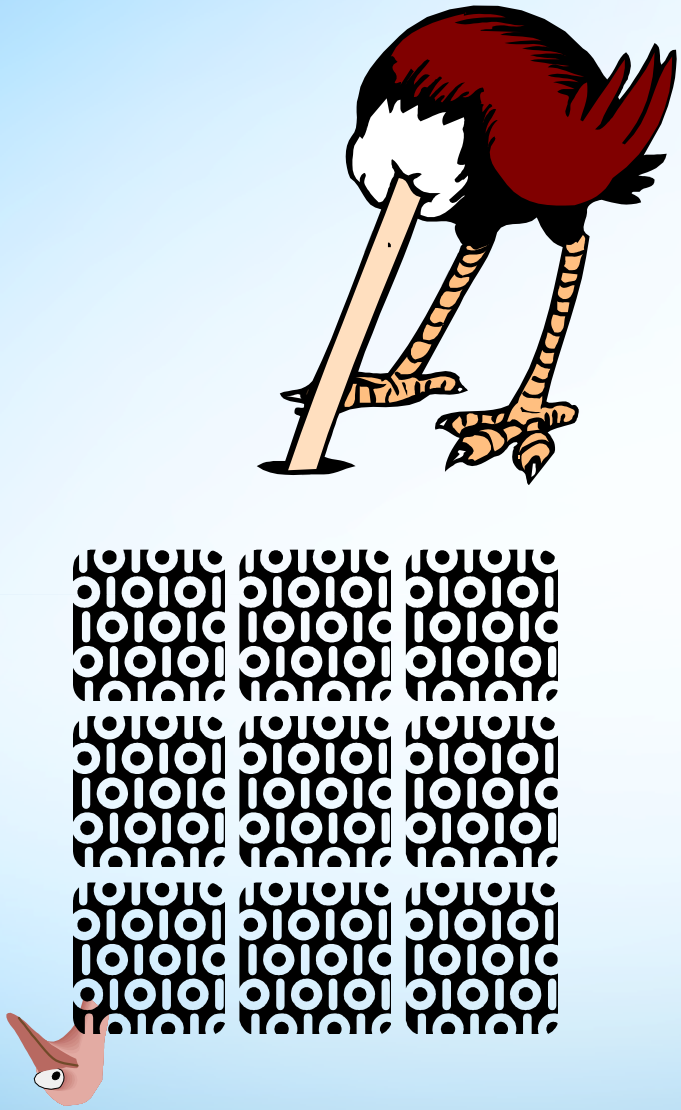


Multicore is dead; Long live Multicore!

Stephen Blair-Chappell
Intel Compiler Labs

Abstract

(Ten) reasons why programming for multicore should be avoided. In this tongue-in-cheek session we take a head-in-the-sand approach to multicore programming. We present a number of anecdotal reasons why you should never program for multicore. Includes a rapid examination of several case studies.

Warning: Content may be subject to exaggeration and hyperbola - after attending this session you may never write a parallel program again.

1. Multicore is just a fad!

The Problem – Technology Shelf Life

Growth in Cores - A well rehearsed story

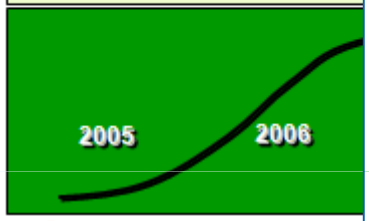
Multi-core ramp accelerates in 2006

Intel® dual-core volume exiting 2006**

Desktop Performance	>70% Dual-Core
Server	>85% Dual-Core
Mobile Performance	>70% Dual-Core

**Data is run-rate exiting the year. All products and dates are preliminary and subject to change without notice.

Intel single core to multi-core shi



By 2007, the majority o

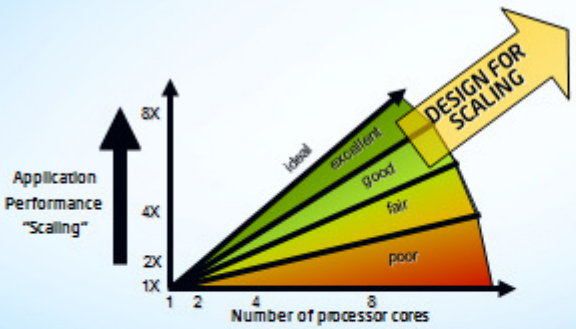
Cores, Cores & more Cores

- Today
  - Dual core
  - Quad core
  - Multi-socket solutions
- The Future
  - 6 & 8 cores
  - many-core
- R & D
  - 80 cores …

Performance com
multi-core gains –

The software industry goes parallel

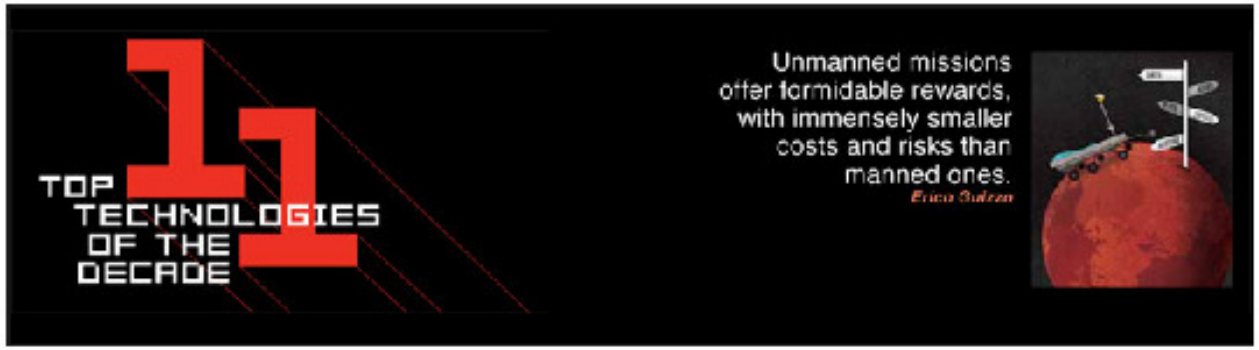
Increase application performance & scalability



The Challenge

- Serial applications can not take advantage of multicore platforms.
- Number of processor cores is increasing.
- Remaining competitive requires parallelizing serial code or creating new parallel applications.

1. Multicore is just a fad!



NO. 1 **Smartphones: The Pocketable PC**
Is your phone smarter than a fifth grader?

NO. 2 **Social Networking: Friended**
Bandwidth, digital cameras, and a hunger for connectedness have created a virtual dinner party

NO. 3 **Voice Over IP: Setting Phone Service Free**
How Ma Bell's cash cow became a free software app

NO. 4 **LED Lighting: Blue + Yellow = White**
Giving LEDs the blues was the key to replacing the incandescent bulb

NO. 5 **Multicore CPUs: Processor Proliferation**
From multicore to many-core to hard-to-describe-in-a-single-word cores

NO. 6 **Cloud Computing: It's Always Sunny in the Cloud**
Cloud computing puts your desktop wherever you want it

NO. 7 **Drone Aircraft: How the Drones Got Their Stingers**
Unmanned aerial vehicles come of age

NO. 8 **Planetary Rovers: Are We Alone?**
Planetary rovers attempt to answer the most profound question in science

NO. 9 **Flexible AC Transmission: The FACTS Machine**
Flexible power electronics will make the smart grid smart

NO. 10 **Digital Photography: The Power of Pixels**
Digital photography changed not only how we take pictures but also how we communicate

NO. 11 **Class-D Audio: The Power and the Glory**
A quiet revolution is transforming audio electronics

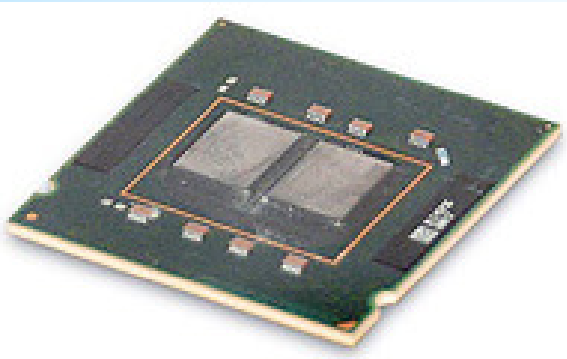
Next-to-the-Best Technologies of 2000-2010
These innovations just barely missed the cut for our Top 11 list

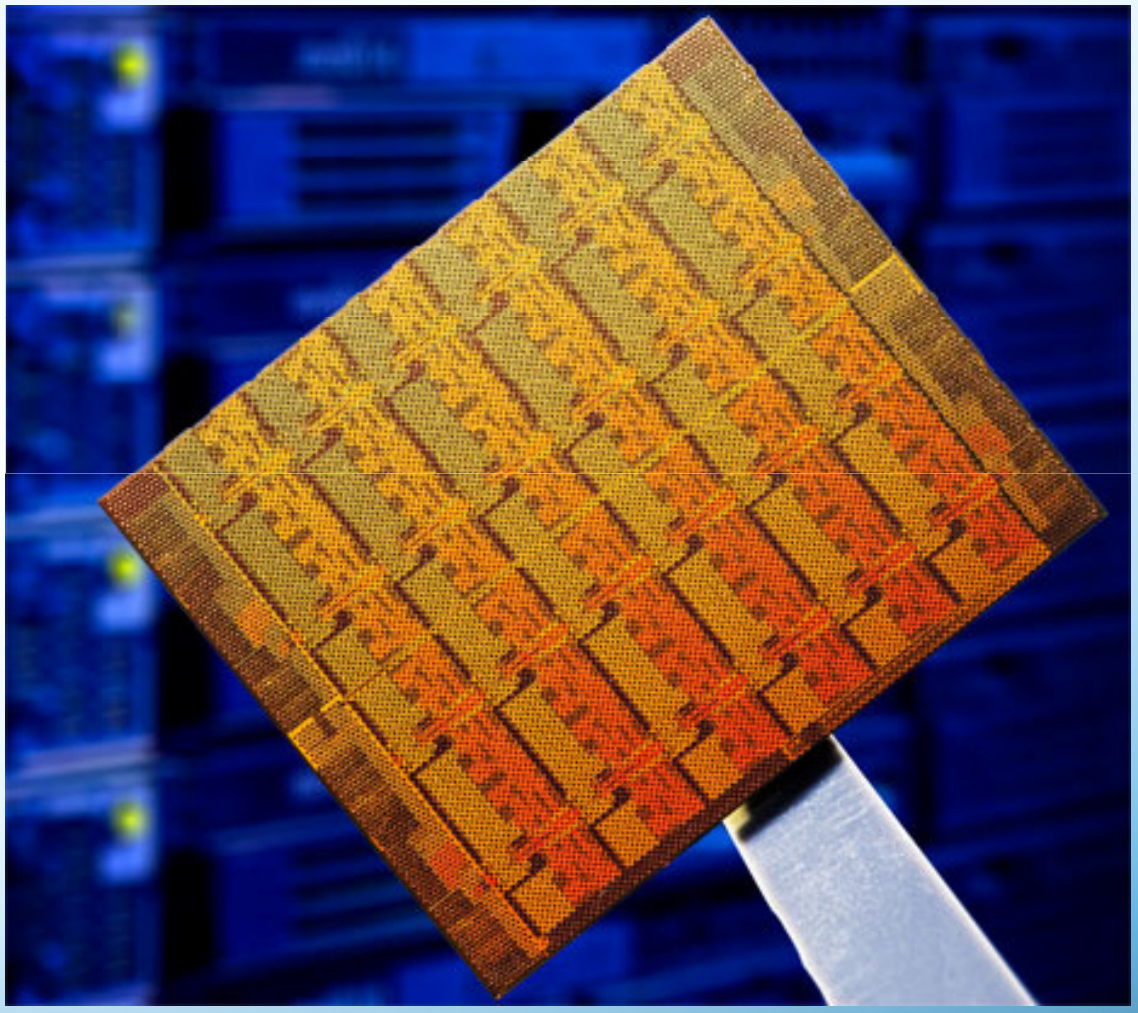
ieee

http://spectrum.ieee.org/semiconductors/processors/multicore-cpus-processor-proliferation/0

1. Multicore is just a fad!

Silicon





1. Multicore is just a fad! ##

1. Multicore is just a fad!

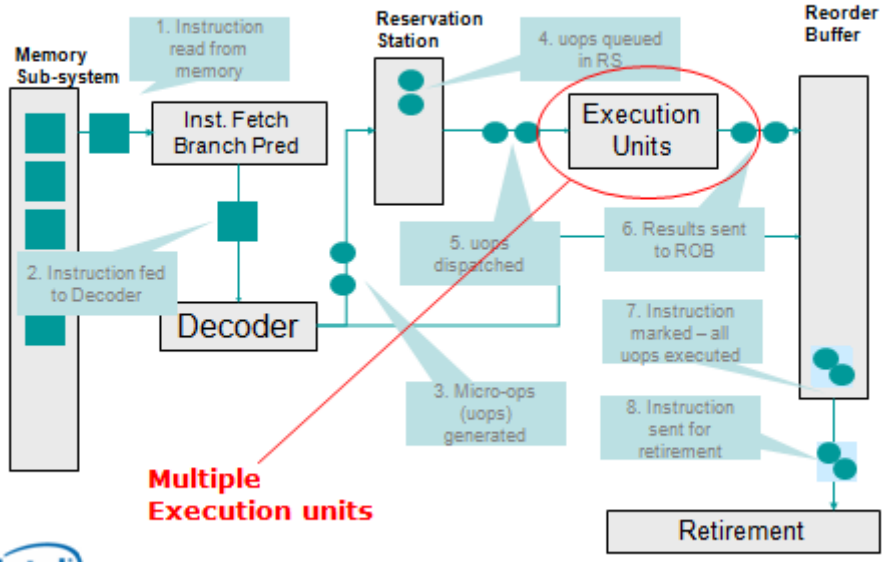
False

2. My Program will run just the same without any effort!

The Problem – No problems!

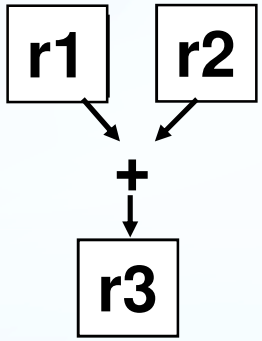
Vector Processing

The life of a program instruction



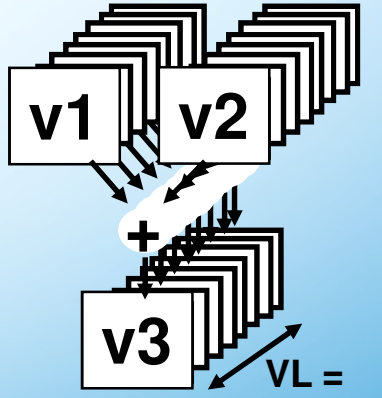
3. The CPU automatically makes things parallel – so I don't need to.

Scalar Processing




```
add.d r3, r1, r2
```

Vector Processing




```
addvec.d v3, v1, v2
```

2. My Program will run just the same without any effort!

Layers of Optimisation

Optimisation	Implementation
Heuristics	Direct Sound
Libraries	IPP
Soft\Hard RT	Win32\RTX
Code Generation	Intel Compiler
Multicore	Core 2 \ i7
ILP	Execution Units
SIMD	SSE\AVX

2. My Program will run just the same without any effort! ##

2. My Program will run just the same without any effort!

✓ True

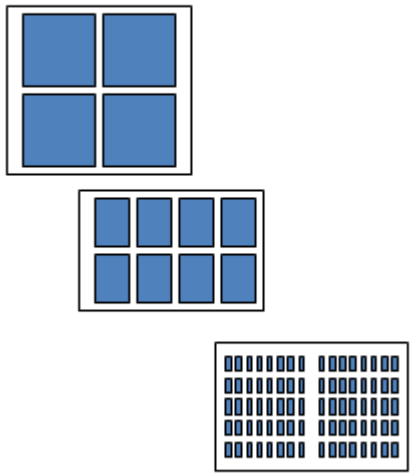
3. The CPU automatically makes things parallel – so I don't need to.

The Problem – Wrong Information

Cores

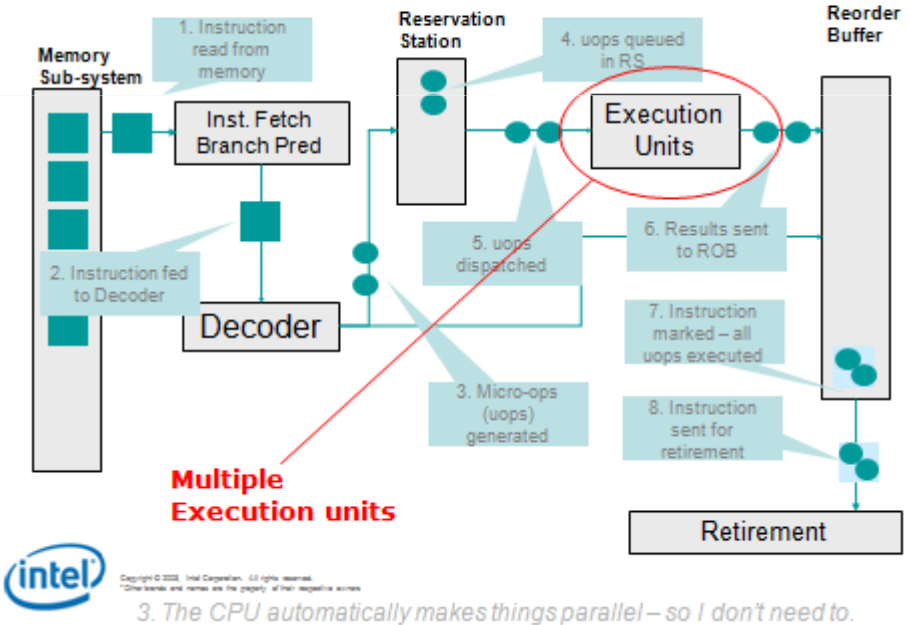
Cores, Cores & more Cores

- Today
  - Dual core
  - Quad core
  - Multi-socket solutions
- The Future
  - 6 & 8 cores
  - many-core
- R & D
  - 80 cores …



Performance comes from multi-core gains – but through parallelism

The life of a program instruction



3. The CPU automatically makes things parallel – so I don't need to.

3. The CPU automatically makes things parallel – so I don't need to.

The New Programming Challenge

"Everyone's happy—except perhaps for the programmers, who must now write code with threads of instructions that must be executed together—in pairs, quartets, or even larger groupings."

Samuel K. Moore / January 2011

3. The CPU automatically makes things parallel – so I don't need to. ##

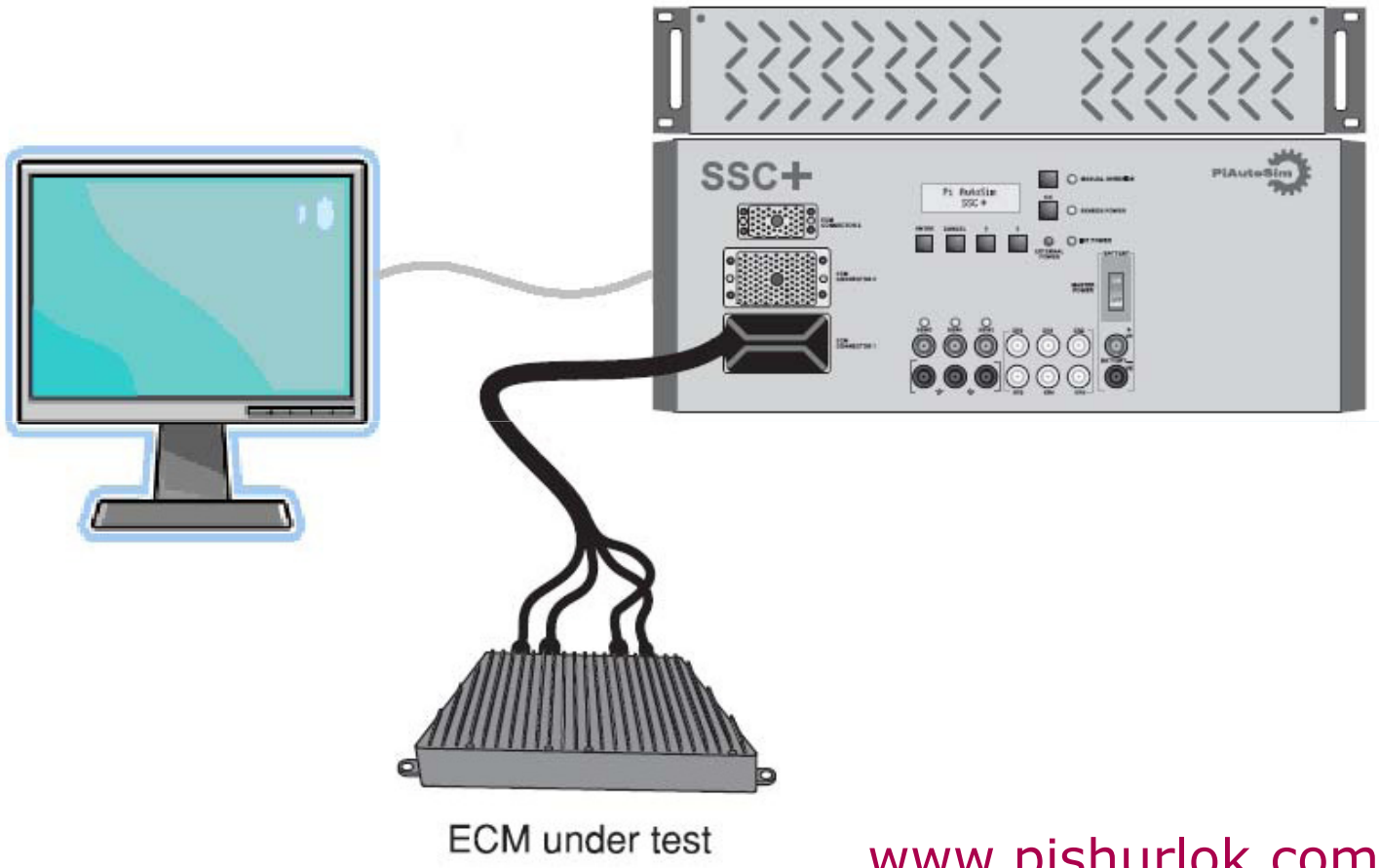
3. The CPU automatically makes things parallel – so I don't need to.

False

Case Study 1

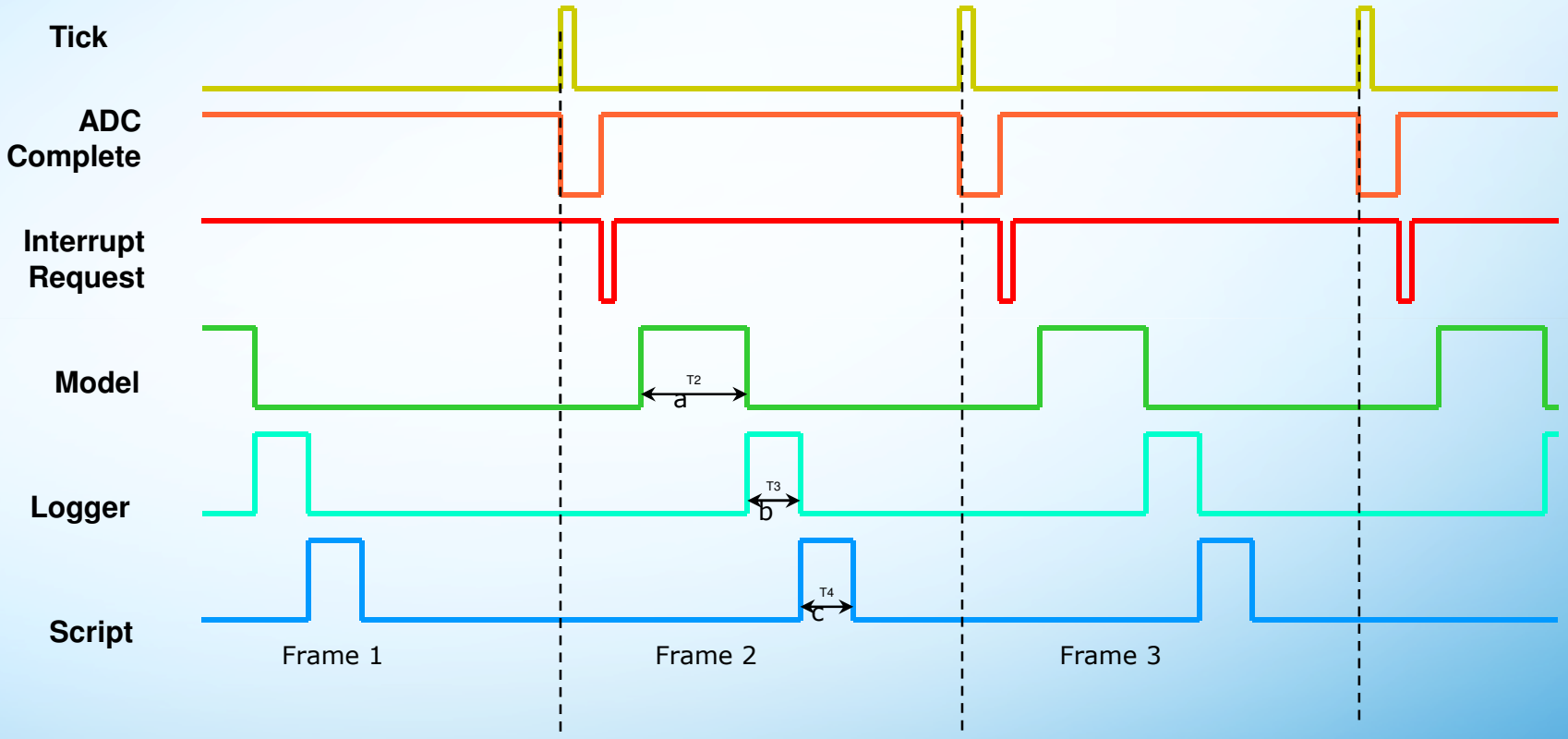
An Engine Simulator

The Simulation Environment

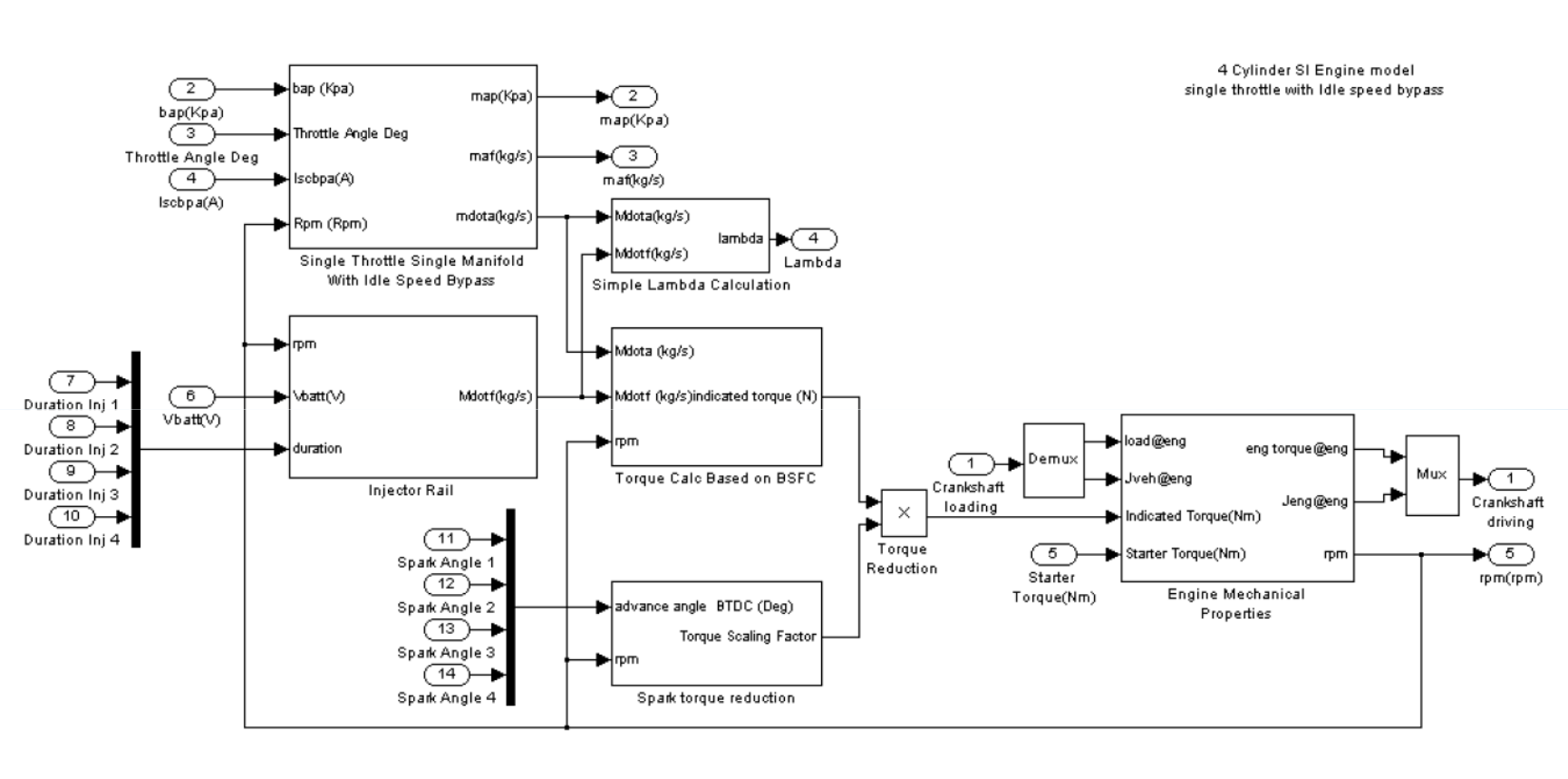


www.pishurlok.com

The Simulation Frames

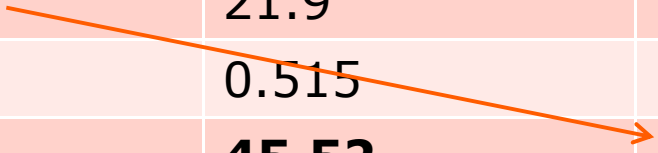


Matlab design of the Engine Simulator



Results on 100k loop simulation

CPU	No Auto-Vectorisation	With Auto-Vectorisation	Speedup
P4	39.344	21.9	**1.80**
Core 2	5.546	0.515	**10.77**
Speedup	**7.09**	**45.52**	**76**



Vtune confirms reason for Speedup

CPU EVENT	Without Vect	With Vect
CPU_CLK_UNHALTED.CORE	16,641,000,448	1,548,000,000
INST_RETIRED.ANY	3,308,999,936	1,395,000,064
X87_OPS_RETIRED.ANY	250,000,000	0
SIMD_INST_RETIRED	0	763,000,000

Full paper available here:
http://edc.intel.com/Link.aspx?id=1045

Reason for not parallelising

- OS did not support threads
  - Old RTOS
  - Incompatible runtime
- Already gained more than enough performance improvements

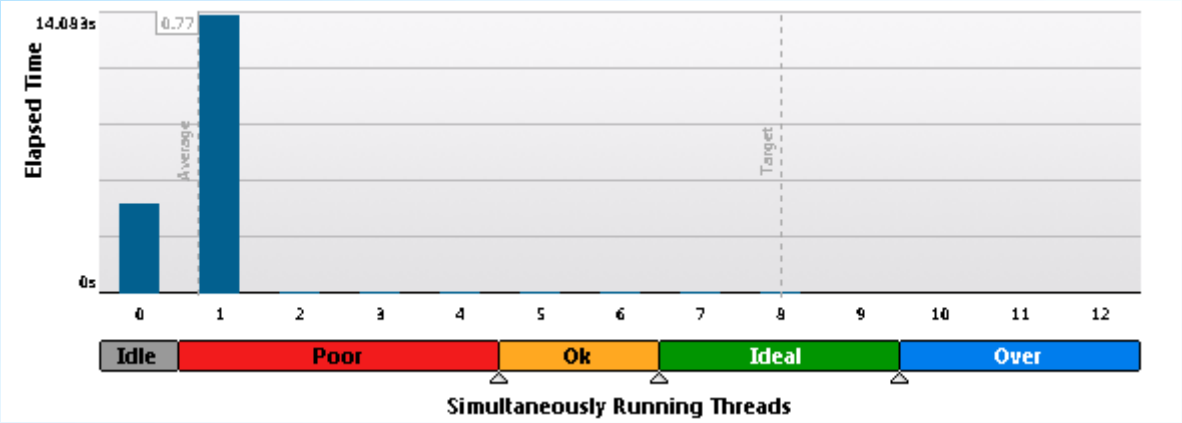
4. Parallel programming makes applications run slower!

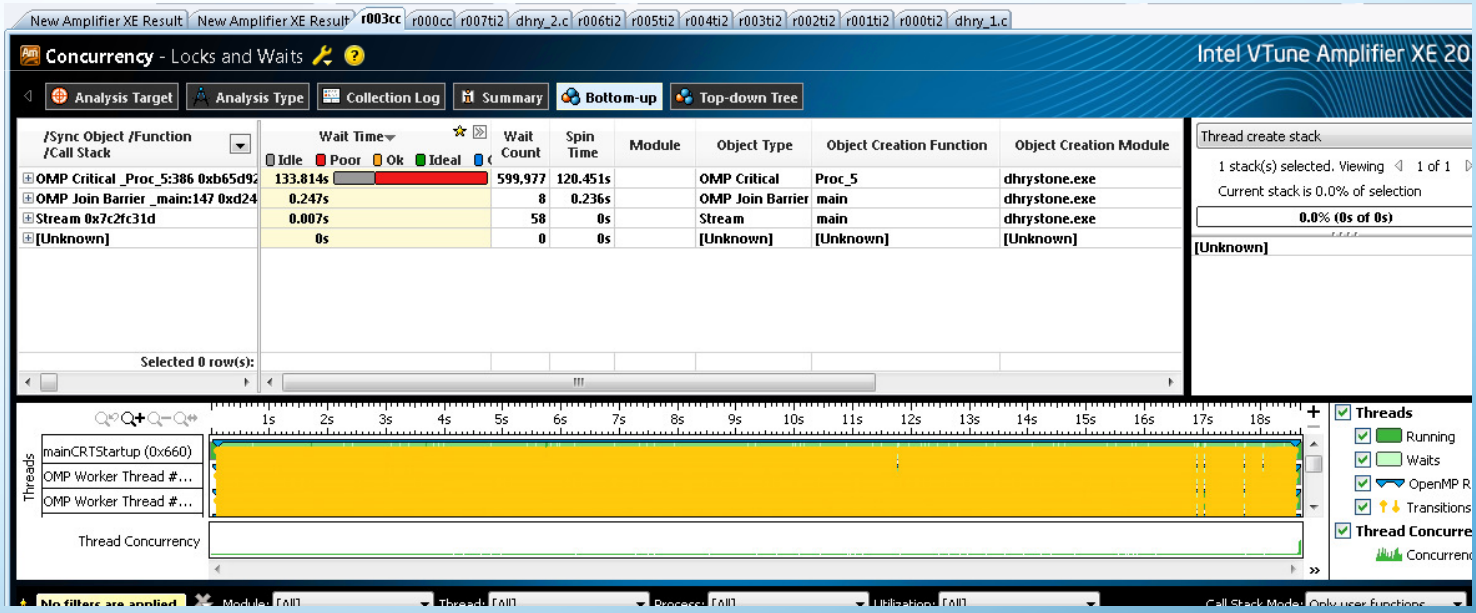
The Issue – Granularity & Overhead

Granularity

4. Parallel programming makes applications run slower!

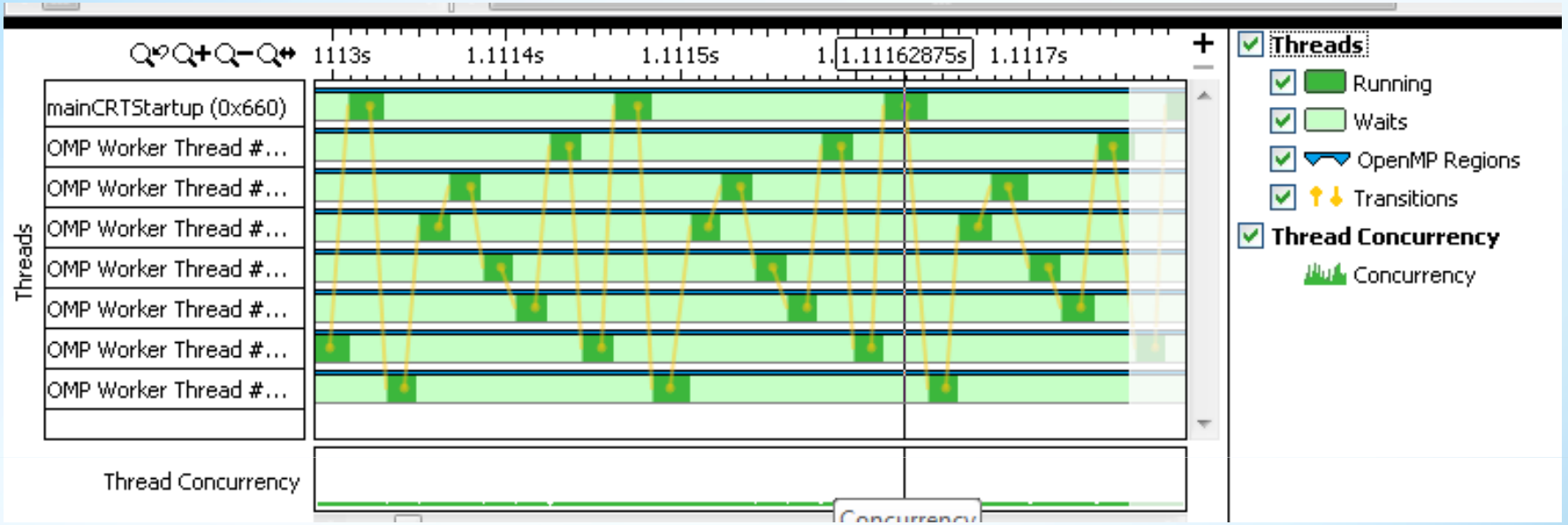
Overhead

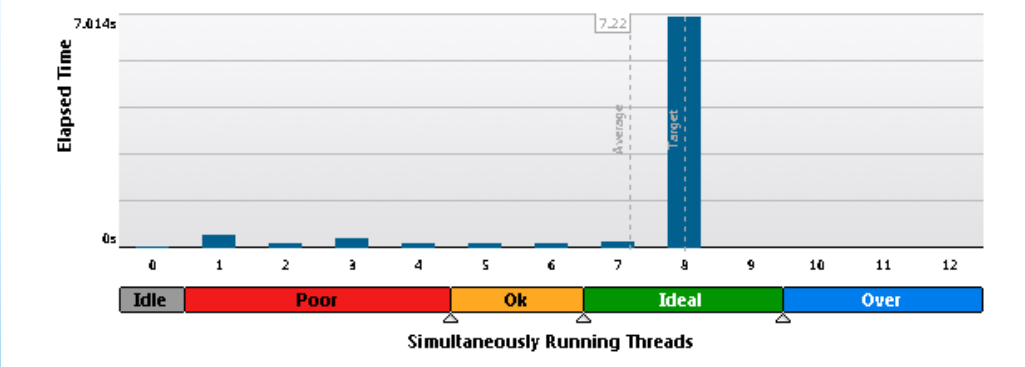


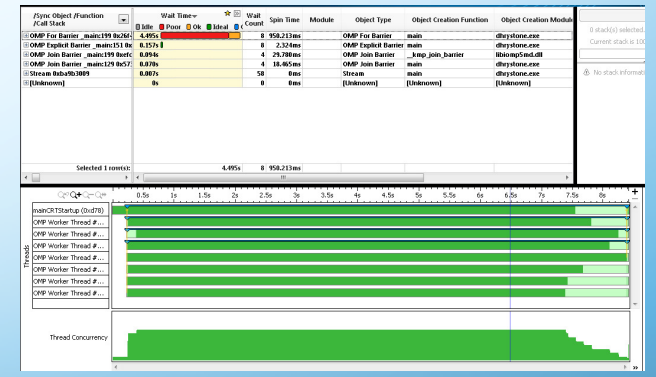


4. Parallel programming makes applications run slower!

Overhead







4. Parallel programming makes applications run slower! ##

4. Parallel programming makes applications run slower!

False

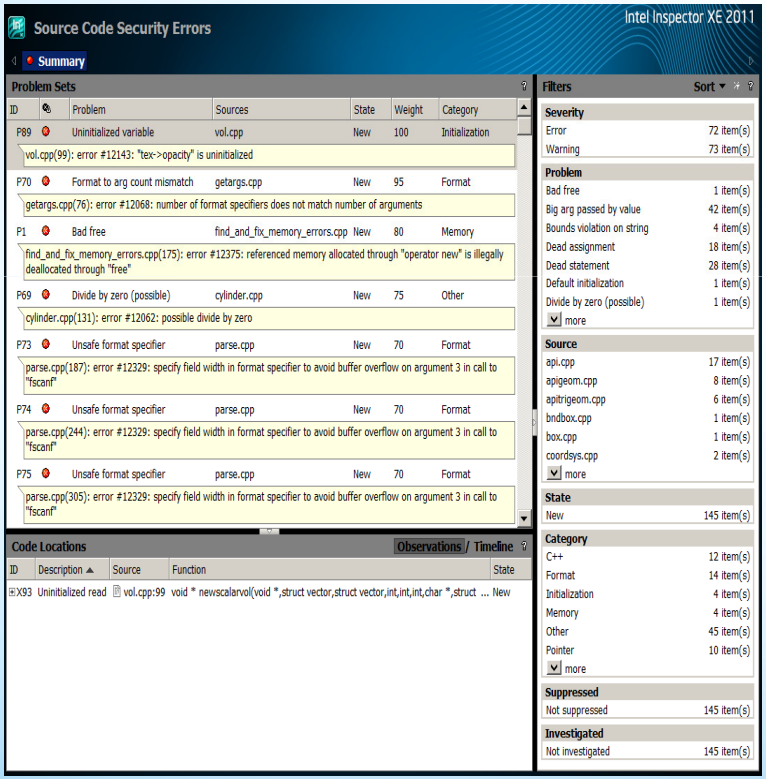
✓ *True*

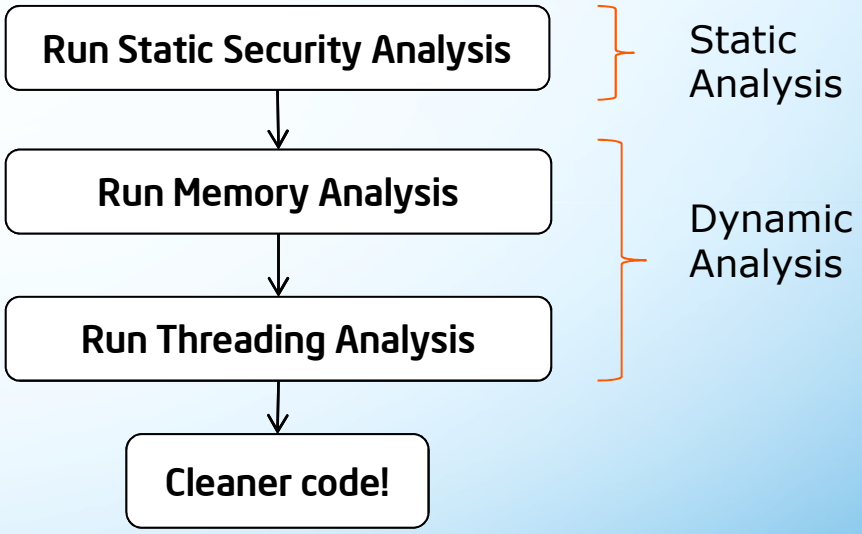
5. No parallelism means no Errors. QED!

The Issue – Data Races and Deadlocks

Data Races and Deadlocks

- The Issue - Dataraces and Deadlocks





5. No parallelism means no Errors. QED! ##

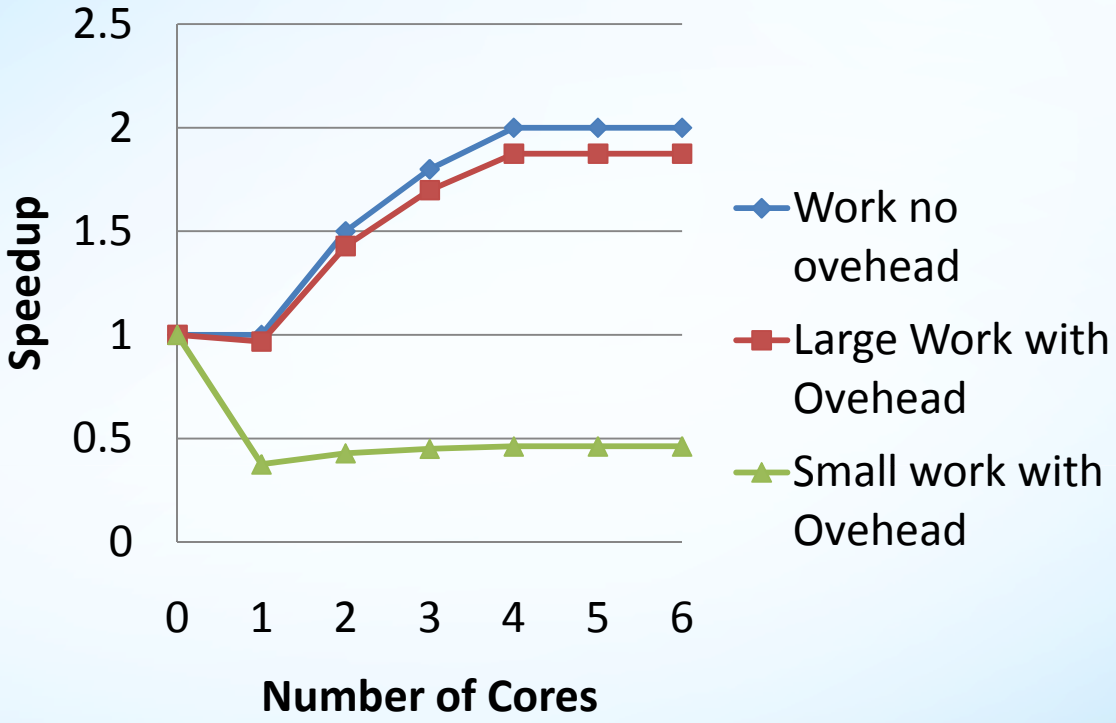
5. No parallelism means no Errors. QED!

✓ True

6. Multicore programs don't get faster on newer generations of CPU

The Issue – Scalability and Future Proofing

Scalability & Future Proofing



6. Multicore programs don't get faster on newer generations of CPU ##

6. Multicore programs don't get faster on newer generations of CPU

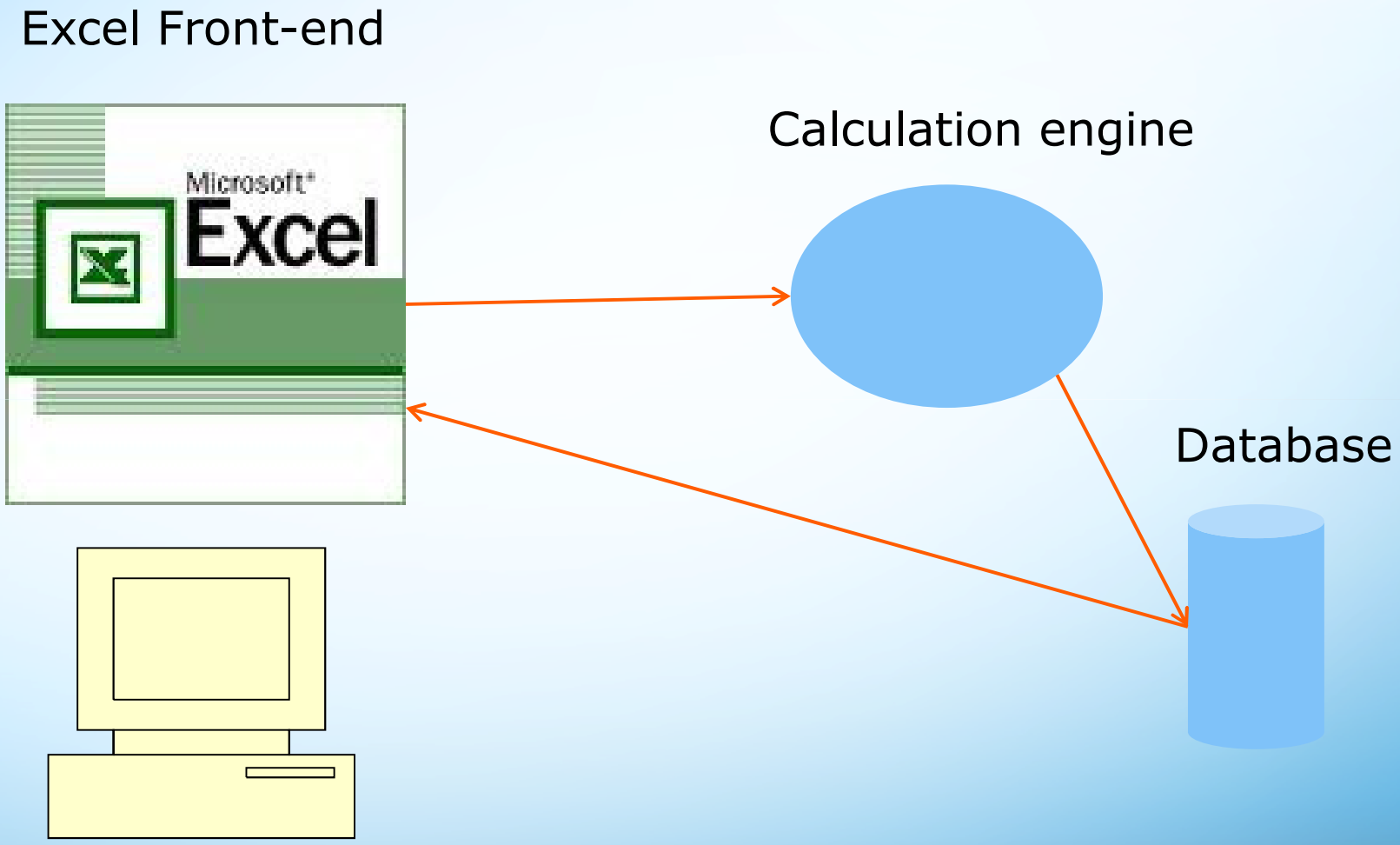
 *False*

✓ *True*

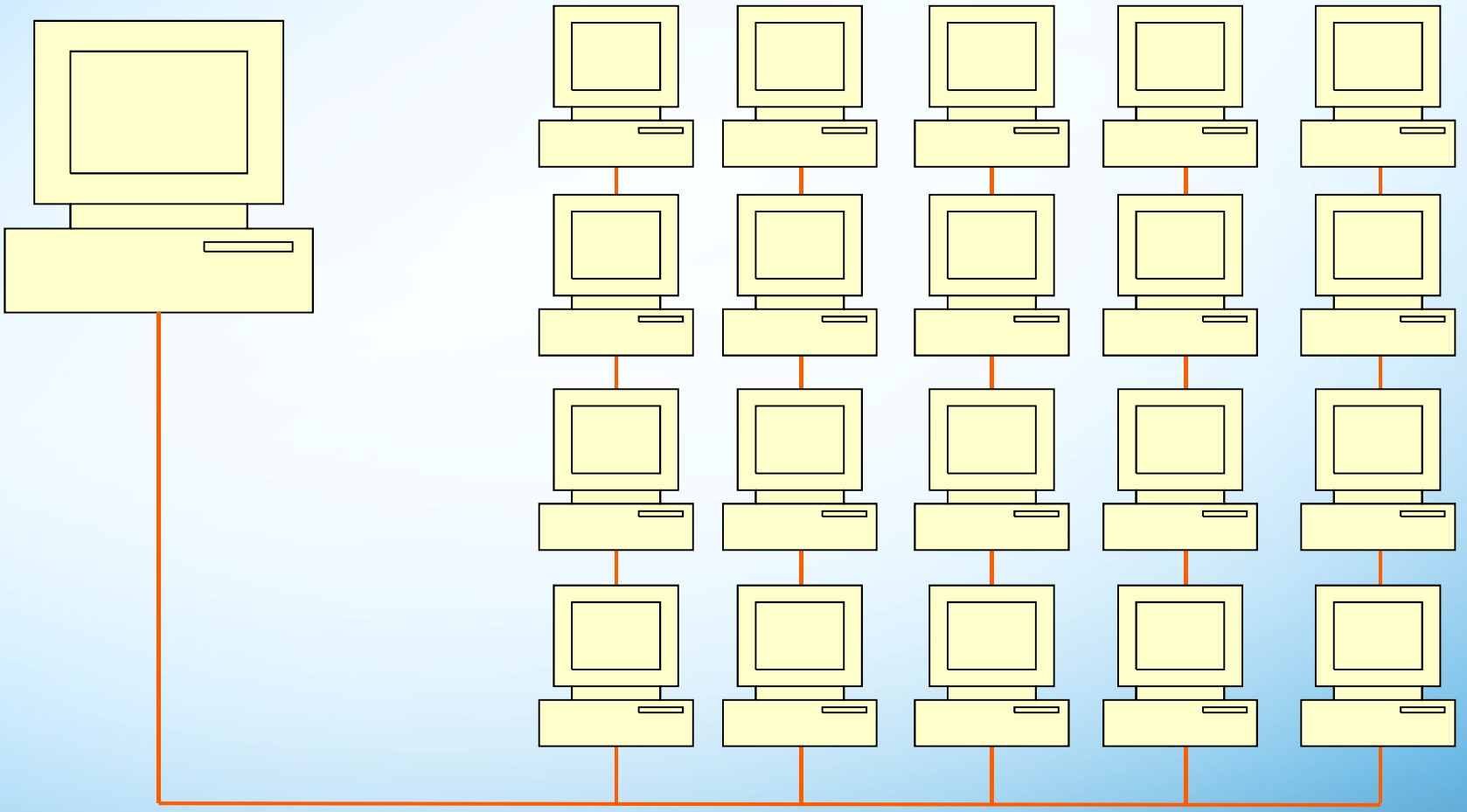
Case Study 2

Financial Institution

Interactive Mode



Computer Farm



Results

Build	Option	Speed	Comments
Msvc		24	Goes to 22.5 with SSE2 changes
iCL	/02	18	
	/PGO	16.9	
	NO pgo	18	
ADDING CILK FOR		25.5	CALLED 1 MILLION TIME Cilk loop has 96 iterations Sum product in each i elements
ARRAY NOTATION		19.5	USING REDUCE_ADD
Using MKL		17.2	
With more MKL		16	WITh CDF (loop of 96)
SSE2		14.4	
/pgo		13.8	

Reason for not parallelising

- Code construct needed some heavy lifting/reconstructing
- All class objects instantiated at start of program
  - Done for performance reasons
  - Means lots of global\shared
- Potentially suitable loops not doing enough work
  - see how adding Cilk slowed down the code

7. We, only let our parallel expert do this, and he's on holiday.

The Issue- Specialism

Moving to Parallel – a view from some developers

- Top 5 challenges
  - Legacy
  - Education
  - Tools
  - Fear of many cores
  - Maintainability

7. We, only let our parallel expert do this, and he's on holiday. ##

7. We, only let our parallel expert do this, and he's on holiday.

✓ *True*

8.Writing parallel programs is expensive.

The Issue – Return on Investment

Return on Investment

"Tip1: Just buy a faster machine!
First look at how much it will cost you to make your program parallel. If it will take say 2 months of coding, can you just buy a faster machine that will give you the speedup you want? Of course once you reach the limits of a machines speed, you are going to have to then do some parallelization."

Dr Yann Golanski, York

8.Writing parallel programs is expensive.

False

True

9. There are too many choices – ask me again in a couple of years

The Issue – Standardisation & Perception

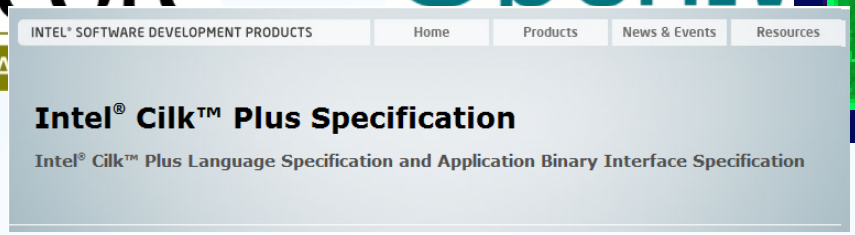
Standardisation

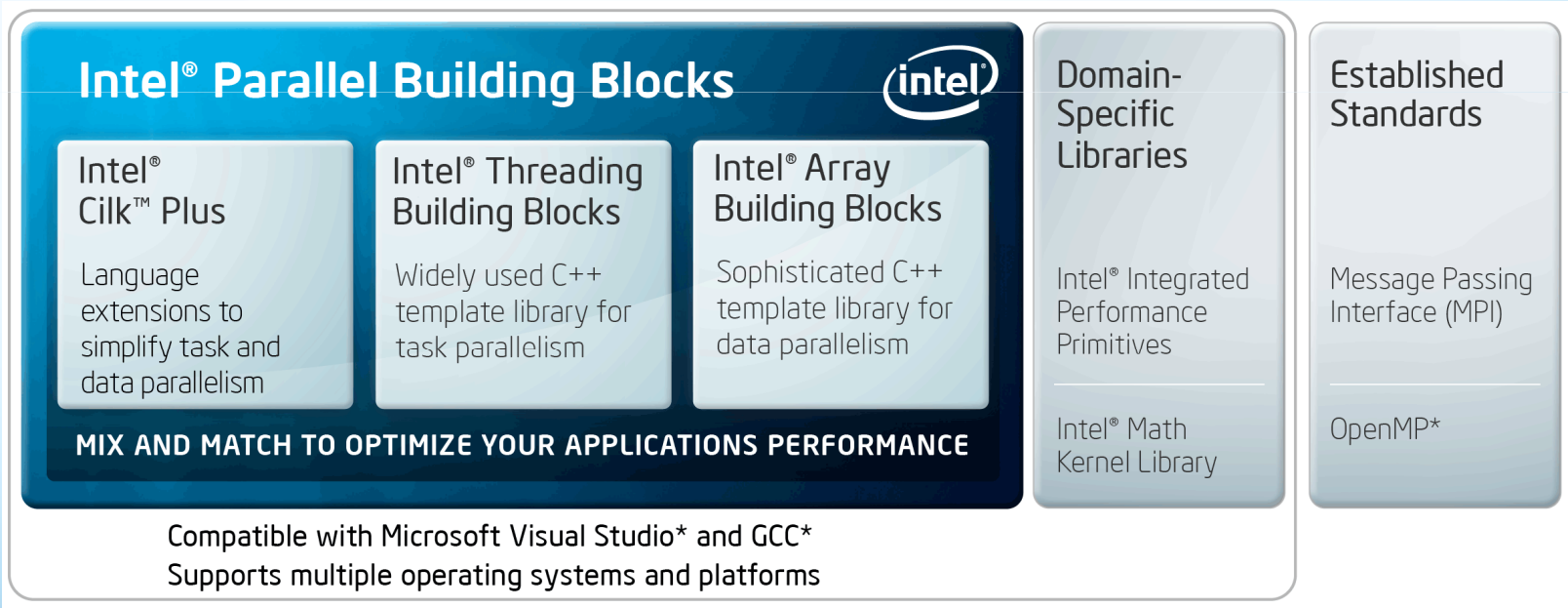
ANSI\ISO

Multicore ASSOCIA

OpenMP

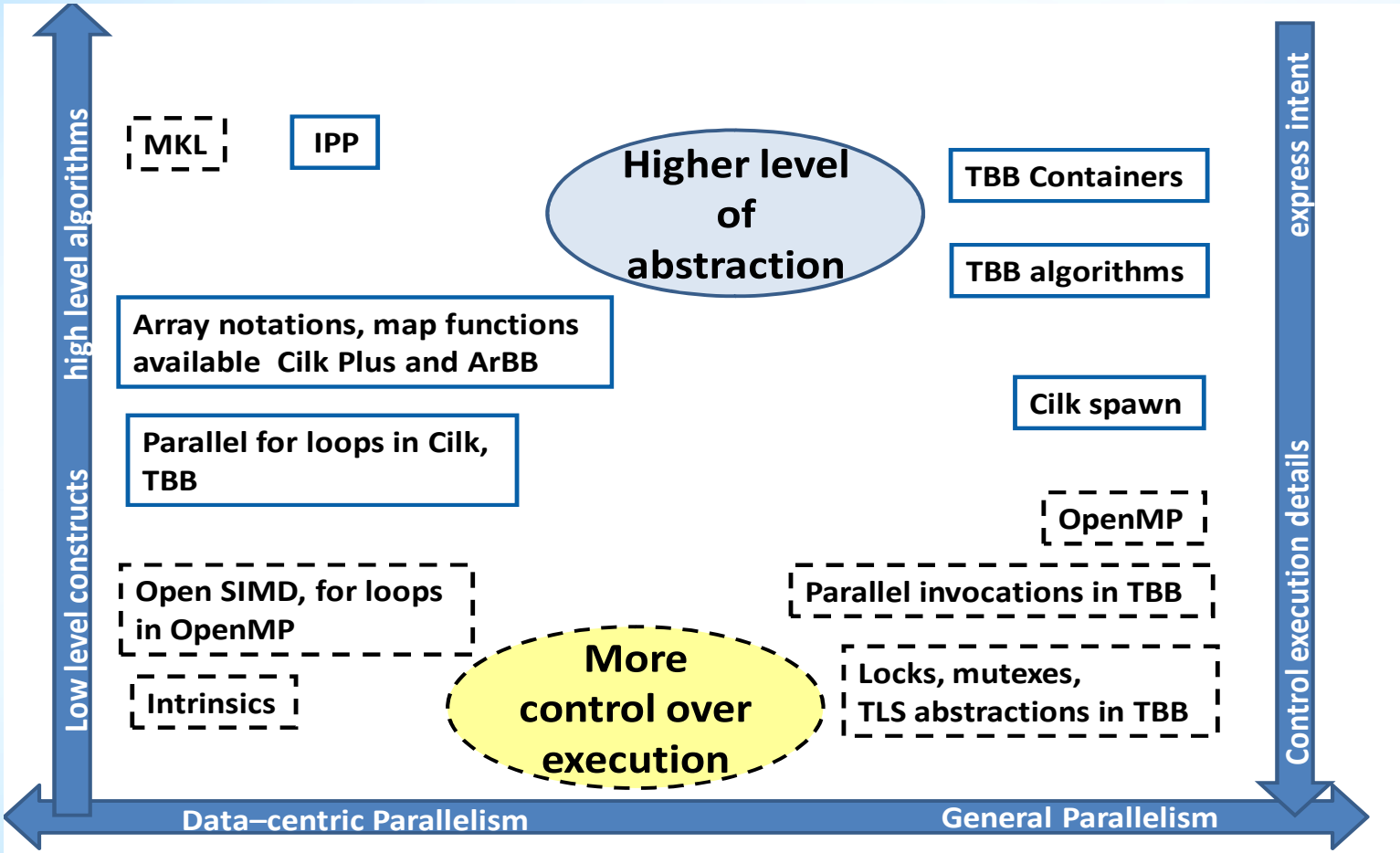
MPI





9. There are too many choices – ask me again in a couple of years

Perception



9. There are too many choices – ask me again in a couple of years ##

9. There are too many choices – ask me again in a couple of years

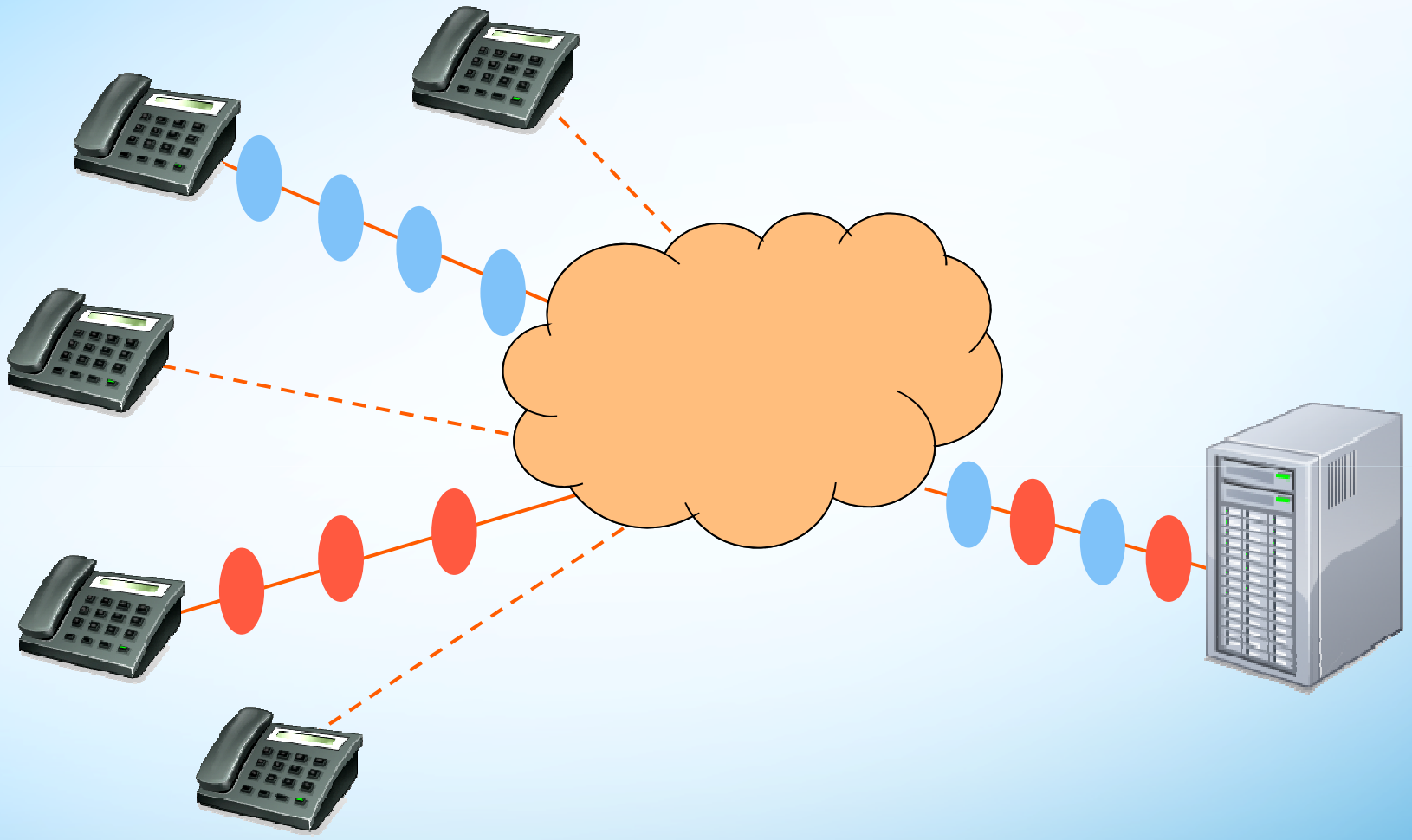
False

✓ *True*

Case Study 3

VOIP Telephone Exchange

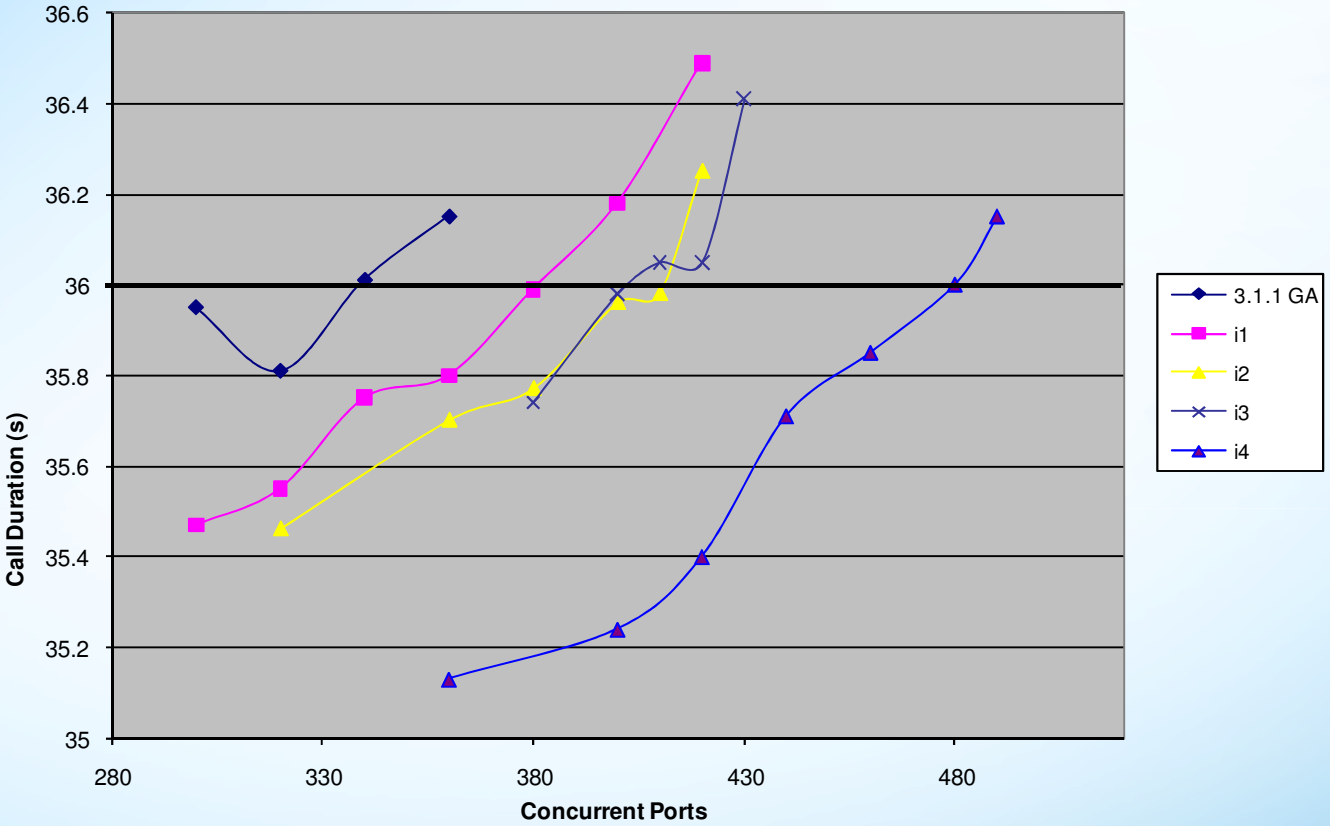
A Voice Over IP telephone Exchange



Goal

- Handle more concurrent calls
- By
  - Migrating to multicore
  - Improving the threading
  - Using VTune to profile

Results



Reason for not parallelising

- Already gained more than enough performance improvements
- Bug-fixed existing parallelism

Nine Reasons Why not to Program for Multicore

Architectural

1. Multicore is just a fad!
2. My Program will run just the same without any effort!
3. The CPU automatically makes things parallel – so I don't need to.

Programming Gotcha's

4. Parallel programming makes applications run slower!
5. No parallelism means no Errors. QED!
6. Multicore programs don't get faster on newer generations of CPU

Resource Issues

7. We, only let our parallel expert do this, and he's on holiday.
8. Writing parallel programs is expensive.
9. There are too many choices – ask me again in a couple of years

Nine Reasons Why not to Program for Multicore

Architectural

1. Multicore is just a fad!
2. My Program will run just the same without any effort!
3. The CPU automatically makes things parallel – so I don't need to.

Programming Gotcha's

4. Parallel programming makes applications run slower!
5. No parallelism means no Errors. QED!
6. Multicore programs don't get faster on newer generations of CPU

Resource Issues

7. We, only let our parallel expert do this, and he's on holiday.
8. Writing parallel programs is expensive.
9. There are too many choices – ask me again in a couple of years

Nine Reasons Why not to Program for Multicore

Architectural

1. Multicore is just a fad!
2. My Program will run just the same without any effort!
3. The CPU automatically makes things parallel – so I don't need to.

Programming Gotcha's

4. Parallel programming makes applications run slower!
5. No parallelism means no Errors. QED!
6. Multicore programs don't get faster on newer generations of CPU

Resource Issues

7. We, only let our parallel expert do this, and he's on holiday.
8. Writing parallel programs is expensive.
9. There are too many choices – ask me again in a couple of years

Thank you !

stephen.blair-chappell@intel.com

Optimization Notice

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel® Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

Legal Disclaimer

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.

http://intel.com/software/products