Threading and Tuning

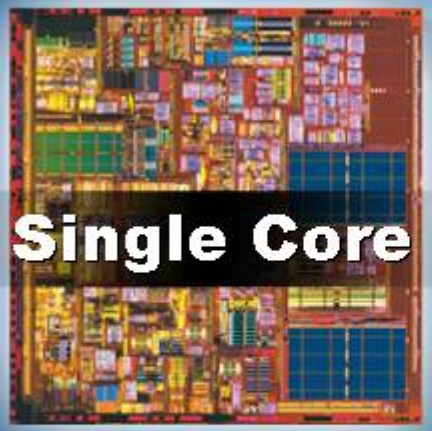
Richard Wirt
Intel Senior Fellow
Corporate Vice President
General Manager,
Software and Solutions Group

Intel Developer FORUM

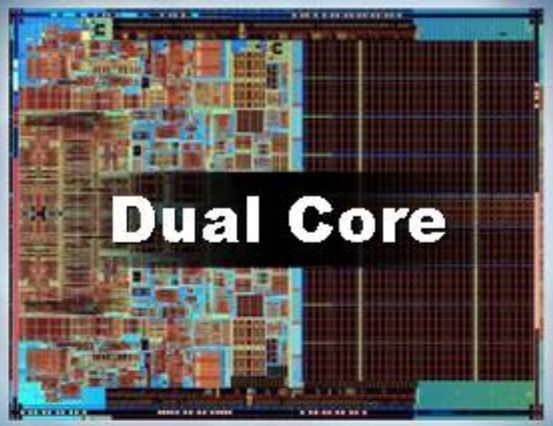

Processor Evolution

Processor Evolution

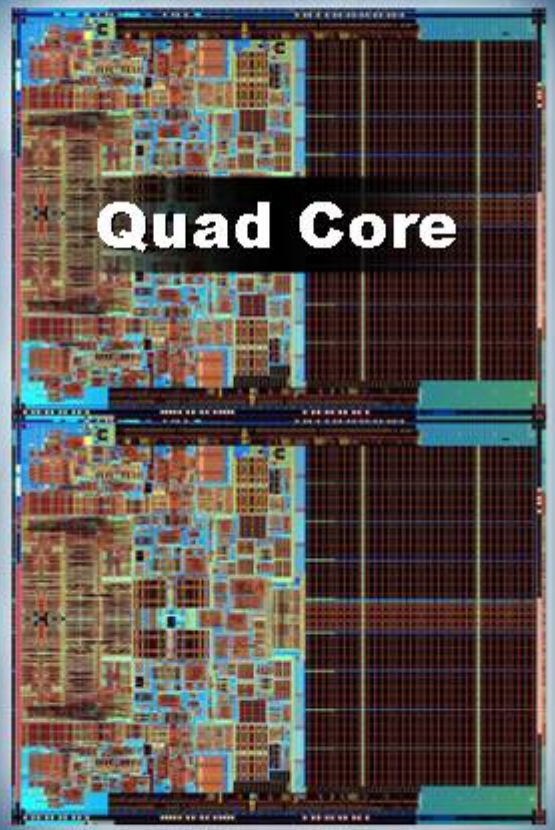
Intel® Xeon® processor



Dual-Core Intel® Xeon® processor 5100 series



New Quad-Core Intel® Xeon® 5300 for 2006

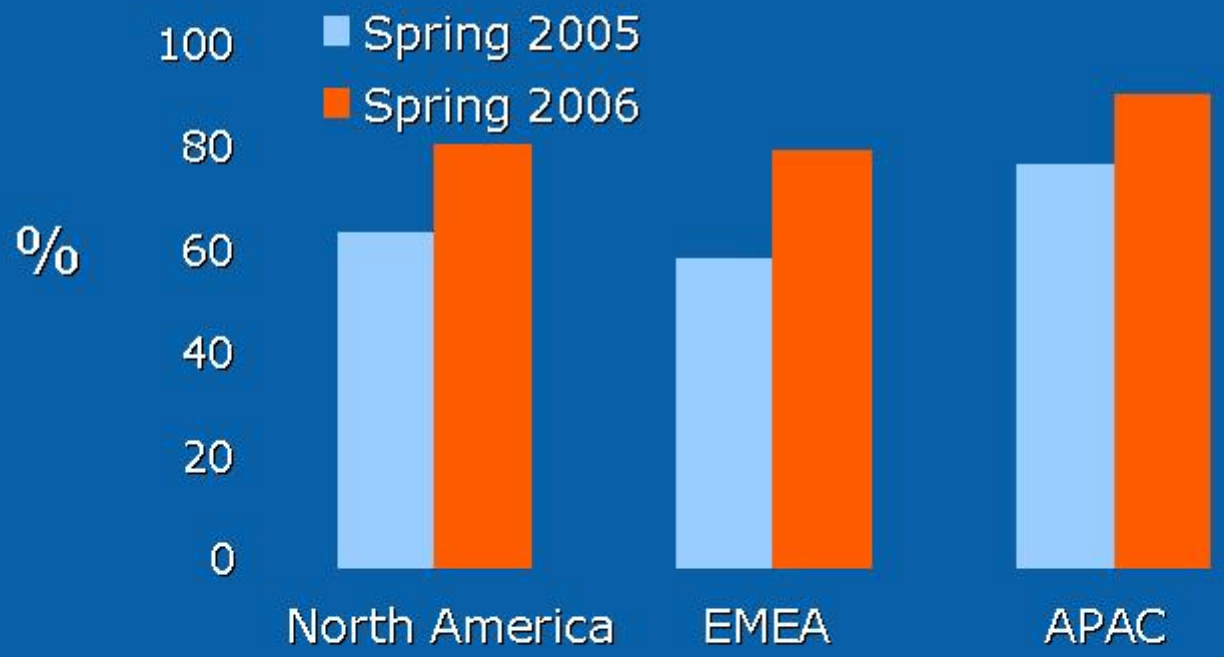


Multi-Core Drives Application Innovation

- Multi-language conversations are voice translated real-time and recorded into text
- Download and burn movies, run a data-mining search, play a demanding game
  - On one PC with smooth performance
- Create system security with robust virtualized systems
  - Stream and store data from multiple Internet sources with one operating system, secure analysis on another
- Immersive 3D and real-time data re-create real locations for virtual visits or inspections

Multi-Core Drives Threading

Percentage of developers working with at least one multithreaded application



Source: EDC Development Surveys, APAC, EMEA, North America, Spring 2005/2006

Growing Momentum for Threading

Activision
Abacus
Adobe
Accela
Algorithmics
Alias
Agilix
Autodesk
Bea
BeInSync
Business Objects
Buzz 3D and Co.
Cadence
Cakewalk
Canopus
CodecPeople
Codemasters
Computer Associates
Corel
CounterPath
Cryptic Studios
Cyberlink
Digidesign
Discreet
EA
Epic Games
ESI
EXA
FlowScience
Fluent
Gaussian
Google
IBM
id Software
Intel
Intervideo
Isis
Juice Games
Kaspersky
Kingsoft
LandMark
Lionhead Studios
Macromedia
MainConcept
Magma
Maxon
mental images
Mentor Graphics
Microsoft
Midway
Morgan Stanley
MSC
Mythic
Namco
Native Instruments
Neatware
NCSoft
Novell SUSE
Oracle
Paradigm
Parametric
Pegasus
Pinnacle
Pixar
RealNetworks
Red Hat
SAP
SAS
Siebel CRM
Signet
Skype
SLB
SnapStream
Sonic (Roxio)
Sony Square - Enix
Stratum Global
Steinberg-Yamaha
SunGard
SyAM
Sybase
Symantec
Thomson
Thompson
THQ
TrendMicro
Ubisoft
UGS
Valve
Webzen
Yahoo
Zenrin

The companies on this list are threading at least one application for multi-core that may be in any one of these phases: design, production or already in the marketplace.

Microarchitecture Tuning

- Optimize for the specific underlying processor and system architecture
  - Data alignment, branching, and cache usage
- These optimizations can lead to significant performance improvements
  - Should only be attempted when other optimizations are complete
  - Requires the greatest developer, sustaining, and support effort

Application Tuning

- Identify sections of code along the critical execution path that can be optimized
  - The critical execution path is the set of functions that take the longest time to execute
- By working to improve the performance of these items, the critical path is reduced and the overall execution time improves
  - Most common tuning area
  - Generally the result of thread interaction and synchronization, choice of data structures, loops, and system APIs

System Tuning

- Identify hardware access slowdowns that limit performance
  - Example: A database application may be limited by disk I/O, not the query
- Common causes here are related to disk I/O, network I/O, memory access, and processor bandwidth/utilization

Parallel Programming

- Parallelism in software is delivered via multi-threading capabilities
- Multi-thread performance will increase as the number of independent threads increases
- Multi-core technology is the next evolution of Hyper-Threading by extending parallelism into the CPU
- Multi-core processors introduce separate sets of processor resources, thus increasing the effectiveness of parallelism

Using More Threads

Do more of the same

- Divide current tasks among multiple processors
- Examples:
  - Process photo effect faster
  - Show more frames per second

Add new functionality

- Extend user experience
- Examples:
  - Download updates in the background
  - Dynamically compute the image of the sky

To program this, we use threading models

- Data (domain) decomposition
  - Split up the data
  - Work on different parts of the data at the same time
- Functional decomposition
  - Split up the kinds of work
  - Work on different kinds of work at the same time

Example: Windows Multiple Threads

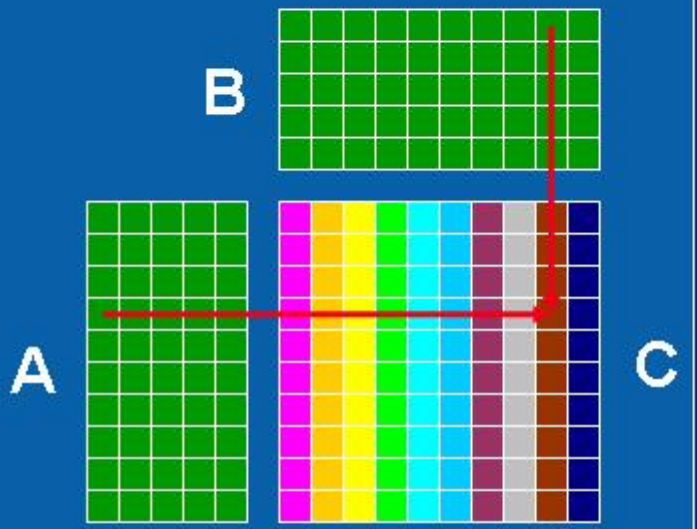
```
#include <iostream>
#include <windows.h>
using namespace std;
const int numThreads = 4;

DWORD WINAPI HelloFunc (LPVOID arg)
{
    cout << "Hello Thread\n";
    return 0;
}
main ()
{
    HANDLE hThread[numThreads];

    for (int i = 0; i < numThreads; i++)
       hThread[i] =
          CreateThread (NULL, 0, HelloFunc, NULL, 0, NULL );

    WaitForMultipleObjects (numThreads, hThread, TRUE,
INFINITE);
}
```

Example: OpenMP Matrix Multiply




```
#pragma omp parallel for
shared(C) private(i,j)
for (i = 0; i < M; i++)
    for (j = 0; j < N; j++)
        C[i][j] = 0.0;

#pragma omp parallel for
shared(A,B,C) private(i,j,k)
for (i = 0; i < M; i++)
    for (k = 0; k < L; k++)
        for (j = 0; j < N; j++)
            C[i][j] +=
                A[i][k] * B[k][j];
```


Intel Developer Tools

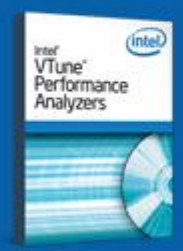
New Developer Resources

- Intel® Threading Building Blocks
  - Template-based product
  - Simplifies threading and code portability
  - Version 1.0
- Intel® Thread Checker & Thread Profiler
  - New version, new features, more platforms
  - Both version 3.0
- Intel XSLT Libraries
  - Adds XML fundamental libraries
  - V1.0 pending

Comprehensive Solutions

Visualization of applications and the system

Architectural Analysis



V8.0, V3.0

Highly optimizing compilers delivering scalable solutions

Introduce Threads

V1.0, V9.1, V5.1

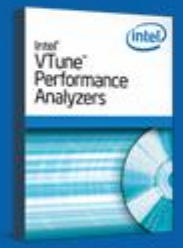
Detect latent programming to address unique challenges

Confidence / Correctness

V3.0

Tune for performance and scalability

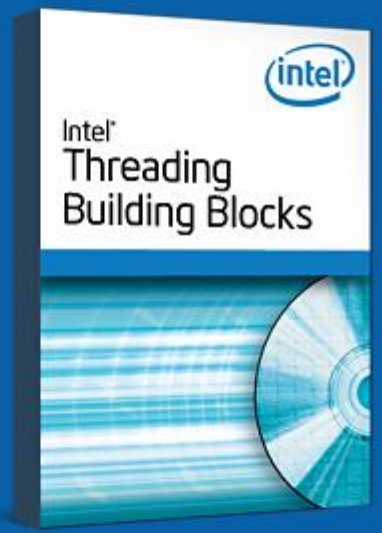
Optimize / Tune



V8.0, V3.0

Intel® Threading Building Blocks
Extending C++ to parallelism using templates

- Designed to work with all C++ compilers, now and without compromises
- Generic, concurrent, thread-safe containers
  - Hash table, vector, queue
- Generic parallel algorithms
  - Parallel for, reduce, scan (prefix), while, pipeline, sort
- Atomic operations
  - Read, write, fetch-and-store, fetch-and-add, compare-and-swap
- Locks
  - Spin, reader-writer, queuing, OS-wrapper
- Scalable Memory Allocation
- Dynamic libraries
  - Support for multiple architectures, platforms, operating systems and compilers.
  - Debug library versions too
- Extensive tutorial and reference documentation



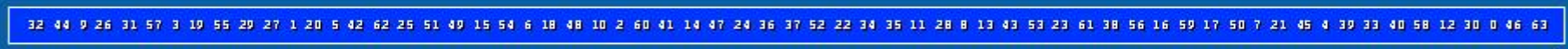
Thread Building Blocks "Qsort" Animation

tbb::parallel_sort (color, color+64);

Quicksort – Step 1

Thread 1 starts with the initial data

THREAD 1

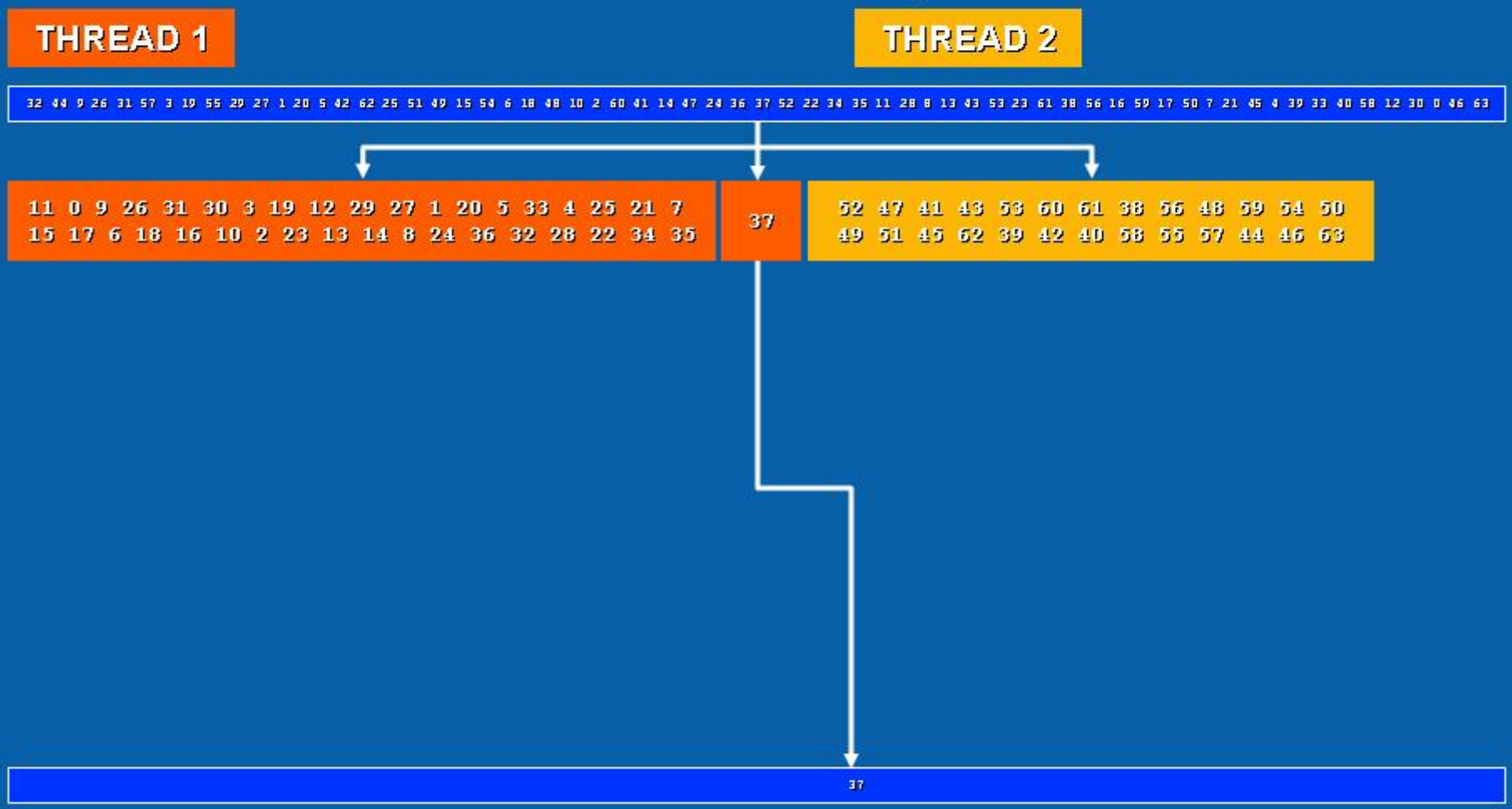


tbb::parallel_sort (color, color+64);

Quicksort – Step 2

Thread 1 partitions/splits the data

Thread 2 gets work by stealing from Thread 1



tbb::parallel_sort (color, color+64);

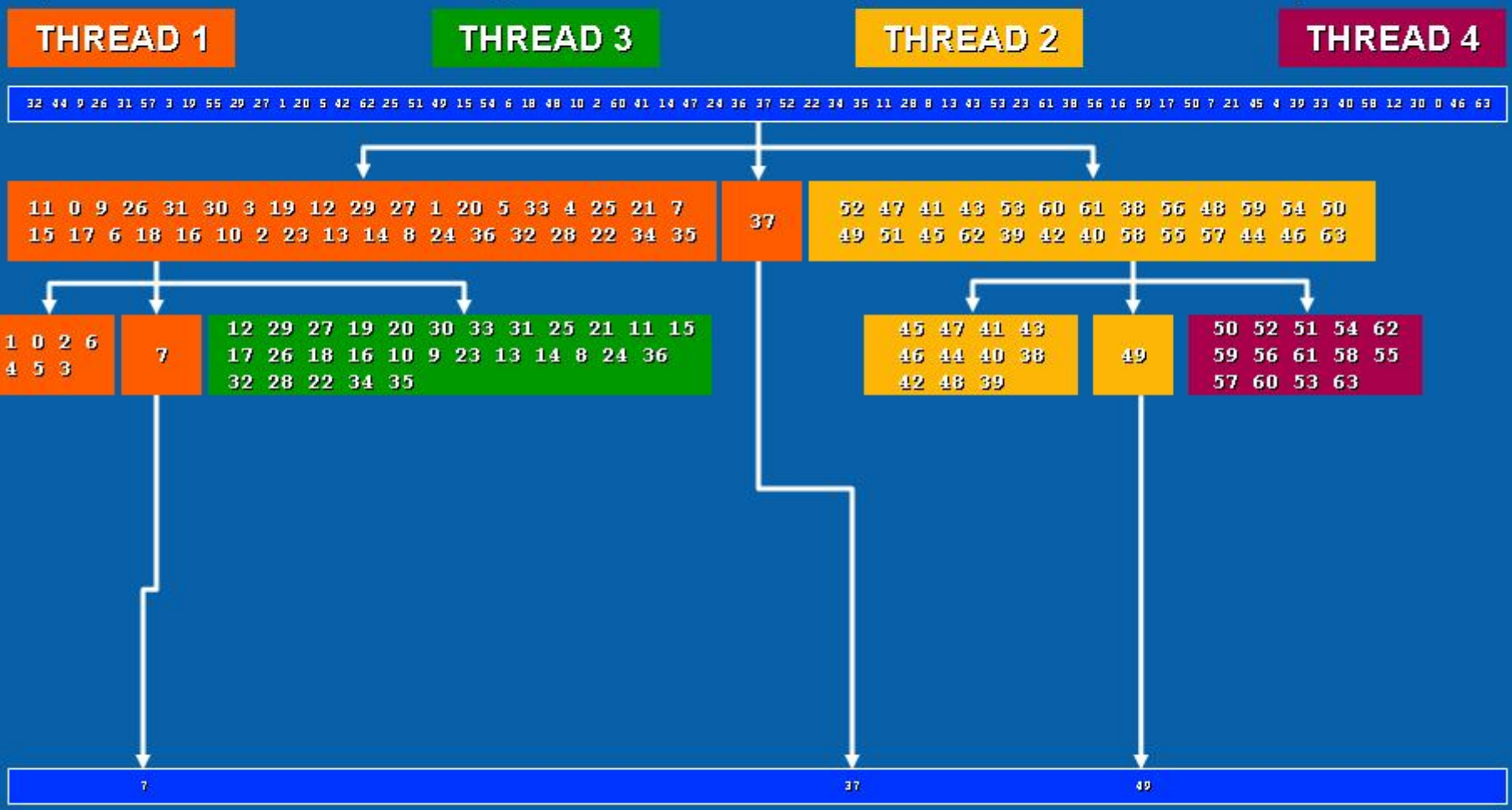
Quicksort – Step 3

Thread 1 partitions/splits a part of its data

Thread 3 gets work by stealing from Thread 1

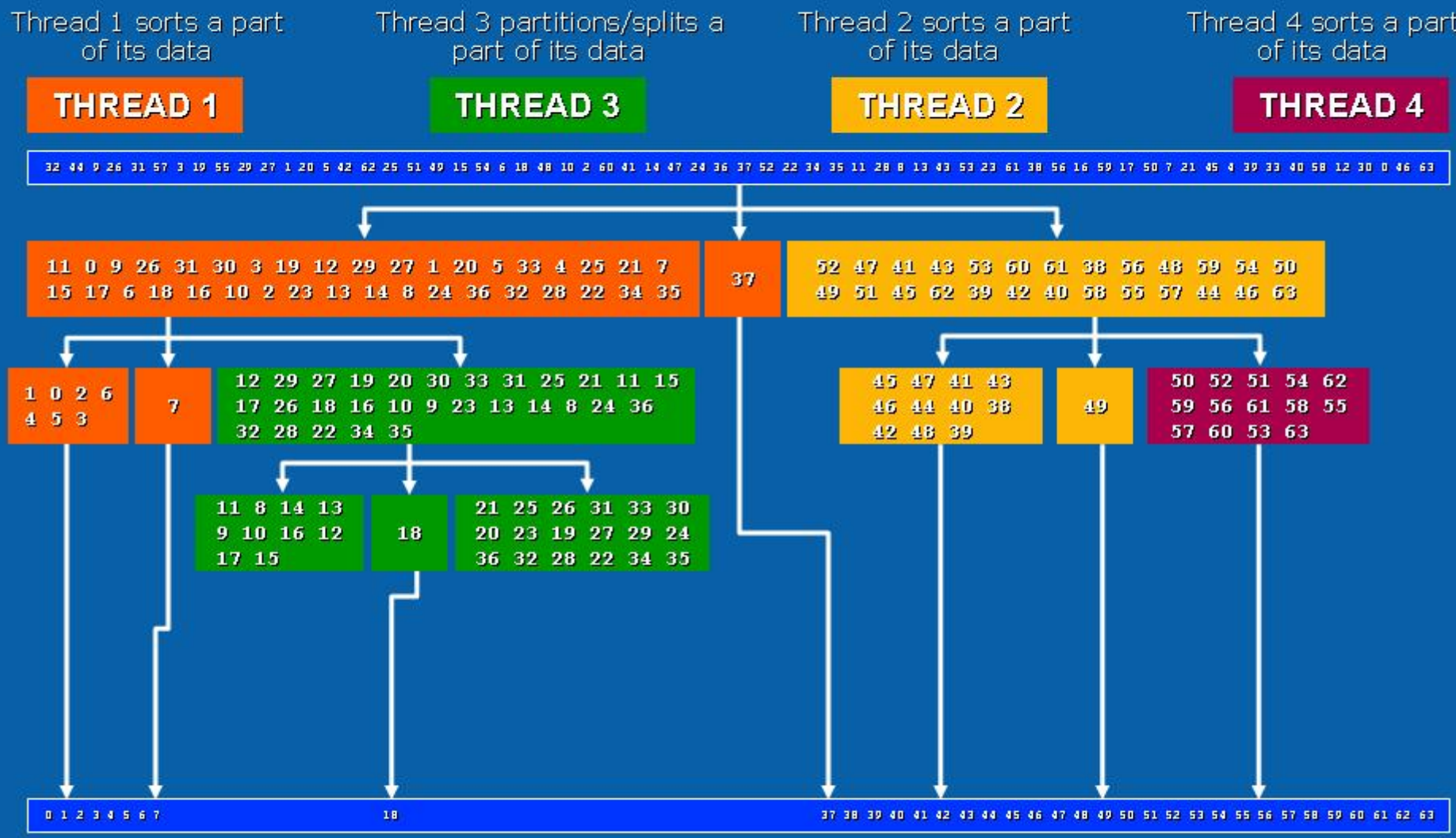
Thread 2 partitions/splits a part of its data

Thread 4 gets work by stealing from Thread 2



tbb::parallel_sort (color, color+64);

Quicksort – Step 4

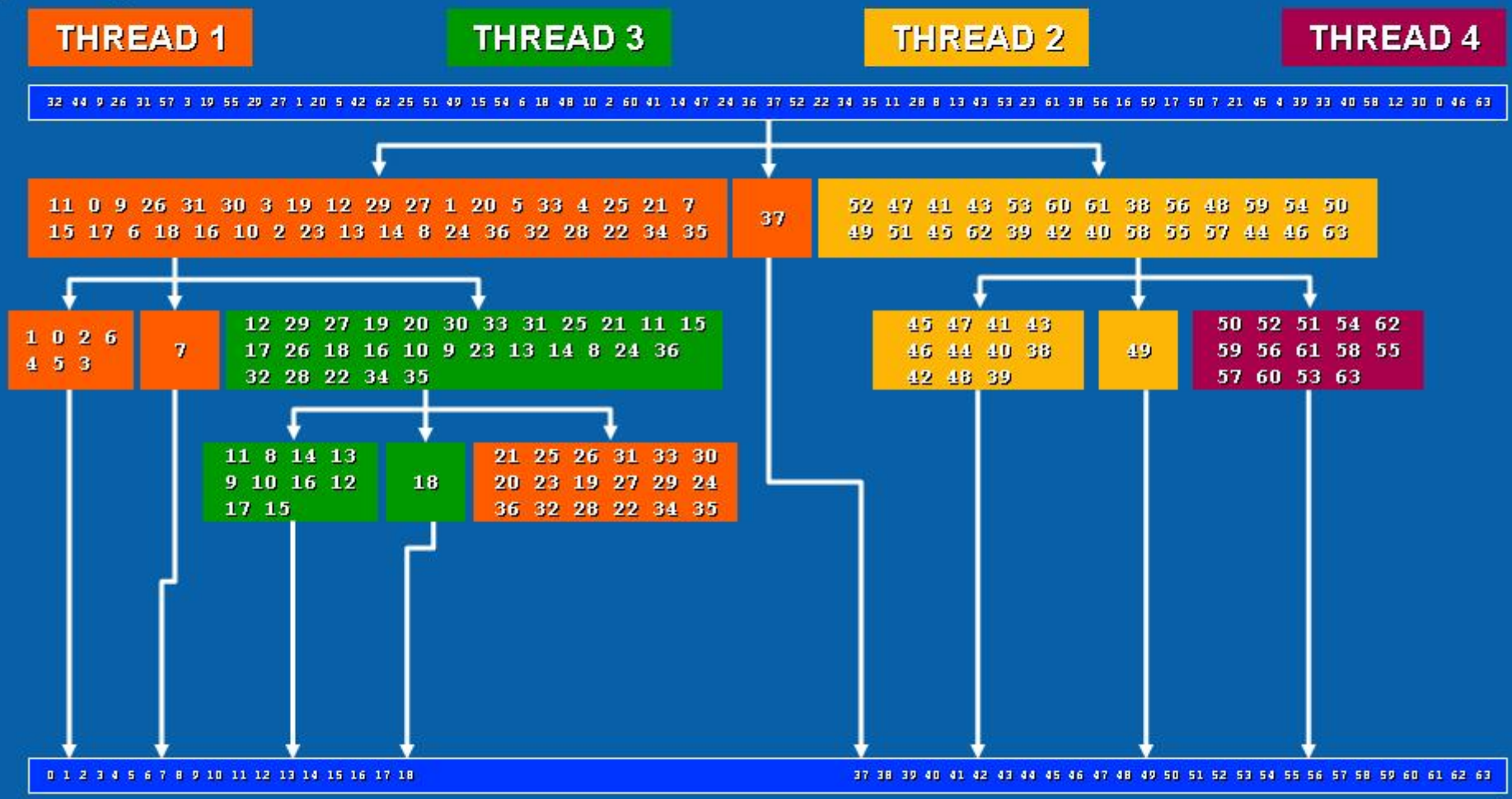


tbb::parallel_sort (color, color+64);

Quicksort – Step 5

Thread 1 gets more work by stealing from Thread 3

Thread 3 sorts a part of its data

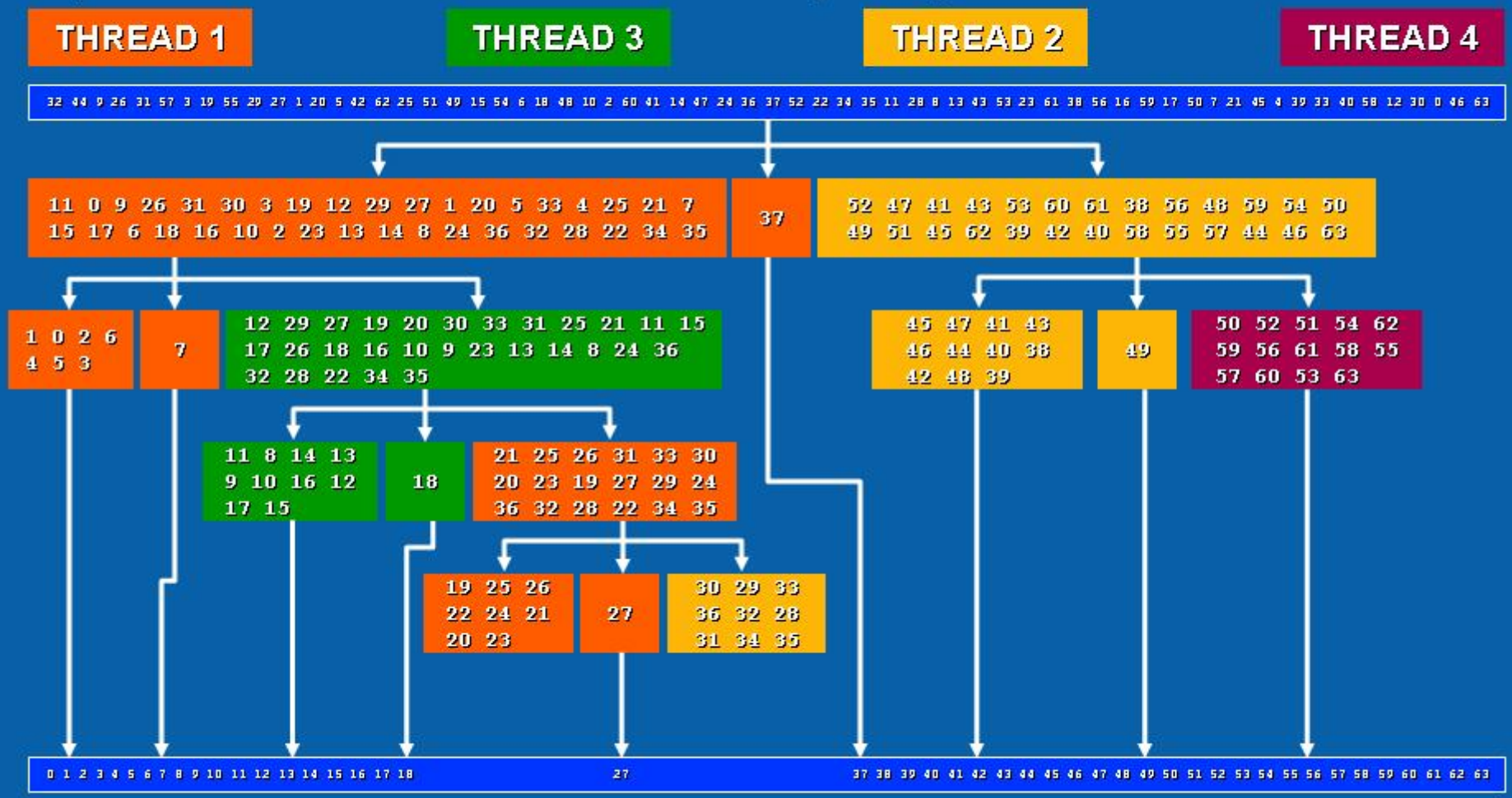


tbb::parallel_sort (color, color+64);

Quicksort – Step 6

Thread 1 partitions/splits a part of its data

Thread 2 gets more work by stealing from Thread 1

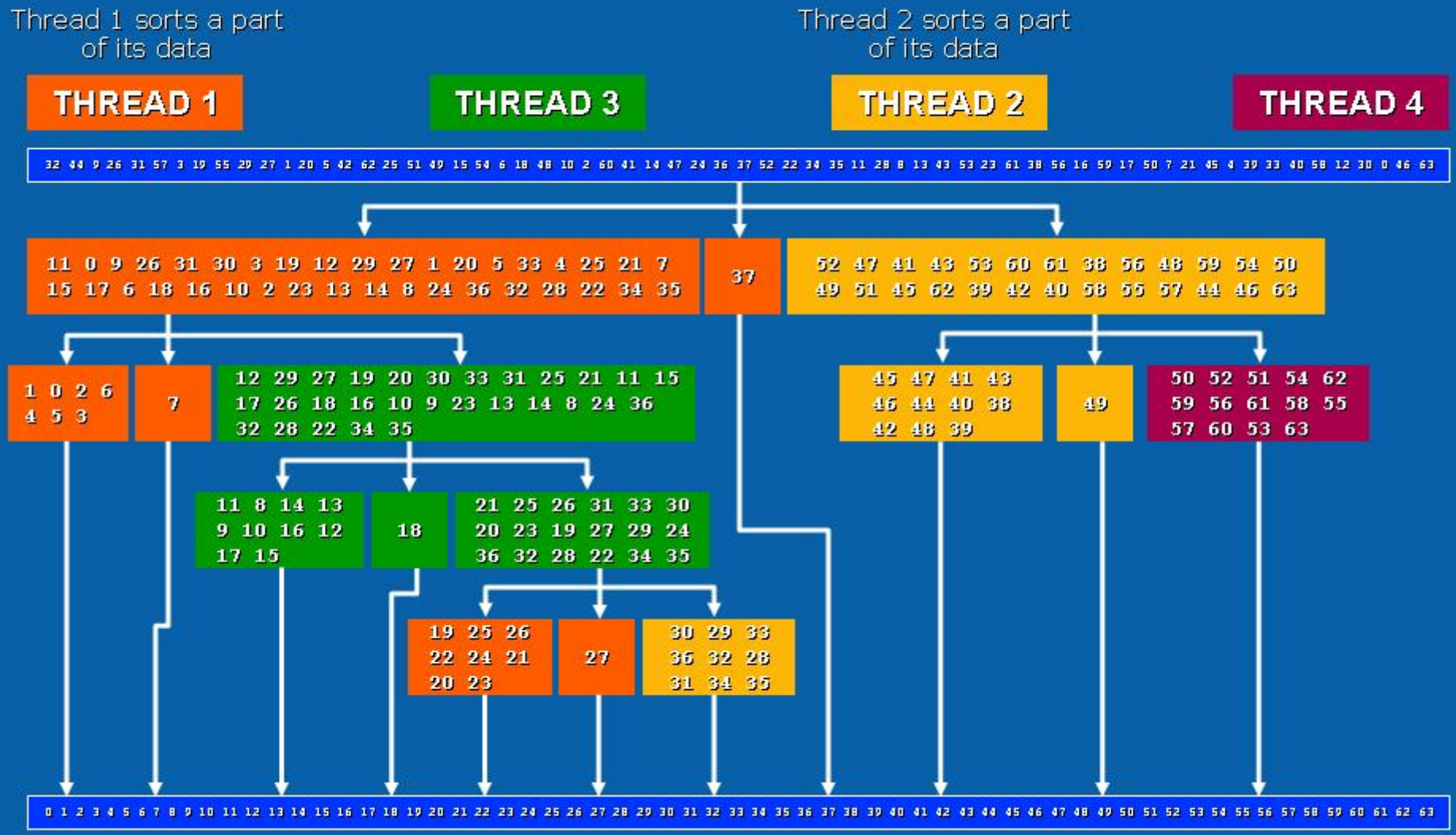


Intel Developer FORUM

intel

tbb::parallel_sort (color, color+64);

Quicksort – Step 7



ALL DONE

Intel® Threading Building Blocks Demonstration

Intel® Thread Building Blocks
A C++ template-based library

- Demonstrates "Tacheon", a 2-D raytracer/renderer parallelized by using Intel® TBB
- View comparison of serial vs. Intel® TBB threading for performance and scalability
- Intel® TBB makes it easier to write multi-core applications that perform and scale well
  - Almost 2x better scalable performance than comparable-effort native threads coding
  - Intel® TBB threaded code takes significantly less coding efforts to achieve performance gains


Intel® C++ Compiler
Demonstration

Intel® C++ Compiler: Bringing Maximum Performance to the Mac OS* X

- Demonstrates "Persistence of Vision" Raytracer; Open source, 3-D graphical rendering tool
  - One binary built with Intel® C++ Compiler for Mac OS* (ICC)
  - One built with GNU* GCC using the best optimization options
- View a side-by-side comparison of the time to render a benchmark image
- ~30% performance advantage with ICC


XML Processing and Security
New Libraries from Intel

Intel XML Translation, Schema Validation And WS-Security Libraries

Intel XSLT 1.0 Library

- XSLT 1.0 compliance, drop-in replacement for JAXP based XSLT processors
- Support for EXSLT extension functions and custom java extension

Intel XML Schema 1.0 Library

- W3C XML Schema 1.0 Support
- Compiled XSDs – thread-safe & re-entrant
- Stored or streamed data input supported

Intel WS-Security Library 1.0

- Policy based and high-performance
- WS-Security 1.0 Signature Generation/Verification, Encryption/Decryption

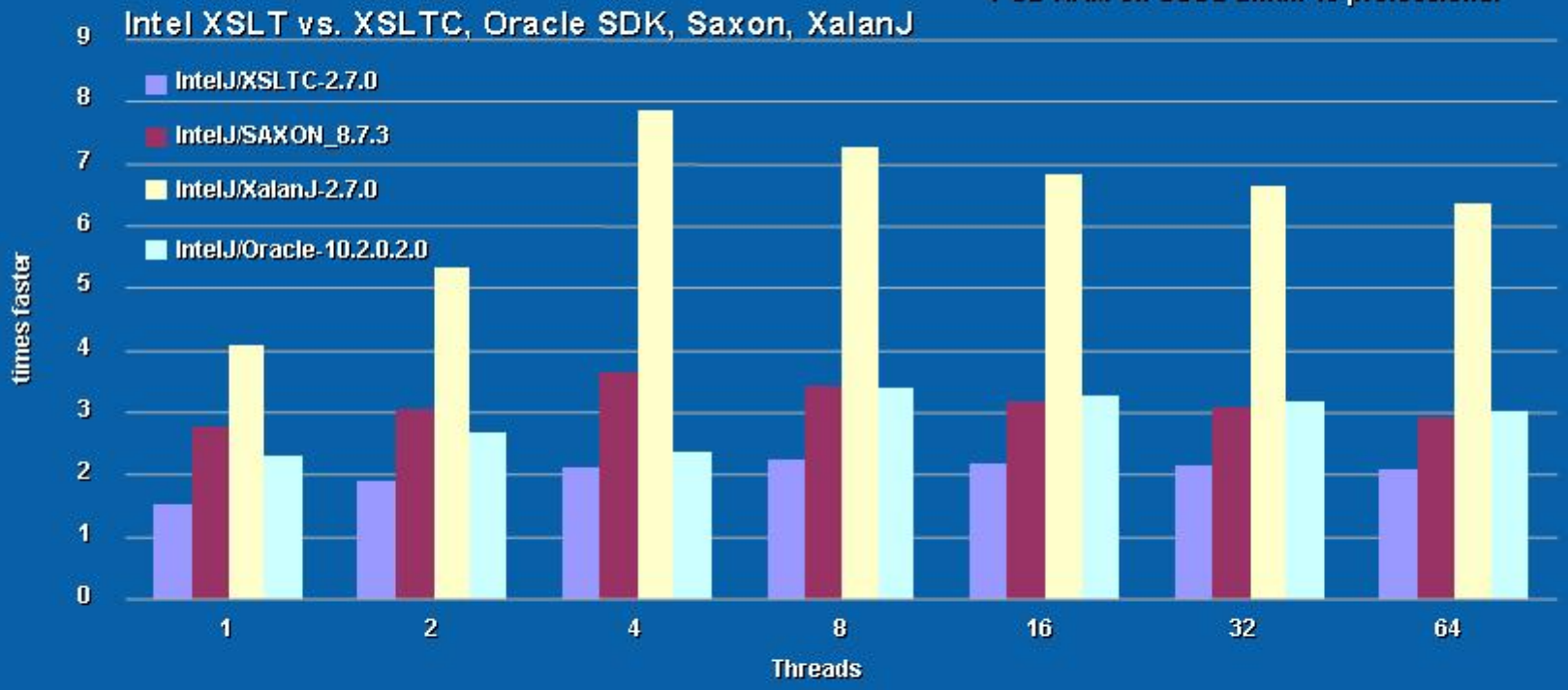
All Libraries:

- Outperform other standards based implementations 2-5x across workloads
- Thread safe and ready to scale with the number of cores
- IA32 and Intel 64 Platform Support

Growing Standards Based SOA Development Tools for Intel Platforms

XSLT 1.0 Library Benchmark Results

2x Dual Core 2.6 GHz Woodcrest with 2 MB cache,
4 GB RAM on SUSE Linux 10 professional



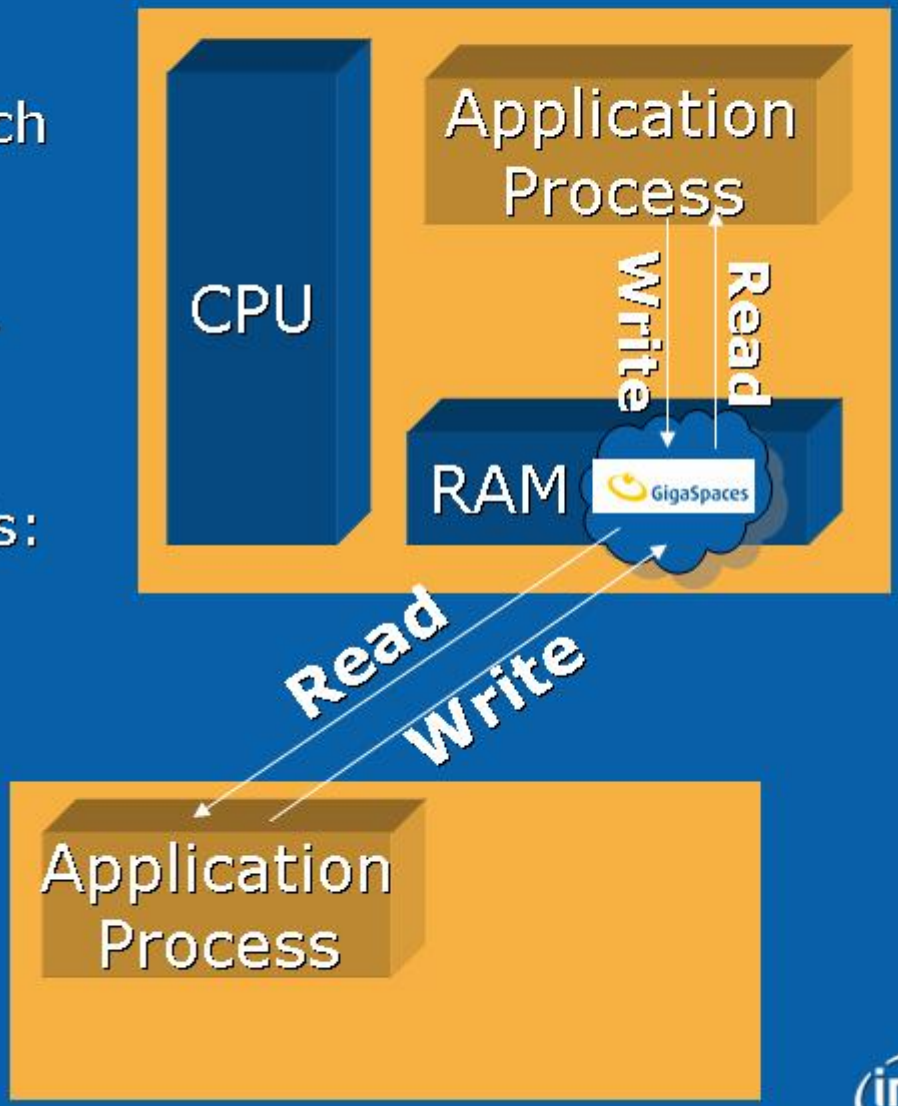
Ready for multi-core platforms and 2-5X improvement vs. other solutions

Geva Perry

Executive Vice President
GigaSpaces

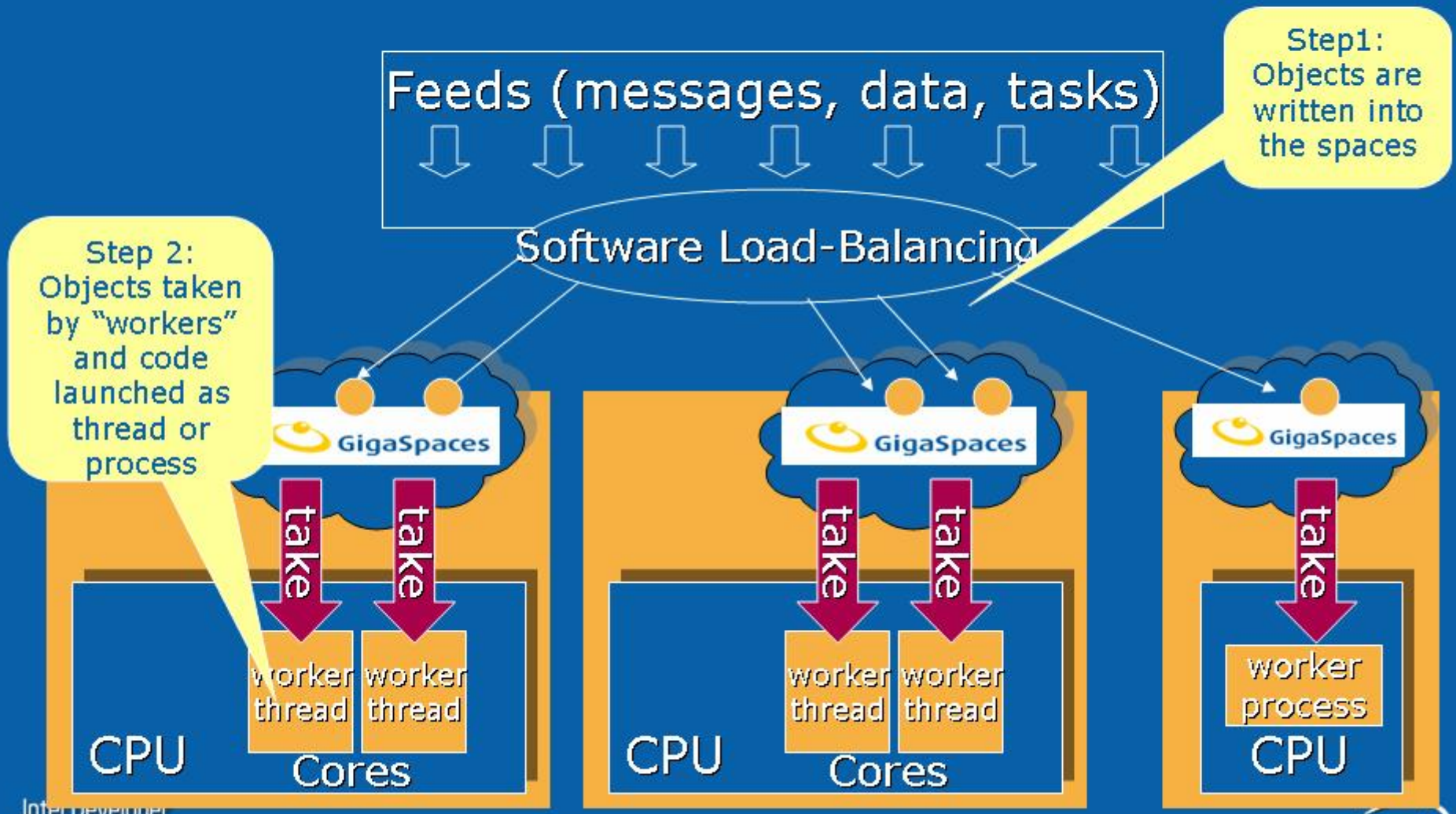
What are Spaces?

- Virtual shared memory address "spaces": from which local and remote processes can read and write objects
- Based on the "Linda" Tuple-Space model1 and the Jini/JavaSpaces open spec
- Interfaces via Standard APIs:
  - JavaSpaces
  - JDBC
  - JMS
  - JCache
  - C++ / C# and POJO

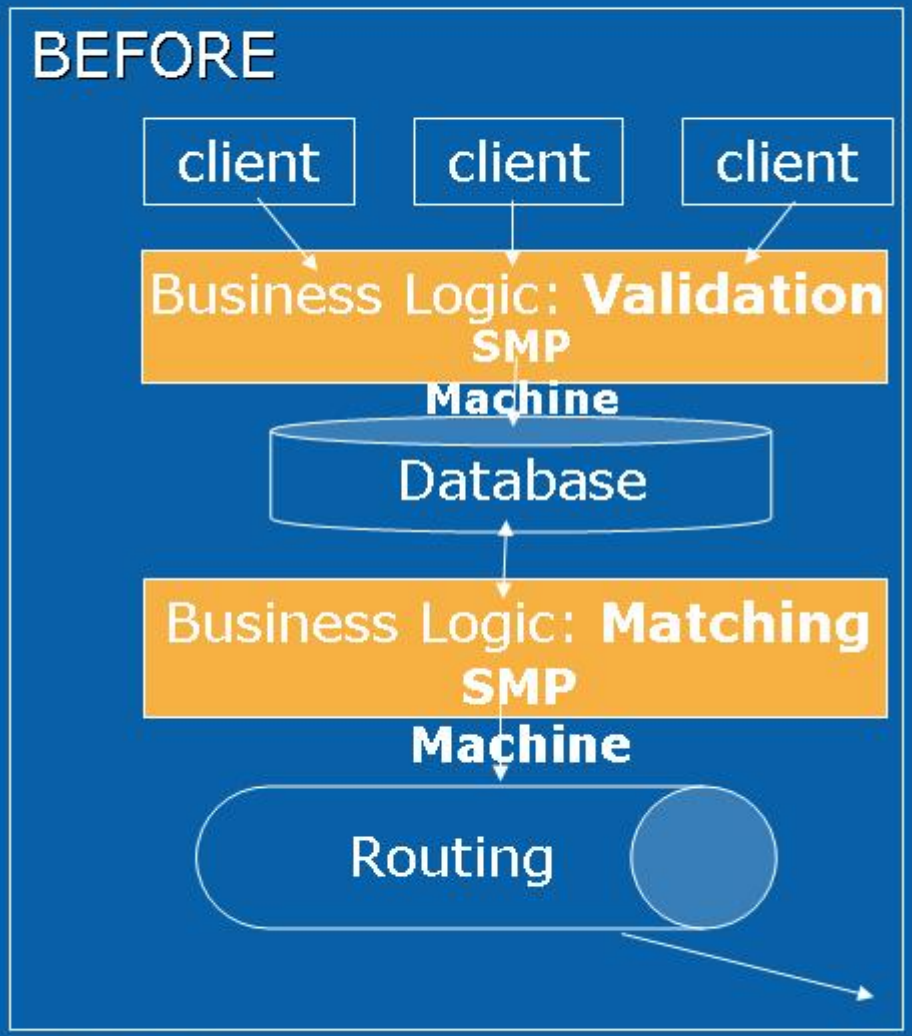


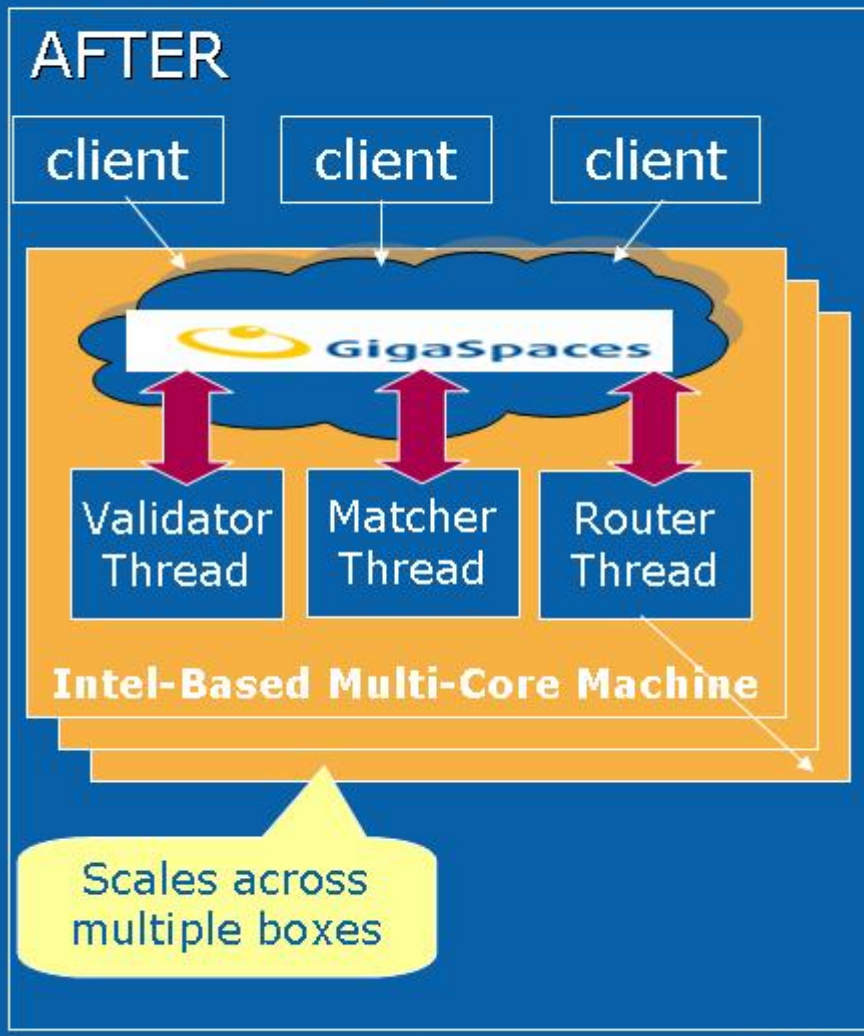
1. N. Carriero, D. Gelernter, 13th ACM Symposium 1986

Parallel Processing Using Spaces



Global Bank – FX Trading Application





Customer Momentum

Financial Services

- Societe Generale
- Deutsche Bank
- Dow Jones
- Monte Paschi Bank (Italy)
- Webster Bank
- Chicago Mercantile Exchange

Telco

- Virgin Mobile
- Nortel Networks
- Ericsson
- Hutchison
- Mitel Networks

Other

- U.S. Government (security agency)
- IBM Tivoli
- Gallup Organization

Benefits of Space-Based Architecture

- No need to code for parallelization or threading
- Simplicity: No special APIs
- Take advantage of any underlying hardware capabilities transparently:
  - Multi-core, Multi-CPU, SMP or any combination thereof
- Linear Scalability due to true parallelization at every tier
- High performance due to utilization of memory and co-location of application processes/threads and related data
- Fault tolerance due to transparent object replication and synchronization in the virtual shared memory


Summary

Summary

- Intel continues to invest in eco system for tools and libraries to make threading easier
- Significant University and Training program to long-term impact programming using multiple cores
- ISVs and End Users are now significantly engaged and focused on apps that use multiple cores to bring to market innovation and scaling solutions

For more information on developer training
www.intel.com/software/college


intel®
Leap ahead™