,976     if (i == ii)
1,977         break;
1,978
1,979     if (i == ii)
1,980         break;
1,981
1,982     if (i == ii)
1,983         break;
1,984
1,985     if (i == ii)
1,986         break;
1,987
1,988     if (i == ii)
1,989         break;
1,990
1,991     if (i == ii)
1,992         break;
1,993
1,994     if (i == ii)
1,995         break;
1,996
1,997     if (i == ii)
1,998         break;
1,999
1,1000     if (i == ii)
1,1001         break;
1,1002
1,1003     if (i == ii)
1,1004         break;
1,1005
1,1006     if (i == ii)
1,1007         break;
1,1008
1,1009     if (i == ii)
1,1010         break;
1,1011
1,1012     if (i == ii)
1,1013         break;
1,1014
1,1015     if (i == ii)
1,1016         break;
1,1017
1,1018     if (i == ii)
1,1019         break;
1,1020
1,1021     if (i == ii)
1,1022         break;
1,1023
1,1024     if (i == ii)
1,1025         break;
1,1026
1,1027     if (i == ii)
1,1028         break;
1,1029
1,1030     if (i == ii)
1,1031         break;
1,1032
1,1033     if (i == ii)
1,1034         break;
1,1035
1,1036     if (i == ii)
1,1037         break;
1,1038
1,1039     if (i == ii)
1,1040         break;
1,1041
1,1042     if (i == ii)
1,1043         break;
1,1044
1,1045     if (i == ii)
1,1046         break;
1,1047
1,1048     if (i == ii)
1,1049         break;
1,1050
1,1051     if (i == ii)
1,1052         break;
1,1053
1,1054     if (i == ii)
1,1055         break;
1,1056
1,1057     if (i == ii)
1,1058         break;
1,1059
1,1060     if (i == ii)
1,1061         break;
1,1062
1,1063     if (i == ii)
1,1064         break;
1,1065
1,1066     if (i == ii)
1,1067         break;
1,1068
1,1069     if (i == ii)
1,1070         break;
1,1071
1,1072     if (i == ii)
1,1073         break;
1,1074
1,1075     if (i == ii)
1,1076         break;
1,1077
1,1078     if (i == ii)
1,1079         break;
1,1080
1,1081     if (i == ii)
1,1082         break;
1,1083
1,1084     if (i == ii)
1,1085         break;
1,1086
1,1087     if (i == ii)
1,1088         break;
1,1089
1,1090     if (i == ii)
1,1091         break;
1,1092
1,1093     if (i == ii)
1,1094         break;
1,1095
1,1096     if (i == ii)
1,1097         break;
1,1098
1,1099     if (i == ii)
1,1100         break;
1,1101
1,1102     if (i == ii)
1,1103         break;
1,1104
1,1105     if (i == ii)
1,1106         break;
1,1107
1,1108     if (i == ii)
1,1109         break;
1,1110
1,1111     if (i == ii)
1,1112         break;
1,1113
1,1114     if (i == ii)
1,1115         break;
1,1116
1,1117     if (i == ii)
1,1118         break;
1,1119
1,1120     if (i == ii)
1,1121         break;
1,1122
1,1123     if (i == ii)
1,1124         break;
1,1125
1,1126     if (i == ii)
1,1127         break;
1,1128
1,1129     if (i == ii)
1,1130         break;
1,1131
1,1132     if (i == ii)
1,1133         break;
1,1134
1,1135     if (i == ii)
1,1136         break;
1,1137
1,1138     if (i == ii)
1,1139         break;
1,1140
1,1141     if (i == ii)
1,1142         break;
1,1143
1,1144     if (i == ii)
1,1145         break;
1,1146
1,1147     if (i == ii)
1,1148         break;
1,1149
1,1150     if (i == ii)
1,1151         break;
1,1152
1,1153     if (i == ii)
1,1154         break;
1,1155
1,1156     if (i == ii)
1,1157         break;
1,1158
1,1159     if (i == ii)
1,1160         break;
1,1161
1,1162     if (i == ii)
1,1163         break;
1,1164
1,1165     if (i == ii)
1,1166         break;
1,1167
1,1168     if (i == ii)
1,1169         break;
1,1170
1,1171     if (i == ii)
1,1172         break;
1,1173
1,1174     if (i == ii)
1,1175         break;
1,1176
1,1177     if (i == ii)
1,1178         break;
1,1179
1,1180     if (i == ii)
1,1181         break;
1,1182
1,1183     if (i == ii)
1,1184         break;
1,1185
1,1186     if (i == ii)
1,1187         break;
1,1188
1,1189     if (i == ii)
1,1190         break;
1,1191
1,1192     if (i == ii)
1,1193         break;
1,1194
1,1195     if (i == ii)
1,1196         break;
1,1197
1,1198     if (i == ii)
1,1199         break;
1,1200
1,1201     if (i == ii)
1,1202         break;
1,1203
1,1204     if (i == ii)
1,1205         break;
1,1206
1,1207     if (i == ii)
1,1208         break;
1,1209
1,1210     if (i == ii)
1,1211         break;
1,1212
1,1213     if (i == ii)
1,1214         break;
1,1215
1,1216     if (i == ii)
1,1217         break;
1,1218
1,1219     if (i == ii)
1,1220         break;
1,1221
1,1222     if (i == ii)
1,1223         break;
1,1224
1,1225     if (i == ii)
1,1226         break;
1,1227
1,1228     if (i == ii)
1,1229         break;
1,1230
1,1231     if (i == ii)
1,1232         break;
1,1233
1,1234     if (i == ii)
1,1235         break;
1,1236
1,1237     if (i == ii)
1,1238         break;
1,1239
1,1240     if (i == ii)
1,1241         break;
1,1242
1,1243     if (i == ii)
1,1244         break;
1,1245
1,1246     if (i == ii)
1,1247         break;
1,1248
1,1249     if (i == ii)
1,1250         break;
1,1251
1,1252     if (i == ii)
1,1253         break;
1,1254
1,1255     if (i == ii)
1,1256         break;
1,1257
1,1258     if (i == ii)
1,1259         break;
1,1260
1,1261     if (i == ii)
1,1262         break;
1,1263
1,1264     if (i == ii)
1,1265         break;
1,1266
1,1267     if (i == ii)
1,1268         break;
1,1269
1,1270     if (i == ii)
1,1271         break;
1,1272
1,1273     if (i == ii)
1,1274         break;
1,1275
1,1276     if (i == ii)
1,1277         break;
1,1278
1,1279     if (i == ii)
1,1280         break;
1,1281
1,1282     if (i == ii)
1,1283         break;
1,1284
1,1285     if (i == ii)
1,1286         break;
1,1287
1,1288     if (i == ii)
1,1289         break;
1,1290
1,1291     if (i == ii)
1,1292         break;
1,1293
1,1294     if (i == ii)
1,1295         break;
1,1296
1,1297     if (i == ii)
1,1298         break;
1,1299
1,1300     if (i == ii)
1,1301         break;
1,1302
1,1303     if (i == ii)
1,1304         break;
1,1305
1,1306     if (i == ii)
1,1307         break;
1,1308
1,1309     if (i == ii)
1,1310         break;
1,1311
1,1312     if (i == ii)
1,1313         break;
1,1314
1,1315     if (i == ii)
1,1316         break;
1,1317
1,1318     if (i == ii)
1,1319         break;
1,1320
1,1321     if (i == ii)
1,1322         break;
1,1323
1,1324     if (i == ii)
1,1325         break;
1,1326
1,1327     if (i == ii)
1,1328         break;
1,1329
1,1330     if (i == ii)
1,1331         break;
1,1332
1,1333     if (i == ii)
1,1334         break;
1,1335
1,1336     if (i == ii)
1,1337         break;
1,1338
1,1339     if (i == ii)
1,1340         break;
1,1341
1,1342     if (i == ii)
1,1343         break;
1,1344
1,1345     if (i == ii)
1,1346         break;
1,1347
1,1348     if (i == ii)
1,1349         break;
1,1350
1,1351     if (i == ii)
1,1352         break;
1,1353
1,1354     if (i == ii)
1,1355         break;
1,1356
1,1357     if (i == ii)
1,1358         break;
1,1359
1,1360     if (i == ii)
1,1361         break;
1,1361
1,1362     if (i == ii)
1,1363         break;
1,1362
1,1363     if (i == ii)
1,1364         break;
1,1364
1,1365     if (i == ii)
1,1366         break;
1,1365
1,1366     if (i == ii)
1,1367         break;
1,1367
1,1368     if (i == ii)
1,1369         break;
1,1368
1,1369     if (i == ii)
1,1370         break;
1,1369
1,1370     if (i == ii)
1,1371         break;
1,1371
1,1372     if (i == ii)
1,1373         break;
1,1372
1,1373     if (i == ii)
1,1374         break;
1,1373
1,1374     if (i == ii)
1,1375         break;
1,1374
1,1375     if (i == ii)
1,1376         break;
1,1375
1,1376     if (i == ii)
1,1377         break;
1,1376
1,1377     if (i == ii)
1,1378         break;
1,1377
1,1378     if (i == ii)
1,1379         break;
1,1378
1,1379     if (i == ii)
1,1380         break;
1,1379
1,1380     if (i == ii)
1,1381         break;
1,1381
1,1382     if (i == ii)
1,1383         break;
1,1382
1,1383     if (i == ii)
1,1384         break;
1,1383
1,1384     if (i == ii)
1,1385         break;
1,1384
1,1385     if (i == ii)
1,1386         break;
1,1385
1,1386     if (i == ii)
1,1387         break;
1,1386
1,1387     if (i == ii)
1,1388         break;
1,1387
1,1388     if (i == ii)
1,1389         break;
1,1388
1,1389     if (i == ii)
1,1390         break;
1,1389
1,1390     if (i == ii)
1,1391         break;
1,1391
1,1392     if (i == ii)
1,1393         break;
1,1392
1,1393     if (i == ii)
1,1394         break;
1,1393
1,1394     if (i == ii)
1,1395         break;
1,1394
1,1395     if (i == ii)
1,1396         break;
1,1395
1,1396     if (i == ii)
1,1397         break;
1,1396
1,1397     if (i == ii)
1,1398         break;
1,1397
1,1398     if (i == ii)
1,1399         break;
1,1398
1,1399     if (i == ii)
1,1400         break;
1,1399
1,1400     if (i == ii)
1,1401         break;
1,1401
1,1402     if (i == ii)
1,1403         break;
1,1402
1,1403     if (i == ii)
1,1404         break;
1,1403
1,1404     if (i == ii)
1,1405         break;
1,1404
1,1405     if (i == ii)
1,1406         break;
1,1405
1,1406     if (i == ii)
1,1407         break;
1,1406
1,1407     if (i == ii)
1,1408         break;
1,1407
1,1408     if (i == ii)
1,1409         break;
1,1408
1,1409     if (i == ii)
1,1410         break;
1,1409
1,1410     if (i == ii)
1,1411         break;
1,1411
1,1412     if (i == ii)
1,1413         break;
1,1412
1,1413     if (i == ii)
1,1414         break;
1,1413
1,1414     if (i == ii)
1,1415         break;
1,1414
1,1415     if (i == ii)
1,1416         break;
1,1415
1,1416     if (i == ii)
1,1417         break;
1,1416
1,1417     if (i == ii)
1,1418         break;
1,1417
1,1419     if (i == ii)
1,1420         break;
1,1419
1,1420     if (i == ii)
1,1421         break;
1,1421
1,1422     if (i == ii)
1,1423         break;
1,1422
1,1423     if (i == ii)
1,1424         break;
1,1423
1,1425     if (i == ii)
1,1426         break;
1,1424
1,1427     if (i == ii)
1,1428         break;
1,1426
1,1429     if (i == ii)
1,1430         break;
1,1428
1,1431     if (i == ii)
1,1432         break;
1,1430
1,1433     if (i == ii)
1,1434         break;
1,1432
1,1435     if (i == ii)
1,1436         break;
1,1434
1,1437     if (i == ii)
1,1438         break;
1,1436
1,1439     if (i == ii)
1,1440         break;
1,1438
1,1441     if (i == ii)
1,1442         break;
1,1440
1,1443     if (i == ii)
1,1444         break;
1,1442
1,1445     if (i == ii)
1,1446         break;
1,1444
1,1447     if (i == ii)
1,1448         break;
1,1446
1,1449     if (i == ii)
1,1450         break;
1,1448
1,1451     if (i == ii)
1,1452         break;
1,1450
1,1453     if (i == ii)
1,1454         break;
1,1452
1,1455     if (i == ii)
1,1456         break;
1,1454
1,1457     if (i == ii)
1,1458         break;
1,1456
1,1459     if (i == ii)
1,1460         break;
1,1458
1,1461     if (i == ii)
1,1462         break;
1,1460
1,1463     if (i == ii)
1,1464         break;
1,1462
1,1465     if (i == ii)
1,1466         break;
1,1464
1,1467     if (i == ii)
1,1468         break;
1,1466
1,1469     if (i == ii)
1,1470         break;
1,1468
1,1471     if (i == ii)
1,1472         break;
1,1470
1,1473     if (i == ii)
1,1474         break;
1,1472
1,1475     if (i == ii)
1,1476         break;
1,1474
1,1477     if (i == ii)
1,1478         break;
1,1476
1,1479     if (i == ii)
1,1480         break;
1,1478
1,1481     if (i == ii)
1,1482         break;
1,1480
1,1483     if (i == ii)
1,1484         break;
1,1482
1,1485     if (i == ii)
1,1486         break;
1,1484
1,1487     if (i == ii)
1,1488         break;
1,1486
1,1489     if (i == ii)
1,1490         break;
1,1488
1,1491     if (i == ii)
1,1492         break;
1,1490
1,1493     if (i == ii)
1,1494         break;
1,1492
```

# Threading for Multi-Core





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, 0x44d9]
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]
```

### Source

```
// Iterate -- follow all of the invalidation sequences
while( IFXSUCCESS( result ) && s_InvDepth > StartDepth )
{
    InvRecord* pCurIterState = s_pInvState + s_InvDepth;

    // Get the current Inv Seq
    IFXModifierDataPacketInternal* pDP =
        pDataPacketState[pCurIterState->ModIdx].m_pDataPacket;
    IFXDidInvElement* pInvEl =
        &(pCurIterState->pDEState->m_pInvSeq[pCurIterState->InvIdx]);
    pCurIterState->InvIdx++;

    // pop this iter state if we are processing the last entry
    if( pCurIterState->InvIdx == pCurIterState->pDEState->m_uInvCount )
    {
        IFXInterlockedDecrement( (U32*)&s_InvDepth );
    }

    // Get the Invalidation Target and Do The Invalidation
    if( pInvEl->uMIndex != APPENDED_DATAPACKET_INDEX )
    {
        IFXDataPacketState* pTrgDPState =
            &nDataPackerState[nInvEl->uMIndex];
        pTrgDPState->
```

### 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, 0x44d9]
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
[IFXImporting.dll, 0x18eb6]
```

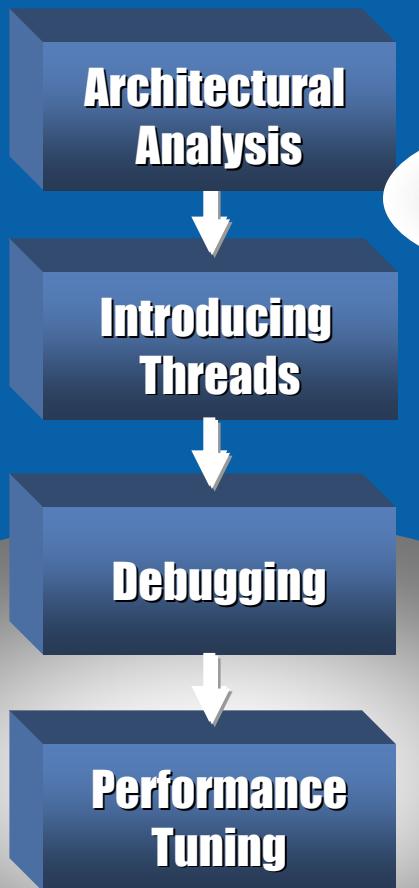
### Source

```
result = IFX_E_INVALID_RANGE;

if( IFXSUCCESS( result ) )
{
    // Set the state for the Initial invalidation
    IFXAquireMutex( s_mInvState );
    s_pInvState[s_InvDepth].ModIdx = uInModifierIndex;
    s_pInvState[s_InvDepth].pDEState =
        &(pDataPacketState[uInModifierIndex].m_pDataElements[uInDataElementIndex]);
    s_pInvState[s_InvDepth].InvIdx = 0;
    IFXReleaseMutex( s_mInvState );
}

// we never actually invalidate the proxy data packet
// all of the proxy data packet entries except for time
// should always be valid
if( IFXSUCCESS( result ) && uInModifierIndex != 0 )
    // invalidate this element
    s_pInvState[s_InvDepth].pDEState->State = IFXDATAELEMENTSTATE_INVALID;
    if( s_nInvState[s_InvDepth].nDEState->AspectBit_1 )
```

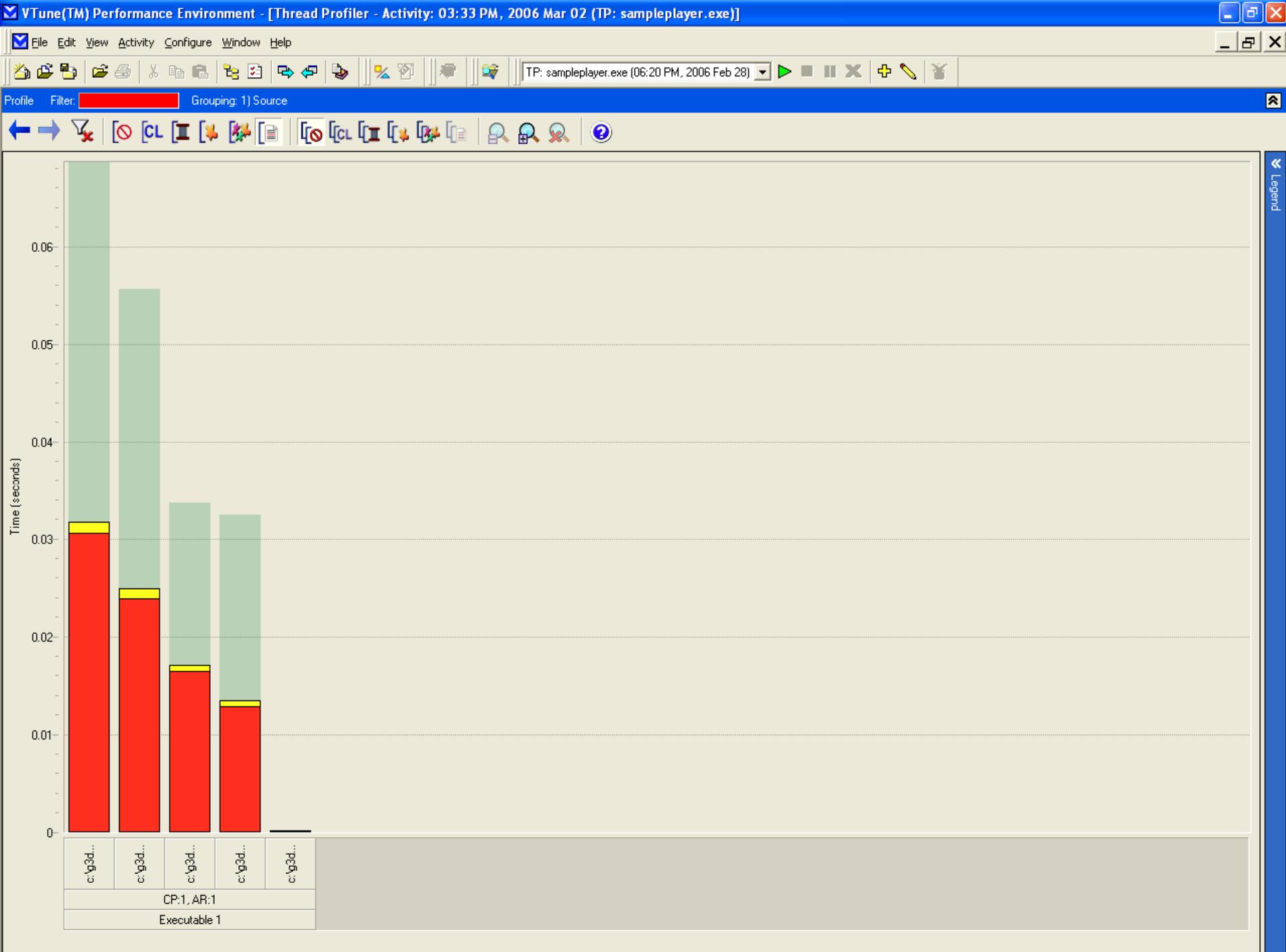
# Threading for Multi-Core



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)]

File Edit View Activity Configure Window Help

Source View

Signal

Receive

Transition Source

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

Stack:  
IFXAcquireMutex  
"ifxosthreads.cpp": 105  
Path: c:\g3force\depot\cwg\mpu3d\source\rtl\platform\win32\int CIFXAuthorCLODResource::SetAuthorMesh(class IFXAuthorC  
"cifxauthorclodresource.cpp": 589  
Path: c:\g3force\depot\cwg\mpu3d\source\rtl\component\ger  
CIFXAuthorCLODDecoder::~CIFXAuthorCLODDecoder(void)  
"CIFXAuthorCLODDecoder.cpp": 257  
Path: c:\g3force\depot\CWG\MPU3D\Source\RTL\Component  
void \* operator new[](unsigned int)  
"IFXCheckX.h": 66  
Path: ..\..\Component\Importing\..\..\Kernel\Include  
Address: 0xb1bguide40.dll  
Module: 3174  
Path: c:\g3force\depot\CWG\MPU3D\Source\Build\J3D

Address	Line	Source
0x223E6	581	}
0x223EF	582	rpAuthorCLODMesh = m_pAuthorMesh;
0x223F2	583	IFXRETURN(rc);
0x23150	584	}
0x23150	585	IFXRESULT rc = IFX_OK;
0x23150	586	IFXAcquireMutex( s_mSetAuthorMesh );
0x23161	587	if(m_pAuthorMesh != pAuthorCLODMesh)
0x2316E	588	{
0x23179	589	ClearMeshGroup();
0x2317D	590	if(pAuthorCLODMesh)
0x2317D	591	{
0x2317D	592	pAuthorCLODMesh->AddRef();
0x2317D	593	}
0x2319C	594	IFXRELEASE(m_pAuthorMesh);
0x231B6	595	m_pAuthorMesh = pAuthorCLODMesh;
0x231C0	596	m_bMeshGroupDirty = TRUE;
0x231D0	597	if(m_pModifierDataPacket) {
0x231D0	598	m_pModifierDataPacket->InvalidateDataElement(m_uMeshGroupDataElementIndex);
0x231D0	599	m_pModifierDataPacket->InvalidateDataElement(m_uBoundSphereDataElementIndex);
0x231D0	600	}
0x231D0	601	IFXReleaseMutex( s_mSetAuthorMesh );
0x231E0	602	IFXRETURN(rc);
0x22400	603	IFXRESULT CIFXAuthorCLODResource::GetAuthorMeshMap(IFXMeshMap **ppAuthorMeshMap)
0x22405	604	{
0x22407	605	IFXRESULT rc = IFX_OK;
0x2240D	606	if (ppAuthorMeshMap)
0x2240D	607	{
0x2240D	608	if(m_pAuthorMeshMap)

Signal

Receive

Transition Source

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

Stack:  
IFXReleaseMutex  
"ifxosthreads.cpp": 113  
Path: c:\g3force\depot\cwg\mpu3d\source\rtl\platform\win32\int CIFXAuthorCLODResource::SetAuthorMesh(class IFXAuthorC  
"cifxauthorclodresource.cpp": 614  
Path: c:\g3force\depot\cwg\mpu3d\source\rtl\component\ger  
CIFXAuthorCLODDecoder::~CIFXAuthorCLODDecoder(void)  
"CIFXAuthorCLODDecoder.cpp": 257  
Path: c:\g3force\depot\CWG\MPU3D\Source\RTL\Component  
void \* operator new[](unsigned int)  
"IFXCheckX.h": 66  
Path: ..\..\Component\Importing\..\..\Kernel\Include  
Address: 0xb1bguide40.dll  
Module: 3174  
Path: c:\g3force\depot\CWG\MPU3D\Source\Build\J3D

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

# Growing Momentum For Software Parallelization

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

# 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

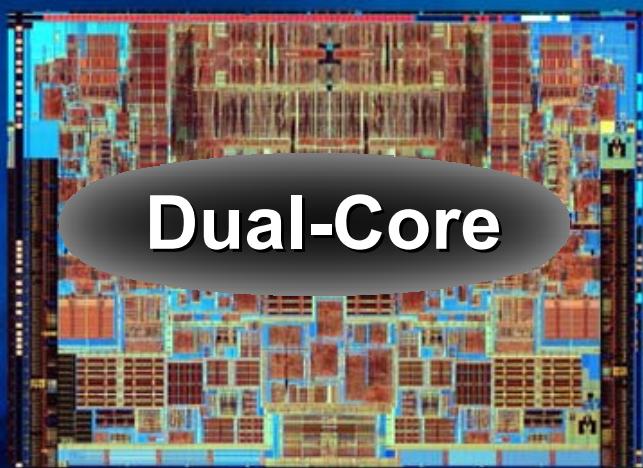
Multi-Core

Power Efficiency

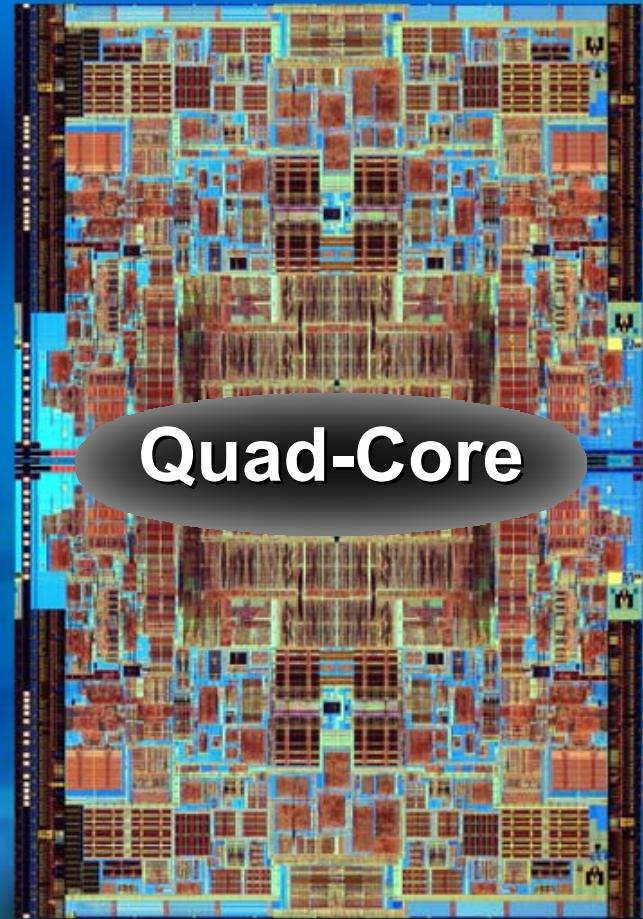
Microarchitecture  
Advancements

*It is  
happening  
fast...*

# Multi-Core Trajectory



2006



2007



# Future Architecture

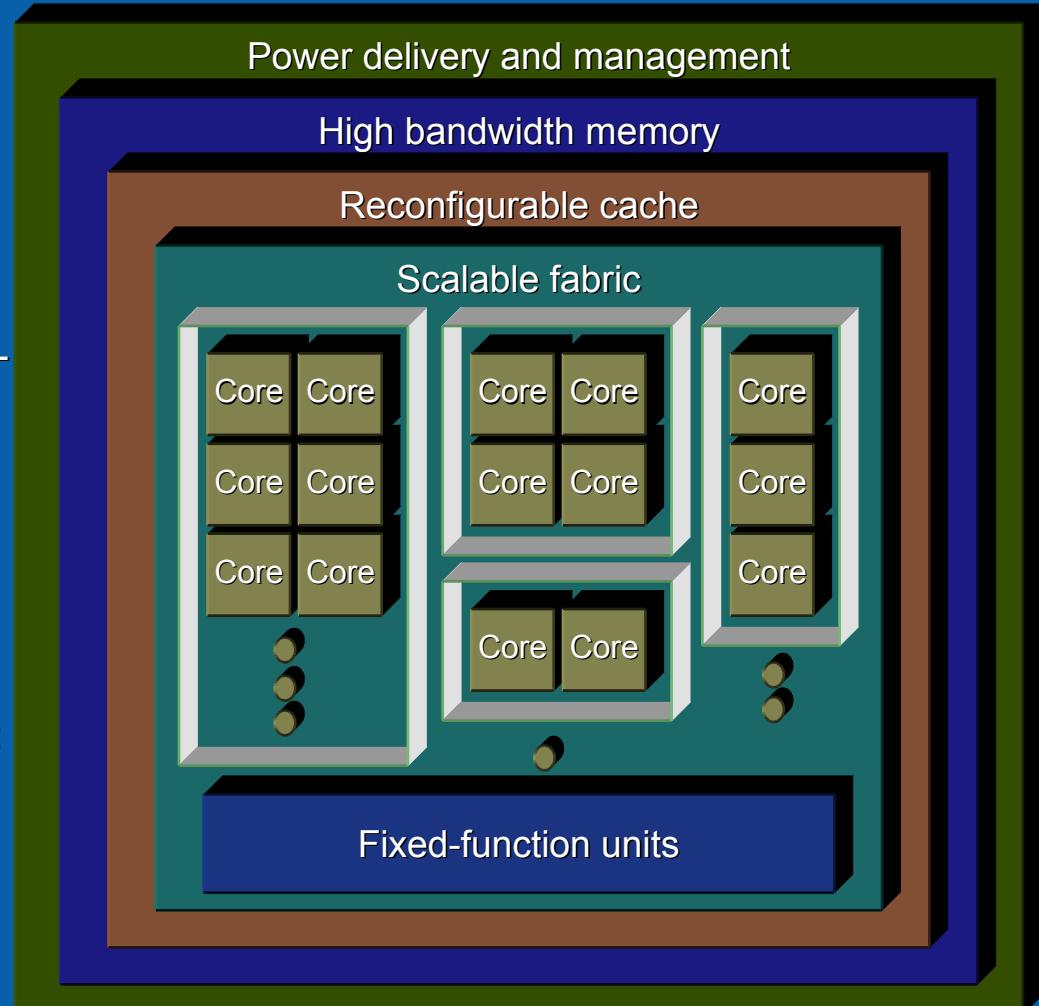
## Many More Cores

### 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

Imagine what can be  
Create what will be

# Parallel Software 2.0

*The Beginning of a New Era*